

Naren Technologies

Jenkins

In the previous chapter we have seen how development process is followed by developers for creating softwares and even maintain that software. So as the developers write code to create softwares, they also merge all that code into a centralised repository or Version Control System like Github. This code is pushed into repositories several times a day and over the period of time all of the code gets merged. Traditional software development methods don't dictate how frequently or regularly you integrate all of the source on a project. Programmers can work separately for hours, days, or even weeks on the same source without realizing how many conflicts (and perhaps bugs) they are generating.

Integration is painful

Agile teams produce workable and robust code in each iteration. All that code if built and evaluated returns lot of conflicts, bugs and errors. Developers needs to solve those conflicts and issues before moving to next iteration. The more programmers are sharing the code, the more problematic this is. For these reasons, agile teams often therefore choose to use Continuous Integration

Some Terminologies before we begin.

Source Code

All the code that developers writes to create the software is called as Source Code.

The Build Process

It is process by which source code is converted into a stand-alone form that can be run on a computer. For example, a source code written to develop a windows software when built will create a .exe or .msi file. Another example if a Java Source code is built it may create a .jar, .war or .ear file. This deployable piece of software is called as Artefact. The source code can be built, packaged and deployed manually. But there are some build tools that make developers life easy when it comes to building artefact or even deploying it. These are called as build automation tools.

Naren Technologies

Jenkins

Some build tools

1. Ant
2. Maven
3. Gradle
4. Msbuild
5. Nant

Unit Testing

Unit testing simply verifies the individual unit of code(mostly functions) works as expected. Developer along with writing the code will write the test cases that can be executed at the build time. Some test cases can be automatically generated. The objective of unit testing is to isolate a section of code(unit) and verify its correctness

What is Continuous Integration

Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control. CI encourages developers to share their code and unit tests by merging their changes into a shared version control repository after every small task completion. Committing code triggers an automated build system to grab the latest code from the shared repository and to build, test, and validate the full master branch (also known as the trunk or main).

The Problem

Developers will write the code and BUILD it in their local system. Once developers test the code and verify locally they push it to the centralised repository like github.

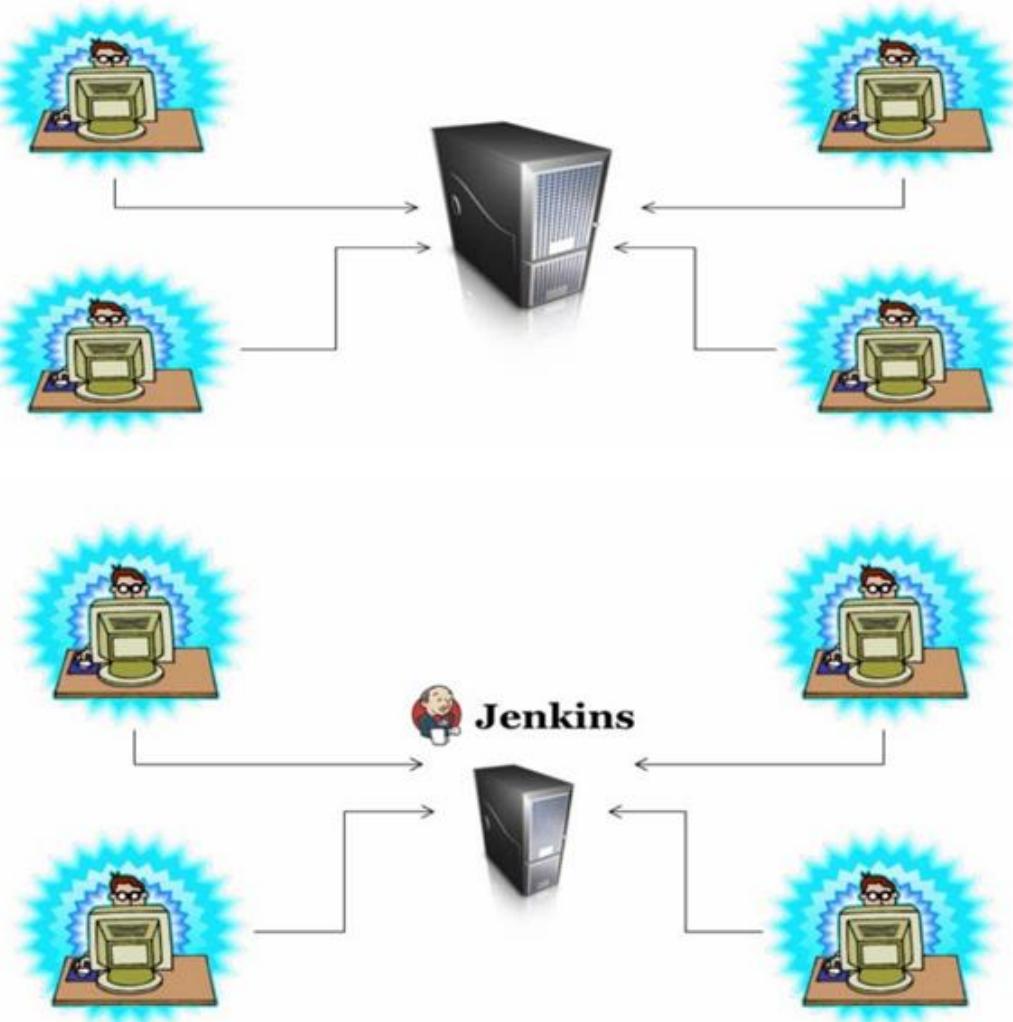
Similarly, all the developers would be pushing their code to VCS several times a day. Developers would be working in their own silos or caves and keep writing the code until they finish a particular task or the project. Now all the code which developers have pushed into the VCS, if built and tested will return lots & lots of conflicts, error due to which build will fail.

Naren Technologies

Jenkins

The Solution

To get around this very problem whenever the developer push the code to the VCS it should be fetched, built & tested by a build server at the same time.



This process repeated several times in a week or day or daily once is called as continuous integration. Developers code is continuously getting integrated so any point in time we have a workable software, if there is any issue in the build process the developers will get notified through email and they will fix the problem.

Naren Technologies

Jenkins

What is Jenkins

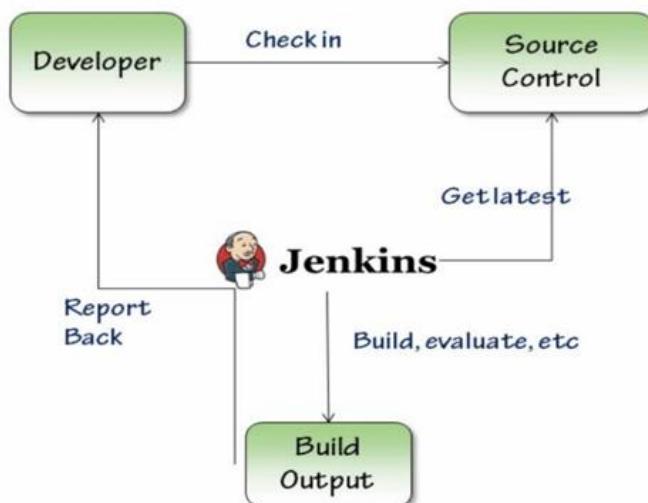
Jenkins is a continuous integration server which can fetch the latest code from VCS, build it, test it and notify it to the developers. Jenkins can do many more things apart from just being a CI server. It was originally known a Hudson, Oracle inc owns Hudson now. Jenkins is an open source project written by Kohsuke Kawaguchi. Jenkins is a java based web application server. As a prerequisite, we need to setup first Java on the machine to run Jenkins server.

Features of Jenkins

OpenSource

As jenkins is opensource there is lot contribution all around the world to the jenkins software. It has all the latest and greatest feature that developers integrating into it regularly.

Where Jenkins Fits In



Extensible

Jenkins comes with lot of goodies but its just not limited by that, Jenkins main power is its extensibility that can be achieved by installing plugins into it.

Naren Technologies

Jenkins

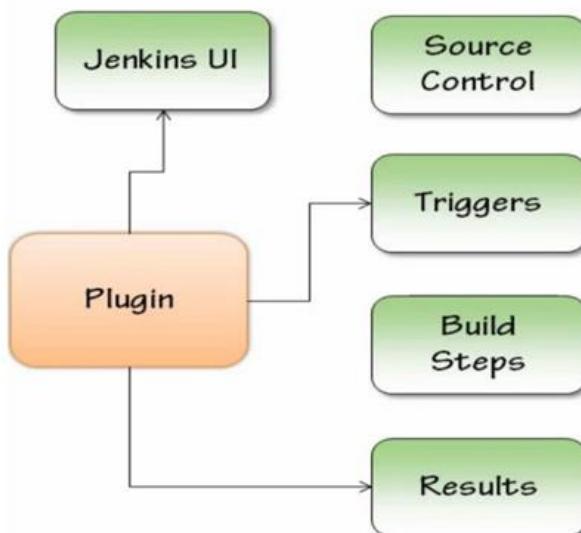
Jenkins opensource community has written tons of plugins, these plugins can do variety of tasks, like integration with external tools or servers.

1. VCS plugins – git, svn, subversion etc
2. Build plugins – Maven, ANT, Msbuild etc
3. Notification plugins – Email, chat, sms etc
4. Cloud plugins – Create cloud instances, deploy code to cloud services etc
5. Testing plugins – Code analysis, Unit test case, Static code analysis etc

The list of plugins is very long, whenever we want Jenkins to do some tasks just search for that plugin and most of the time you will find something.

For example, if you want jenkins to deploy java artefact to tomcat server, search for the plugin named “deploy to container”.

Plugin Architecture



Jenkins Setup

Jenkins can be installed on windows, Linux or Mac OS. Jenkins just needs java

Naren Technologies

Jenkins

software to run. In this tutorial, we will install jenkins on a ubuntu server. You can setup a vm or a cloud instance.

Prereqs

Java runtime environment/ JRE can be installed on the system but we will install JDK as we will setup maven moving along and build some java code. To Build the java code we will need JDK.

```
sudo add-apt-repository ppa:openjdk-r/ppa  
sudo apt-get update  
sudo apt-get install openjdk-8-jdk
```

Installing Jenkins

```
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
```

Install Git Client And Maven In Jenkins Server.

We will integrate Jenkins with github to download the source code. We are testing the java source code which will be built by Maven, so we also need to install Maven on Jenkins server. This is not a mandatory requirement to run Jenkins if you are not using git and maven.

Naren Technologies

Jenkins

```
sudo apt sudo apt-get install maven  
-get install git
```

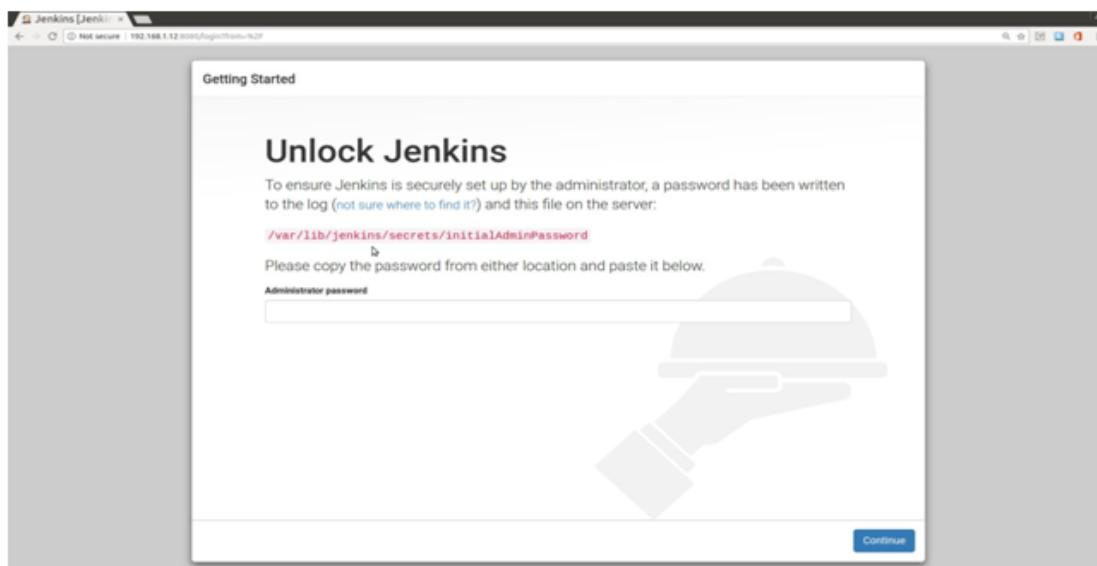
Accessing Jenkins

Jenkins runs on port 8080 by default

Open up a browser and use below url.

<http://jenkinsIP:8080>

Jenkins will set a random password to unlock jenkins setup.



The password would be stored in `/var/lib/jenkins/secrets/initialAdminPassword` file.

Read that file and get the password. Use that password to unlock jenkins.

Naren Technologies

Jenkins

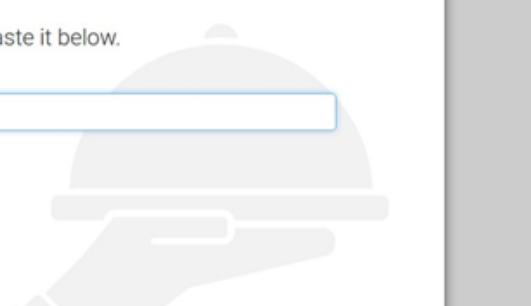
```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
f05498523c3547f89754376def663c6e  
vagrant@vagrant-ubuntu-trusty-64:~$ █
```

/var/lib/jenkins/secrets/initialAdminPassword

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

Administrator password

.....



Jenkins gives you an option to install some suggested plugins at the time of setup. You can select individual plugins or install suggested group of plugins. Select suggested plugin for now.

```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
f05498523c3547f89754376def663c6e  
vagrant@vagrant-ubuntu-trusty-64:~$ █
```

Getting Started

X

Customize Jenkins

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

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Naren Technologies

Jenkins

Getting Started

Create First Admin User

Username:	admin
Password:	*****
Confirm password:	*****
Full name:	Admin
E-mail address:	admin@devimranops.com

Jenkins 2.46.3

Continue as admin

Save and Finish

Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ build timeout plugin	✓ Credentials Binding Plugin	** Pipeline: Job ** Pipeline: Graph Analysis Plugin ** Pipeline: REST API Plugin ** JavaScript GUI Lib: Handlebars bundle plugin ** JavaScript GUI Lib: Moment.js bundle plugin Pipeline: Stage View Plugin ** Pipeline: Model API ** Pipeline: Basic Steps ** Pipeline: Stage Tags Metadata ** Authentication Tokens API Plugin ** Docker Commons Plugin ** Docker Pipeline ** Pipeline: Declarative Extension Points API ** Pipeline: Declarative Agent
✓ Timestamper	✓ Workspace Cleanup Plugin	✓ Ant Plugin	✓ Gradle Plugin	
✓ Pipeline	GitHub Organization Folder Plugin	✓ Pipeline: Stage View Plugin	Git plugin	
Subversion Plug-in	SSH Slaves plugin	✓ Matrix Authorization Strategy Plugin	✓ PAM Authentication plugin	
✓ LDAP Plugin	Email Extension Plugin	✓ Mailer Plugin		

Getting Started

Jenkins is ready!

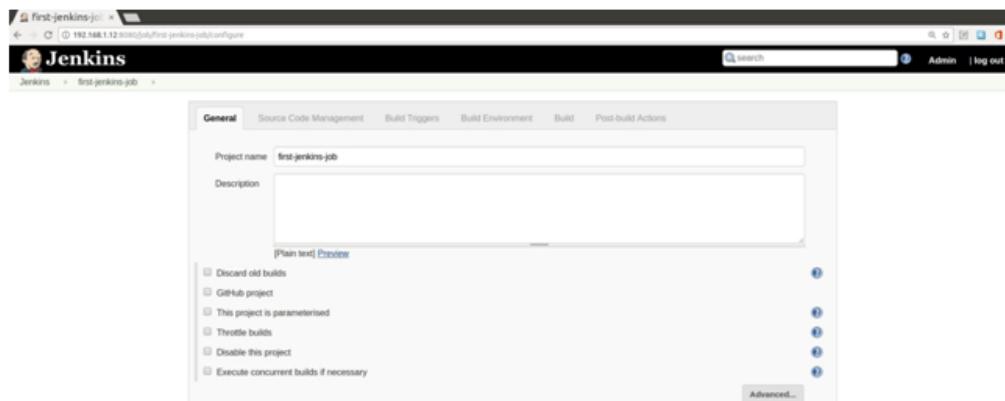
Your Jenkins setup is complete.

Start using Jenkins

Naren Technologies

Jenkins

Main Dash Board Of Jenkins.

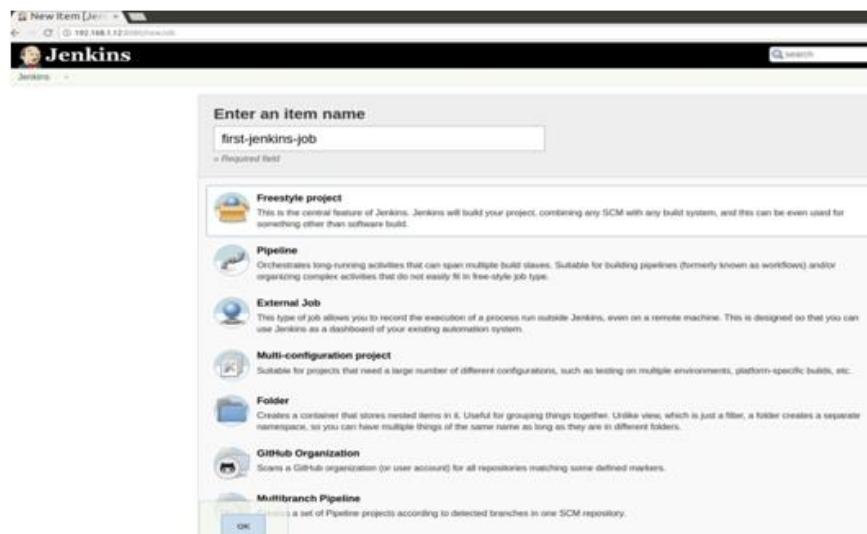


Creating first jenkins job.

Jenkins manages all the tasks into something called as jobs. Each job represents some set of activity like build process or software deploy or executing some scripts.

We will create a sample job to understand it better.

- 1.Click Create new jobs or New Item => Enter name – “first-jenkins-job” => Select Freestyle project => OK



Go to Build section => From drop down select shell => Enter some Linux commands “free -m” ; w => Save.

Build

Naren Technologies

Jenkins



After saving the project, we land up into the job's dashboard. Click Build Now to execute this project.



After the execution is completed we can check the history of the executed jobs and check its output from it.



Click on the blue ball to see the output of the job.

Jenkins > first-jenkins-job > #1

- Back to Project
- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete Build

Console Output

```
Started by user Admin
Building in workspace /var/lib/jenkins/workspace/first-jenkins-job
[first-jenkins-job] $ /bin/sh -xe /tmp/hudson8869116767161659177.sh
+ free -m
      total        used        free      shared      buffers      cached
Mem:   2001       1323        678          0         31        857
-/+ buffers/cache:    435      1566
Swap:      0          0          0
+ w
11:56:43 up 40 min, 2 users,  load average: 0.21, 0.06, 0.06
USER     TTY      FROM             LOGIN@  IDLE   JCPU   PCPU WHAT
vagrant pts/0    10.0.2.2          11:17  36:02  0.07s  0.01s sshd: vagrant [priv]
vagrant pts/1    10.0.2.2          11:22  34:02  0.05s  0.05s -bash
Finished: SUCCESS
```

Naren Technologies

Jenkins

4

We have seen from above example that setting up a jenkins job is not such a challenging task. But you need to know what information goes up in the Jenkins job. In the next task, we will create an actual build job.

Setup a Java build job with Maven

We are going to use a publicly available java source code from github to test the build job.

<https://github.com/wakaleo/game-of-life.git>

This java source code can be built by Maven. It also has the pom.xml file which maven needs to build the code.

Go to Jenkins main dashboard => Click New Item => Give a name to your job => Freestyle => OK

The screenshot shows the Jenkins 'New Item' creation interface. The 'Item Name' field is filled with 'first-maven-build'. Below the field, there are two options: 'Freestyle project' (selected, indicated by a blue border) and 'Pipeline'. A tooltip for 'Freestyle project' states: '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.' At the bottom of the screen, the 'Source Code Management' section is partially visible.

Source Code Management

Copy the URL of gameoflife source code from github.

Go to source code management section in Jenkins and select on Git

Enter the game of life github url.

Naren Technologies

Jenkins

Source Code Management

None
 Git

Repositories

Repository URL: <https://github.com/wakaleo/game-of-life.git>

Failed to connect to repository : Error performing command: git ls-remote -h https://github.com/wakaleo/game-of-life.git HEAD

Credentials: - none - [Add](#)

Advanced...
Add Repository

Branches to build

Branch Specifier (blank for 'any'): */master

Add Branch



This is a public repository so no credentials required and we are selecting master branch.

Create new file | Upload files | Find file | **Clone or download ▾**

...

Clone with HTTPS ⓘ | Use SSH

Use Git or checkout with SVN using the web URL.

<https://github.com/wakaleo/game-of-life> [Copy](#)

... and related

Download ZIP

5 years ago

...er classes

3 months ago



Go to build section => From drop down select Invoke top-level Maven targets => In the goals give "install" => Save

Naren Technologies

Jenkins



Click on Build Now => In Build history click on the loading symbol or the blinking ball to see the runtime console output.

```
11/17 KB  
14/17 KB  
17/17 KB  
19/17 KB  
Downloaded: http://repo.maven.apache.org/maven2/org/apache/cacknotch/commons/2.0.0/commons-2.0.0.jar (37 KB at 24.2 KB/sec)  
Downloaded: http://repo.maven.apache.org/maven2/org/apache/cacknotch/commons/2.0.0/commons-2.0.0.jar  
4/18 KB  
8/18 KB  
8/18 KB  
11/18 KB  
13/18 KB  
Downloaded: http://repo.maven.apache.org/maven2/org/apache/mache/lambda/lambda-0.1.0/lambda-0.1.0.jar (13 KB at 28.5 KB/sec)  
Downloaded: http://repo.maven.apache.org/maven2/org/apache/mache/lambda/lambda-0.1.0/lambda-0.1.0.jar  
3/3 KB  
Downloaded: http://repo.maven.apache.org/maven2/org/maven/commons/commons/2.7.5/commons-2.7.5.jar (9 KB at 4.8 KB/sec)  
Downloaded: http://repo.maven.apache.org/maven2/org/maven/commons/commons/2.7.5/commons-2.7.5.jar  
3/3 KB  
Downloaded: http://repo.maven.apache.org/maven2/org/maven/commons/commons/2.7.5/commons-2.7.5.jar (9 KB at 5.0 KB/sec)  
Downloaded: http://repo.maven.apache.org/maven2/org/maven/commons/commons/2.7.5/commons-2.7.5.jar  
4/18 KB  
8/18 KB  
8/18 KB  
10/18 KB  
Downloaded: http://repo.maven.apache.org/maven2/org/maven/commons/commons/2.7.5/commons-2.7.5.jar (10 KB at 5.0 KB/sec)  
Downloaded: http://repo.maven.apache.org/maven2/org/maven/commons/commons/2.7.5/commons-2.7.5.jar  
4/18 KB  
8/18 KB  
8/18 KB  
Downloaded: http://repo.maven.apache.org/maven2/commons/httpclient/commons_httpclient/3.1.2/commons_httpclient-3.1.2.jar  
Downloaded: http://repo.maven.apache.org/maven2/commons/httpclient/commons_httpclient/3.1.2/commons_httpclient-3.1.2.jar  
4/18 KB  
8/18 KB  
8/18 KB  
Downloaded: http://repo.maven.apache.org/maven2/commons/httpclient/commons_httpclient/3.1.2/commons_httpclient-3.1.2.jar (39 KB at 15.4 KB/sec)  
Downloaded: http://repo.maven.apache.org/maven2/commons/httpclient/commons_httpclient/3.1.2/commons_httpclient-3.1.2.jar  
4/18 KB  
8/18 KB  
8/18 KB  
Downloaded: http://repo.maven.apache.org/maven2/commons/httpclient/commons_httpclient/3.1.2/commons_httpclient-3.1.2.jar (8 KB at 13.0 KB/sec)
```

As per the POM.xml of this project, this particular maven job will do below mentioned tasks.

1. Download maven dependencies to build the job.
2. Build the java source code
3. Generate artifact
4. Archive artifact
5. Run unit test cases in the source code.

Naren Technologies

Jenkins

```
[INFO] Reactor Summary:  
[INFO]  
[INFO] gameoflife ..... SUCCESS [2:52.094s]  
[INFO] gameoflife-build ..... SUCCESS [1:04.176s]  
[INFO] gameoflife-core ..... SUCCESS [8.782s]  
[INFO] gameoflife-web ..... SUCCESS [2:39.458s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 7:05.313s  
[INFO] Finished at: Mon Jun 05 12:30:58 UTC 2017  
[INFO] Final Memory: 34M/82M  
[INFO] -----  
Finished: SUCCESS
```

Artefact Of Game Of Life Web Application.

After the build process is completed, you can find the artefact for this job in the Workspace of this job.

Workspace:

Workspace is the place where all the data of the job gets stored, e:g source code, artefacts etc. Every job in jenkins has its own workspace.

The screenshot shows the Jenkins interface for the project 'first-maven-build'. The top navigation bar includes links for 'Jenkins', 'first-maven-build', 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. The main content area is titled 'Project first-maven-build'. It features two links: 'Workspace' (represented by a folder icon) and 'Recent Changes' (represented by a notepad icon). The 'Workspace' link is underlined, indicating it is the active section.

Click on workspace => gameoflife-web => target => gameoflife.war gameoflife.war is the artefact that got generated and archived by maven build process. This artefact can be deployed to the java web application server like tomcat or jboss server.

Naren Technologies

Jenkins

 test-classic.com/war/c/gameoflife/webtests/controllers	
 gameoflife.war	3.04 MB view
 iaconon.exe	17.95 KB view

Jenkins Administration.

Once we had a little taste of jenkins and how to run jobs, we can understand how to administer Jenkins. Jenkins gives amazing features and its very flexible, we can setup jenkins as per our need. Jenkins can do variety of tasks apart from just being a CI server but we need to configure it as per our need. In coming section, we will see how jenkins is flexible and extensible.

You can open Jenkins admin settings by clicking on Manage Jenkins from main dashboard

Naren Technologies

Jenkins



[Build History](#)



[Manage Jenkins](#)

Manage Jenkins



[Configure System](#)

Configure global settings and paths.



[Configure Global Security](#)

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



[Configure Credentials](#)

Configure the credential providers and types.



[Global Tool Configuration](#)

Configure tools, their locations and automatic installers.



[Reload Configuration from Disk](#)

Discard all the loaded data in memory and reload everything from file system. Useful when you n



[Manage Plugins](#)

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



[System Information](#)

Displays various environmental information to assist trouble-shooting.



[System Log](#)

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



[Load Statistics](#)

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



[Jenkins CLI](#)

Access/manage Jenkins from your shell, or from your script.



[Script Console](#)

Executes arbitrary script for administration/trouble-shooting/diagnostics.



[Manage Nodes](#)

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

Naren Technologies

Jenkins

There are variety of settings you can tinker with. We will dig into it one by one.

Note: Sometimes after making any config change you may need to restart Jenkins. In the browser, you can use below url to restart jenkins.

<http://<JenkinsIP>:8080/restart>

Manage Plugins

Plugins are the most powerful feature of jenkins. You can customize jenkins as per your need by installing and setting up plugins. You can use jenkins to automate almost anything, it's just matter of the plugins you setup and there are wide variety of choices in plugins. We have already used some plugins in our build job like Git SCM, Invoke top level Maven target etc.

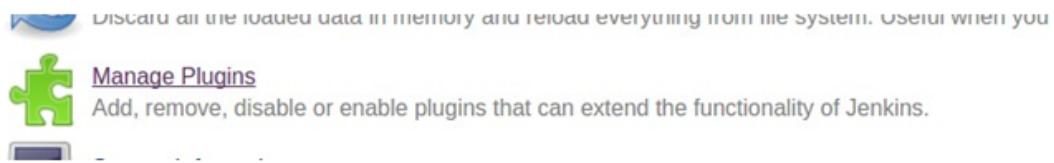
Some plugin comes by default installed in Jenkins and then you can install any plugin as per your choice and need.

Click [Manage plugins](#)

Naren Technologies

Jenkins

Click Manage plugins



There are four tabs

1. Update

If any plugin is outdated or a newer version of that plugin is available we can update the plugins from this tab.

2. Available

List of available plugins to install. Find your plugin from filter, just put a checkmark on your favourite plugin and click “Install without restart”. If the settings does not take effect restart jenkins server. Every plugin will have a wiki page, click on the plugin to read its wiki.

A screenshot of the Jenkins Manage Plugins Available tab. It shows a list of plugins with their names, versions, and descriptions. The "Available" tab is selected. Plugins listed include: Amazon EC2 Container Service plugin (version 1.11), Jenkins plugin to run dynamic slaves in a Amazon ECS/Docker environment; es2-cloud-slave (version 1.2), Amazon EC2 Container Service plugin with autoscaling capabilities (version 1.0), Amazon EC2 plugin (version 1.36), Deployment Dashboard Plugin for Jenkins (version 1.0.10), EC2 Fleet Jenkins Plugin (version 1.1.2), Support EC2 SpotFleet for Jenkins; Amazon ECR plugin (version 1.6), and Integrate Jenkins with Amazon ECI Container Registry (ECR). Buttons at the bottom include "Install without restart", "Download now and install after restart", "Check now", and a note "Update information obtained: 21 hr ago".

3. Installed

List of Installed plugin, if you choose to uninstall a plugin. Put a check mark and click uninstall

Updates	Available	Installed	Advanced	
Enabled	Name	Version	Previously installed version	Uninstall
	Ant Plugin Adds Apache Ant support to Jenkins	1.5		<button>Uninstall</button>

Naren Technologies

Jenkins

4. Advanced

Some time you sit behind a proxy server and don't have a direct internet connection.

That time you won't be able to see the list of Available plugin and won't be able to install it.

You can mention proxy settings in this page, restart Jenkins and then you will see the list of plugins to choose from. If you are java programmer and have written your own plugin, you can upload your plugin to Jenkins in .hpi format.

The screenshot shows the Jenkins 'Advanced' configuration page. At the top, there are tabs for 'Updates', 'Available', 'Installed', and 'Advanced', with 'Advanced' being the active tab. Below the tabs is a section titled 'HTTP Proxy Configuration' containing fields for 'Server', 'Port', 'User name', 'Password', and 'No Proxy Host'. There is also an 'Advanced...' button. Below this is a 'Submit' button. Underneath is a section titled 'Upload Plugin' with a note: 'You can upload a .hpi file to install a plugin from outside the central plugin repository.' It includes a 'File' input field with 'Choose file' and 'No file chosen' options, and a 'Upload' button.

Configure System



Configure System

Configure global settings and paths.

Settings option in this page depends on the number of plugin you have. Many plugins need some global configuration settings which can be modified from this page. More number of plugins more settings you will see in this page. From this page, we can change few Jenkins global settings like

Number Of Executor:

Number of jobs that can parallelly run in Jenkins If you have number of executor 2 and has initiated 3 jobs at the same time, then the third job will go in queue.

Naren Technologies

Jenkins

Environment Variables & Tools Path Can Also Be Set.

Email Notification:

SMTP server address and account details to send emails from Jenkins

Configure Global Security

Adding, removing and updating user and its permission can be handled from this page.

Enable security

TCP port for JNLP agents Fixed : Random Disable

Agent protocols...

Disable remember me

Access Control **Security Realm**

Delegate to servlet container

Jenkins' own user database

Allow users to sign up

LDAP

Unix user/group database

Authorization

Anyone can do anything

Legacy mode

Security Realm

First, establish the user authentication method. For smaller, more informal installations, you can use Jenkins' own user database. For enterprise installations, you will want to use your corporate service, which allows users to log in to Jenkins with their usual username and password.

Jenkins' Own User Database

This is the simplest authentication scheme--Jenkins maintains its own independent user database. People can sign up for their own accounts, and you as the administrator decide who can do what in Jenkins.

1. Select Jenkins' own user database
2. Place a check mark next to Allow users to sign up
3. Continue with Authorization, below. In particular, do not forget to press the Save

Naren Technologies

Jenkins

button at the bottom of the page

Active Directory On Linux Server

If Jenkins is running on a Windows server then it is better to install the Active Directory plugin. On a Linux host you have an option to either use the Active Directory plugin or an LDAP based authentication. To configure the LDAP to work with Active Directory, provide the following:

Server	<i>mydomaincontroller.mycompnay.com:389</i>
Root DN	<i>dc=<u>mycompnay</u>,dc=com</i>
User Search Filter	<i>sAMAccountName={0}</i>
Manager DN	<i>cn=<u>mymanageruser</u>,ou=<u>users</u>,ou=<u>na</u>,ou=<u>mycompany</u>,dc=<u>mycompany</u>,dc=com</i>

Naren Technologies

Jenkins

Manager DN	<code>cn=mymanageruser,ou=users,ou=na,ou=mycompany,dc=mycompany,dc=com</code>
Manager Password	*****

Note that the correct Manager DN value can vary greatly depending on your Active Directory set up.

UNIX NIS

To set up Network Information System:

1. Go to the Jenkins dashboard, usually `http://_server_:8080` or `http://_server_/jenkins:8080`, where server is the host on which Jenkins is running
2. Select Manage Jenkins, then Configure Global Security
3. Click Enable Security. The page will expand to offer a choice of access control
4. Select Unix user/group database#* Push the Test button (on the extreme right)
 1. If Success is displayed, everything is set up properly
 2. If not, follow the instructions to fix the problem and repeat

Naren Technologies

Jenkins

3. If you still do not succeed, push the Advanced button and specify Service Name sshd and repeat
5. Continue with Authorization, below. In particular, do not forget to press the Save button at the bottom of the page.

LDAP

See LDAP Plugin. Then continue with Authorization, below. In particular, do not forget to press the Save button at the bottom of the page.

Authorization

The Authorization section of the Configure Global Security page allows you to configure what users are allowed to do once authenticated.

Matrix-Based Security

Matrix-based security offers the most precise control over user privileges.

1. Select Matrix-based security as the Authorization
2. Give the Anonymous user only Overall Read access
3. In the text box below the matrix, type your user name (or the user name you plan to use when you register as a new Jenkins user) and click Add
4. Give yourself full access by checking the entire row for your user name
5. Repeat for other users who deserve full access. The configuration should look like the picture below:

The screenshot shows the Jenkins Global Security configuration page. Under the "Authorization" section, the "Matrix-based security" option is selected. A matrix table defines permissions for users across various Jenkins operations. The columns are: User/group, Overall, Job, Run, View, SCM. The rows are: Anonymous and kohsuke. The matrix values are as follows:

User/group	Overall	Job	Run	View	SCM							
	Administer	Read	Create	Delete	Configure	Build	Delete	Update	Create	Delete	Configure	Tag
Anonymous	<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"/>
kohsuke	<input checked="" type="checkbox"/>											

Below the matrix, there is a text input field labeled "User/group to add:" containing "kohsuke" and a "Add" button.

Naren Technologies

Jenkins

6. Click Save at the bottom of the page. You will be taken back to the top page. Now Jenkins is successfully secured.

7. Restart Jenkins (service Jenkins restart on Linux) If you set up a service like NIS, Active Directory or LDAP, you can now log in to Jenkins using your network credentials. If you are using Jenkins' own user database, create a user account for yourself:

Click the Login link at the top right portion of the page Choose Create an account Specify the user name you used in the above step, and fill in the rest If everything works smoothly, you are now logged on as yourself with full permissions. If something goes wrong, follow this to reset the security setting.

Global Tool Configuration

Jenkins gets integrated with variety of tools. In the build job, we have seen it gets integrated with git and maven. Similarly, you can have other tools in your OS installed or you can install it from this page. Also, sometimes we need different versions of Java or Maven or Git etc. We can manage multiple versions of the tools from this page as

The screenshot shows the Jenkins Global Tool Configuration interface. It includes sections for Maven Configuration, JDK, Git, Gradle, Ant, and Maven. The Maven Configuration section has 'Default settings provider' set to 'Use default maven settings'. The JDK section shows 'JDK installations' with an 'Add JDK' button. The Git section shows 'Git installations' with an entry for 'git' (Name: Default, Path to Git executable: git). The Gradle section shows 'Gradle installations' with an 'Add Gradle' button. The Ant section shows 'Ant installations' with an 'Add Ant' button. The Maven section is partially visible at the bottom.

Naren Technologies

Jenkins

Installing Maven 2.2.1

The screenshot shows the Jenkins 'Maven' configuration page under 'Manage Jenkins > Manage Nodes'. It displays a table for managing Maven installations. A new entry for 'Maven-2.2.1' is being added, with the 'Name' field set to 'Maven-2.2.1', the 'Install automatically' checkbox checked, and the 'Install from Apache' dropdown set to 'Version 2.2.1'. A red 'Delete Installer' button is visible on the right.

Maven
Maven installations
<input type="checkbox"/> Maven Name: Maven-2.2.1 <input checked="" type="checkbox"/> Install automatically <input type="checkbox"/> Install from Apache Version: 2.2.1 Add Installer

Adding A Windows Slave Node.

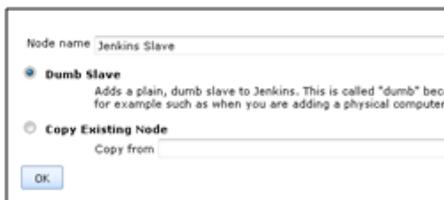


1. On your master machine go to Manage Jenkins>Manage Nodes.
2. New Node --

Naren Technologies

Jenkins

>Enter Node Name. 3. Select Dumb Slave --> Press OK.



4. Fill out the following:

- a. Set a number of executors (one or more) as needed.
- b. Set a Remote FS Root, a home directory for the master on the slave machine.
 - i. For a Windows slave, use something like: "C:\Jenkins\"
 - ii. TODO: add details.
- c. Select the appropriate Usage setting:
 - i. For an additional worker: Utilize this slave as much as possible
 - ii. For specialized jobs: Leave this machine for tied jobs only
- d. Launch Method:
 - i. An easy way to control a Windows slave is by using Launch slave agents via Java Web Start (Recommended for Windows)

Naren Technologies

Jenkins

- ii. TODO: add steps for other methods.
- e. Availability -->Keep this slave online as much as possible
- i. TODO: add details for each option.
- f. Press OK.

Name	Jenkins Slave
Description	
# of executors	2
Remote FS root	C:\Jenkins\
Labels	
Usage	Leave this machine for bid jobs only
Launch method	Launch slave agents via Java Web Start
Availability	Keep this slave on-line as much as possible
Node Properties	
<input type="checkbox"/> Environment variables	
<input type="checkbox"/> Prepare jobs environment	
<input type="checkbox"/> Tool Locations	
<input type="button" value="Save"/>	

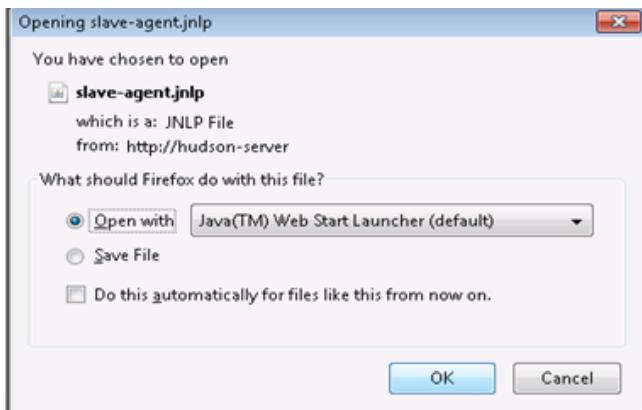
- 5. Now you need to connect your slave machine to the master using the following steps.
 - a. Open a browser on the slave machine and go to the Jenkins master server url (<http://yourjenkinsmaster:8080>).
 - b. Go to Manage Jenkins>Manage Nodes, Click on the newly created slave machine. You will need to login as someone that has the "Connect" Slave permission if you have configured global security.
 - c. Click on the Launch button to launch agent from browser on slave.



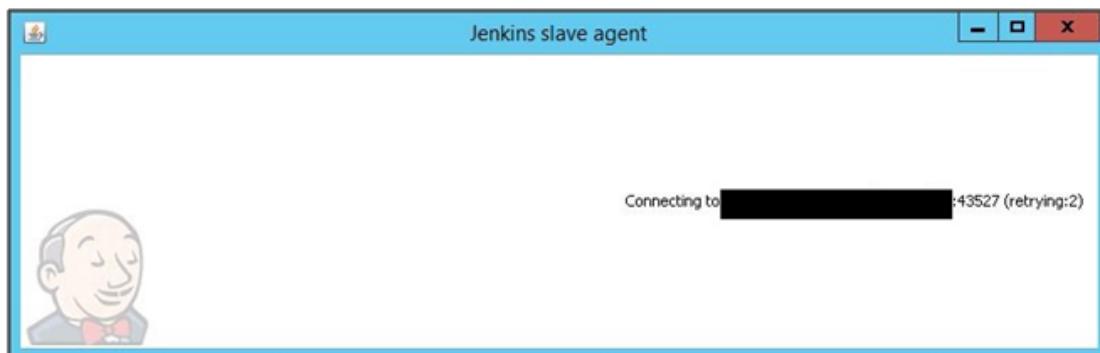
- d. Run the program.

Naren Technologies

Jenkins



If you encounter connection issue, then you could enlarge the popup windows to see the master port used and check your network configuration (firewall, port forward, ...)



- e. Now you should see the Slave machine connected under Nodes.
6. If you want the service to run on start-up of the slave machine do the following (Windows only directions):
 1. In the Slave agent program running on your slave machine, 2. click File -->Install as Windows Service.

Naren Technologies

Jenkins



- Note that this feature requires ".Net Framework 3.5"
3. Start, type Services and Select the Services program.
 4. Find Jenkins Slave in the list, double click to open.
 5. Select Startup type -->Automatic.
 6. Go to the Log On tab, change the Log on as to a user of your choice (Special user account Jenkins recommended).
 7. Make sure that auto login is set for the slave machine for the user account, then the VM (or physical computer) should connect and be available when needed.

Adding A Linux Slave Node.

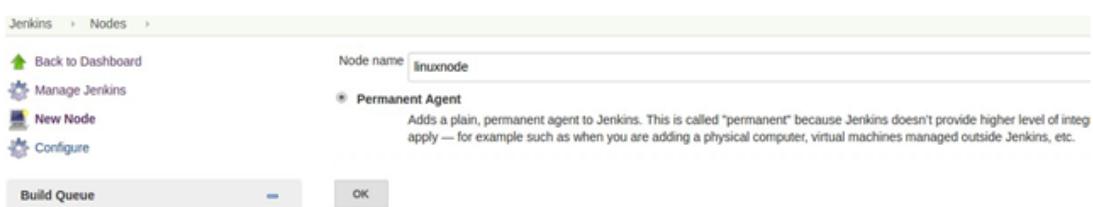
First setup the node.

Adding A Linux Slave Node.

First setup the node.

```
sudo apt-get install default-jre
sudo mkdir /opt/jenkins
sudo chown <username>:<groupname> /opt/jenkins -R
```

Manage nodes => New Node => node name => Permanent Agent/Dumb slave => OK



Naren Technologies

Jenkins

The screenshot shows the Jenkins Node Configuration page for a node named 'linuxnode'. The configuration fields include:

- Name: linuxnode
- Description: (empty)
- # of executors: 1
- Remote root directory: /opt/jenkins
- Labels: testlinuxnode
- Usage: Use this node as much as possible
- Launch method: Launch slave agents via SSH
- Host: 192.168.1.9
- Credentials: vagrant***** (vagrantlinuxnode) (selected)
- Host Key Verification Strategy: Known hosts file Verification Strategy
- Availability: Keep this agent online as much as possible

At the bottom, there are 'Node Properties' sections for Environment variables and Tool Locations, and a 'Save' button.

Continuous Integration Project.

We will setup a project that will build the artefact, version it and upload the versioned artefact to a Software Repository.

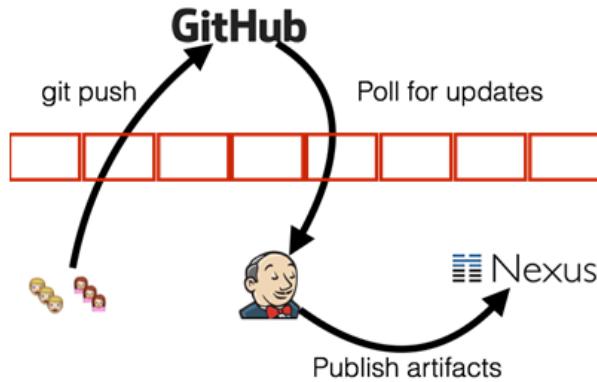
Nexus

Software repository or repository managers are becoming very central part of Continuous Integration and Continuous Delivery projects. We have seen in our second build job, whenever we run the build job it will create gameoflife.war artefact. This artefact will get replaced every time we run the job. If we generate an artefact that does not work or have any issues with it then we may need to go back to the previous version of the artefact. If we start versioning artefacts in Jenkins then we may fill up Jenkins disk space very quickly as these jobs runs several times in a day. For this we should have a mechanism of versioning and storing our versioned artefact to some centralised place.

For that very purpose we can use Nexus Repository Manager.

Naren Technologies

Jenkins



There are other benefits to it. It gives a hosted repository so anybody with right credentials can download the artefact.

For example, from our deployment scripts we can select our artefact from Nexus and download it to a target location like tomcat server.

Project Setup

Access Jenkins server from browser

<http://<jenkinsIP>:8080>

Jenkins Plugin Setup

Install plugins:- 1. Git plugin

Checkout source code from github. Integrates Jenkins with git

2. Zentimestamp plugin

Creates variable named \$BUILD_TIMESTAMP which can be used for versioning/naming our artifact.

After installing the plugin, we have to set its value from Configure System page.

Manage Jenkins => Configure System => Global properties.

Global properties
<input checked="" type="checkbox"/> Date pattern for the BUILD_TIMESTAMP (build timestamp) variable
Date and Time Pattern
yyyyMMddHHmm

3. Nexus plugin Uploads our versioned artifact to Nexus repository. Integrates Nexus with Jenkins

Create New Build Job

Naren Technologies

Jenkins

New item --> Enter project name --> Select freestyle project

The screenshot shows the Jenkins 'New Item' configuration page. The project name 'gol-mavenbuild' is entered in the 'Enter an item name' field. A 'Freestyle project' section is expanded, showing a brief description: '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.'

Select Git --> Enter GameOfLife git project URL (<https://github.com/wakaleo/game-of-life.git>)

The screenshot shows the 'Source Code Management' configuration for a Git repository. The 'Repository URL' is set to 'https://github.com/wakaleo/game-of-life.git'. The 'Branch Specifier' is set to 'master'. There are buttons for 'Advanced...', 'Add Repository', and 'Add Branch'.

life.git)

Add Build step --> Invoke top level maven project --> In Goals enter "install"

The screenshot shows the 'Build' configuration step. Under 'Invoke top-level Maven targets', the 'Goals' field contains 'install'. There is also an 'Advanced...' button.

Save --> Build Now.

Build Verification.

In your project's dashboard => Go to the workspace => gameoflife-web => target

You should see gameoflife.war.

Nexus Setup

We will setup nexus server on Centos in this tutorial.

Create a centos vm or cloud instance and login to it.

Follow below steps to setup Nexus

Naren Technologies

Jenkins

```
sudo -i
yum install -y java-1.8.0-openjdk.x86_64 vim wget
export RUN_AS_USER=root
wget http://www.sonatype.org/downloads/nexus-latest-bundle.tar.gz
sudo cp nexus-latest-bundle.tar.gz /usr/local/
cd /usr/local
sudo tar xvzf nexus-latest-bundle.tar.gz
sudo ln -s <nexus directory name> nexus
/usr/local/nexus/bin/nexus start
service iptables stop
```

Accessing Nexus dashboard

1. From browser hit URL <Nexus server IP>:8081/nexus.
2. Click login button and enter the credentials. (admin/admin123)
3. Create hosted repository named “gol-repo” with all default settings

The screenshot shows the Nexus Repository Manager OSS web interface. The main page displays a list of repositories, including Public Repositories, Apache Snapshots, Central, and Central M2 shadow. Below this, there is a modal or dropdown menu with the following options:

- Hosted Repository (selected)
- Proxy Repository
- Virtual Repository
- Repository Group

Naren Technologies

Jenkins

New Hosted Repository

Repository ID	goi-repo
Repository Name	goi-repo
Repository Type	hosted
Provider	Maven2
Format	maven2
Repository Policy	Release
Default Local Storage Location	
Override Local Storage Location	
Access Settings	
Deployment Policy	Disable Redeploy
Allow File Browsing	True
Include in Search	True
Publish URL	True
Expiration Settings	
Not Found Cache TTL	1440 minutes

Save Cancel

Configure Nexus Plugin To Push The Artefact To Nexus Repository.

Open the Jenkins build job => Add build step => Nexus artifact uploader

Add build step ▾

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Nexus artifact uploader
- Run with timeout
- Set build status to "pending" on GitHub commit

GameOfLife-prj Config 192.168.1.10:8080/jobs/GameOfLife-prj/configure Jenkins GameOfLife-prj

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

Nexus Details

Nexus Version	NEXUS2
Protocol	HTTP
Nexus URL	192.168.1.11:8081/nexus
Credentials	admin***** (Nexus)
GroupID	Dev
Version	\$BUILD_ID
Repository	gameoflife-repo
Artifact	Artifact
ArtifactId	\$BUILD_TIMESTAMP
Type	war
Classifier	
File	gameoflife-web/target/gameoflife.war

Add Save Apply

Naren Technologies

Jenkins



Console Output

```
File: gameoflife.war
Repository:gol-repo
Uploading: http://192.168.1.9:8081/nexus/content/repositories/gol-repo/Dev/$BUILD_TIMESTAMP/3/$BUILD_TIMESTAMP-3.war
10 % completed (319 KB / 3.2 MB).
20 % completed (639 KB / 3.2 MB).
30 % completed (958 KB / 3.2 MB).
40 % completed (1.3 MB / 3.2 MB).
50 % completed (1.6 MB / 3.2 MB).
60 % completed (1.9 MB / 3.2 MB).
70 % completed (2.2 MB / 3.2 MB).
80 % completed (2.5 MB / 3.2 MB).
90 % completed (2.8 MB / 3.2 MB).
100 % completed (3.2 MB / 3.2 MB).
Uploaded: http://192.168.1.9:8081/nexus/content/repositories/gol-repo/Dev/$BUILD_TIMESTAMP/3/$BUILD_TIMESTAMP-3.war (3.2 MB at 9.2 MB/s)
Uploading artifact gameoflife.war completed.
Finished: SUCCESS
```

Nexus Output

Login to nexus and verify the repository data. You should see a versioned artifact their.



Click On Its Hosted Url To Verify And Download The Artefact.



Name	Last Modified	Size	Description
Parent Directory			
201706061054-5.war	Tue Jun 06 10:54:12 UTC 2017	3192600	
201706061054-5.war.md5	Tue Jun 06 10:54:12 UTC 2017	32	

If you run this job multiple times you will see every time we get an artefact with a new name. In any point in time we can use older versions of the artefact if something breaks in newer version.

Static Code Analysis for Game of life Dev project.

What is Static Code Analysis?

Static Code Analysis (also known as Source Code Analysis) is usually performed as part of a Code Review (also known as white-box testing) and is carried out at the

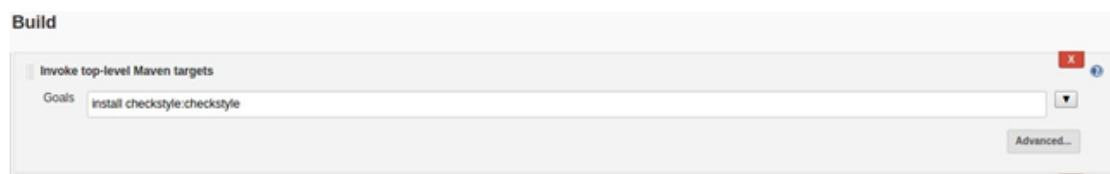
Naren Technologies

Jenkins

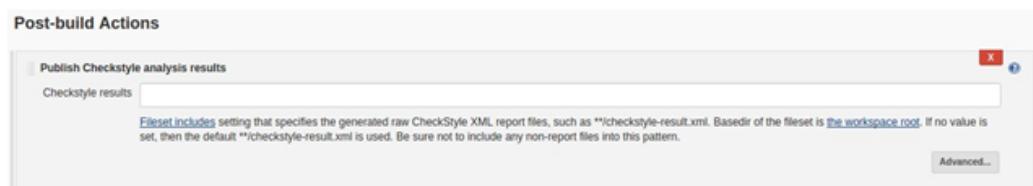
Implementation phase of a Security Development Lifecycle (SDL). Static Code Analysis commonly refers to the running of Static Code Analysis tools that attempt to highlight possible vulnerabilities within 'static' (non-running) source code by using techniques such as Taint Analysis and Data Flow Analysis.

Steps:

1. Install checkstyle plugin.
2. In Maven build step update the Goals as displayed below.



3. Click Post build action and select “Publish checkstyle analysis results”.



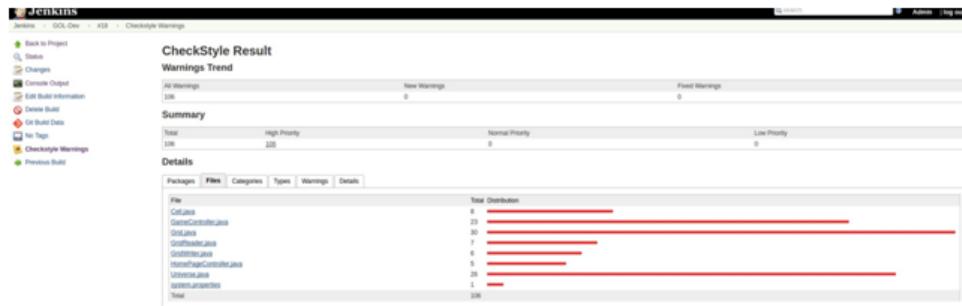
4. Save the project and run it minimum two times.
5. Go to Dev Jobs dashboard and see the checkstyle trend graph.



6. Click on checkstyle warnings and see the analysis in detail

Naren Technologies

Jenkins



Continuous Delivery with Jenkins

We are going to see now how deployment happens from Jenkins to tomcat server.

This is going to be a full-fledged Continuous delivery project by using Jenkins. We will need to setup two tomcat servers where we will deploy our artefact.

Tomcat Setup

Login as root user

Login as root user

```
sudo -i
apt-get update
apt-get install default-jre
apt-get install zip
wget https://www-eu.apache.org/dist/tomcat/tomcat-8/v8.5.15/bin/apache-
tomcat-8.5.15.zip
mv apache-tomcat-8.5.15.zip /opt
cd /opt
unzip apache-tomcat-8.5.15.zip
cd apache-tomcat-8.5.15
```

Give Tomcat Start & Stop Script Executable Permission

Naren Technologies

Jenkins

```
cd /opt/apache-tomcat-8.5.15/bin  
chmod u+x startup.sh  
chmod u+x shutdown.sh  
chmod u+x catalina.sh
```

Open Browser And Verify Tomcat Service

<http://tomcatIP:8080>

Set User, Password & Role For Tomcat Server.

```
cd /opt/apache-tomcat-8.5.15/  
vi conf/tomcat-users.xml
```

Replace Below Mentioned Content

Naren Technologies

Jenkins

```
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
-->
</tomcat-users>
```

With

```
-->
<role rolename="manager-script"/>
<role rolename="manager-gui"/>
<user username="tomcat" password="tomcat" roles="manager-script,manager-gui"/>
</tomcat-users>
```

Comment AntiResourceLocking Values.

Naren Technologies

Jenkins

```
vi webapps/manager/META-INF/context.xml
```

Replace

```
<Context antiResourceLocking="false" privileged="true">
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\\.\\d+\\.\\d+\\.\\d+|::1|0:0:0:0:0:0:1" />
</Context>
```

With

```
<Context antiResourceLocking="false" privileged="true">
<!--
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\\.\\d+\\.\\d+\\.\\d+|::1|0:0:0:0:0:0:1" />
-->
</Context>
```

Stop And Start Tomcat.

```
bin/shutdown.sh
bin/startup.sh
```

Naren Technologies

Jenkins

Verify The Settings

<http://tomcatIP:8080/manager/>

Username: tomcat

password: tomcat

You should see below mentioned screen.



The screenshot shows the Tomcat Web Application Manager interface. At the top, there's a logo of a yellow cat and the text "APACHE". Below that is the title "Tomcat Web Application Manager". A message bar says "Message: OK". The main area has a table titled "Applications" with columns: Path, Version, Display Name, Status, Sessions, and Commands. The table lists four applications:

Path	Version	Display Name	Status	Sessions	Commands
/	None specified	Welcome to Tomcat	Up	0	Start Stop Restart Undeploy Expire sessions with idle > 30 minutes
/docs	None specified	Tomcat Documentation	Up	0	Start Stop Restart Undeploy Expire sessions with idle > 30 minutes
/examples	None specified	Servlet and JSP Examples	Up	0	Start Stop Restart Undeploy (Expire sessions with idle > 30 minutes)
/host-manager	None specified	Tomcat Host Manager Application	Up	0	Start Stop Restart Undeploy Expire sessions with idle > 30 minutes

Project Setup

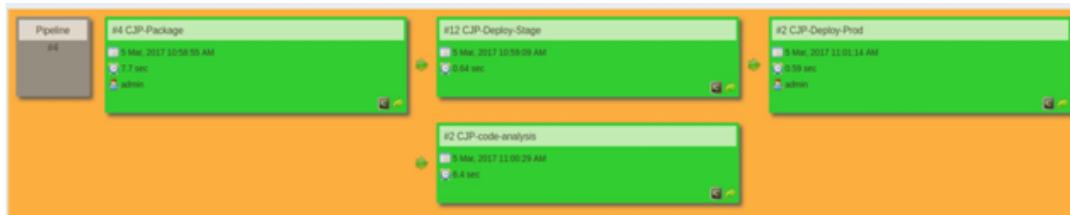
This project is a complete build pipeline which will build java SC, create war package, run static code analysis, deploy package to staging tomcat server & deploy to prod tomcat server.

Pipeline consist of below mentioned Jenkins job working together to create entire continuous delivery pipeline.

1. Package job – This job will checkout source code from git repository, use Maven to build the code, archive the artefacts, trigger Static code analysis job & staging deploy job
2. Static code analysis – This job will checkout source code from git repository, use Maven to run checkstyle code analysis and publish graph for STA.
3. Deploy to staging Tomcat server – This job will copy the war artefact from package job to deploy job, Deploy the artefacts to Staging tomcat server & trigger Prod deploy job.
4. Deploy to Prod Tomcat Server – This job will copy the war artefact from package job to deploy job, Deploy the artefacts to Staging tomcat server & trigger Prod deploy job.

Naren Technologies

Jenkins



Prerequisites:

1. Jenkins server: - Jenkins server should have openjdk 1.7, Git, Maven (Follow Jenkins setup doc)
2. Plugins: - git, maven, copy artifacts, deploy to container, checkstyle, build pipeline
3. Staging tomcat server: - Create a vm or ec2 instance with ubuntu OS & Follow tomcat setup doc.
4. Prod tomcat server:- Create a vm or ec2 instance with ubuntu OS & Follow tomcat setup doc.

Create Jenkins Jobs:

1. CJP-package: - Create a new job with CJP-package name, freestyle project.
Refer below mentioned screenshots to setup the job

The screenshot shows the 'Source Code Management' configuration for a Jenkins job. It is set up for a Git repository. The 'Repository URL' is set to <https://github.com/jleetutorial/maven-project.git>. The 'Branches to build' section specifies a 'Branch Specifier' of */master. Other settings include 'Repository browser' set to '(Auto)' and 'Additional Behaviours' set to 'Subversion'.

2. CJP-code-analysis: - Create a new job with CJP-code-analysis name, freestyle project.

Naren Technologies

Jenkins

Build

Invoke top-level Maven targets

Goals: clean package

Add build step ▾

Post-build Actions

Archive the artifacts

Files to archive: **/*.war

Advanced...

Build other projects

Projects to build: CJP-code-analysis, CJP-Deploy-Stage

- Trigger only if build is stable
- Trigger even if the build is unstable
- Trigger even if the build fails

Refer below mentioned screenshots to setup the job

Build

Invoke top-level Maven targets

Goals: checkstyle:checkstyle

Add build step ▾

Post-build Actions

Publish Checkstyle analysis results

Checkstyle results:

Fileset includes setting that specifies the generated raw CheckStyle XML report files, such as */checkstyle-result.xml. Basedir of the fileset is [the workspace root](#). If no value is set, then the default */checkstyle-result.xml is used. Be sure not to include any non-report files into this pattern.

Advanced...

Add post-build action ▾

Naren Technologies

Jenkins

Source Code Management

Repositories

Repository URL: <https://github.com/jleetutorial/maven-project.git>

Credentials: - none -

Advanced...

Branches to build

Branch Specifier (blank for 'any'): */master

Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

Subversion

3. CJP-Deploy-Stage:- Create a new job with CJP-Deploy-Stage name, freestyle project.
Refer below mentioned screenshots to setup the job

Build

Copy artifacts from another project

Project name: CJP-Package

Which build: Latest successful build

Stable build only

Artifacts to copy: **/*.war

Artifacts not to copy:

Target directory:

Parameter filters:

Flatten directories Optional Fingerprint Artifacts

Advanced...

Naren Technologies

Jenkins

Post-build Actions

Deploy war/ear to a container

WAR/EAR files: `**/*.war`

Context path: `/`

Containers

Tomcat 7.x

Manager user name: `tomcat`

Manager password: `*****`

Tomcat URL: `http://192.168.1.9:8080`

Add Container

Deploy on failure:

Build other projects (manual step)

Downstream Project Names: `CJP-Deploy-Prod`

Add Parameters

Deploy-Prod: - Create a new job with CJP-Deploy-Prod name, freestyle project.

Refer below mentioned screenshots to setup the job

Post-build Actions

Deploy war/ear to a container

WAR/EAR files: `**/*.war`

Context path: `/`

Containers

Tomcat 7.x

Manager user name: `tomcat`

Manager password: `*****`

Tomcat URL: `http://192.168.1.10:8080`

Add Container

Deploy on failure:

Add post-build action

Setup Build Pipeline View

Naren Technologies

Jenkins

Build

Copy artifacts from another project

Project name: CJP-Package

Which build: Latest successful build

Stable build only

Artifacts to copy: **/*.war

Artifacts not to copy:

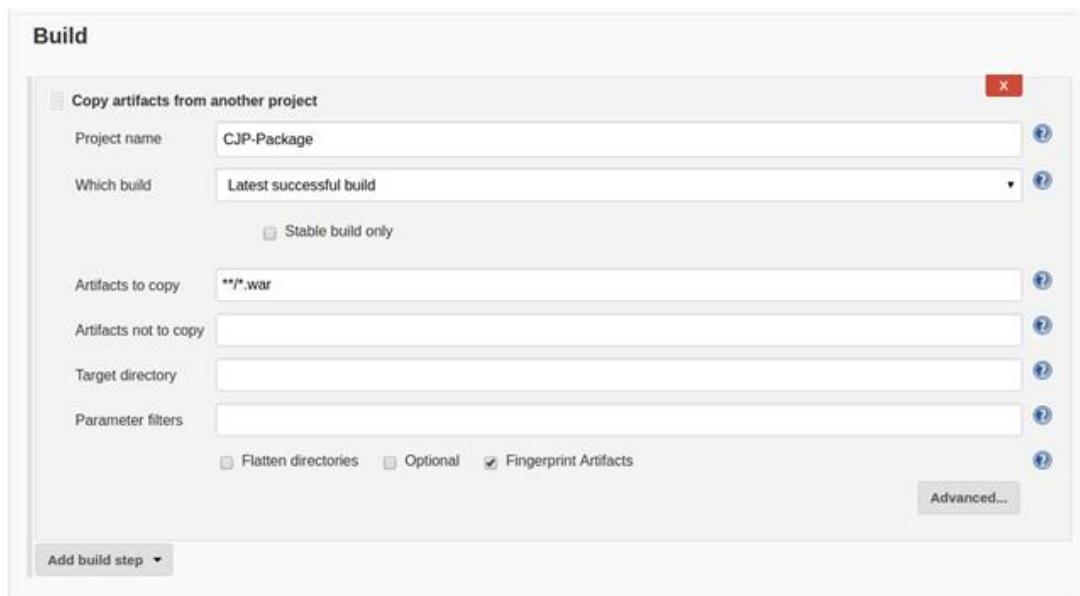
Target directory:

Parameter filters:

Flatten directories Optional Fingerprint Artifacts

Advanced...

Add build step ▾



Go to Jenkins main dashboard

Click on the plus symbol



Select Build Pipeline View, give pipeline view a name and click OK

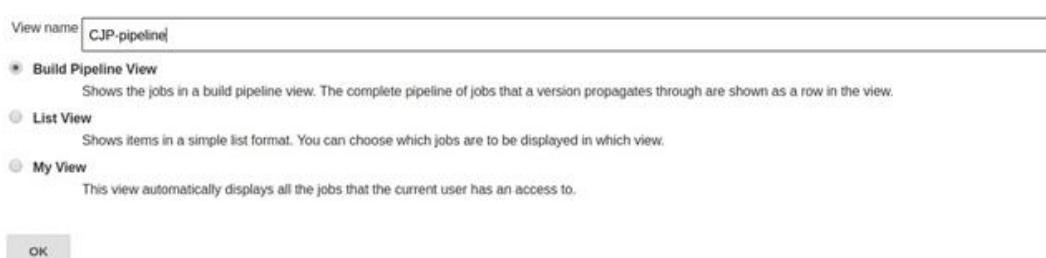
View name: CJP-pipeline

Build Pipeline View
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View
This view automatically displays all the jobs that the current user has an access to.

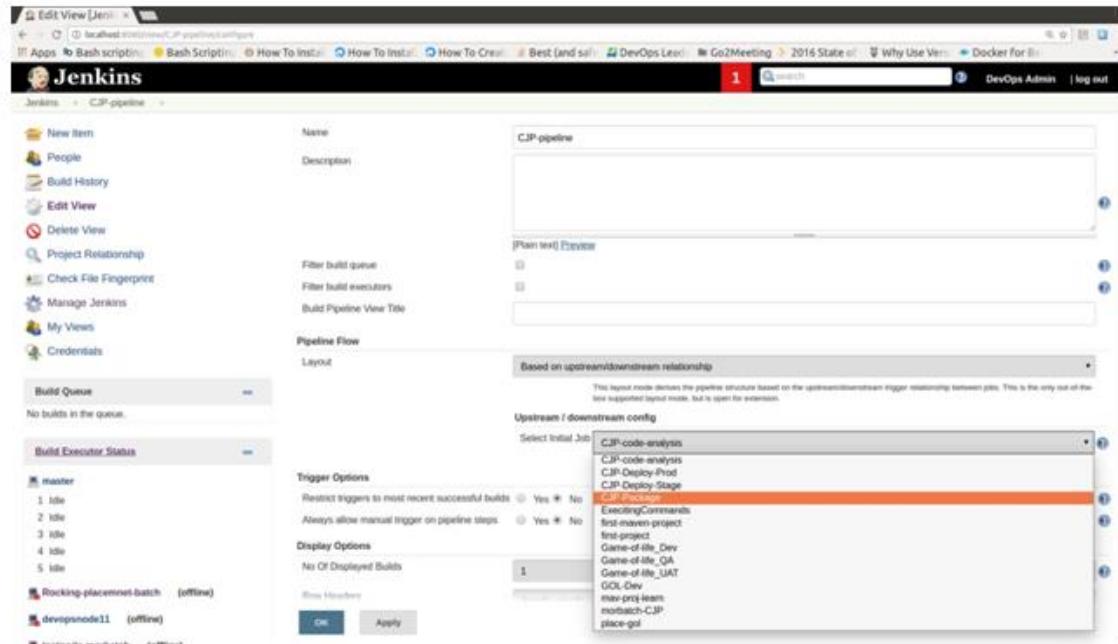
OK



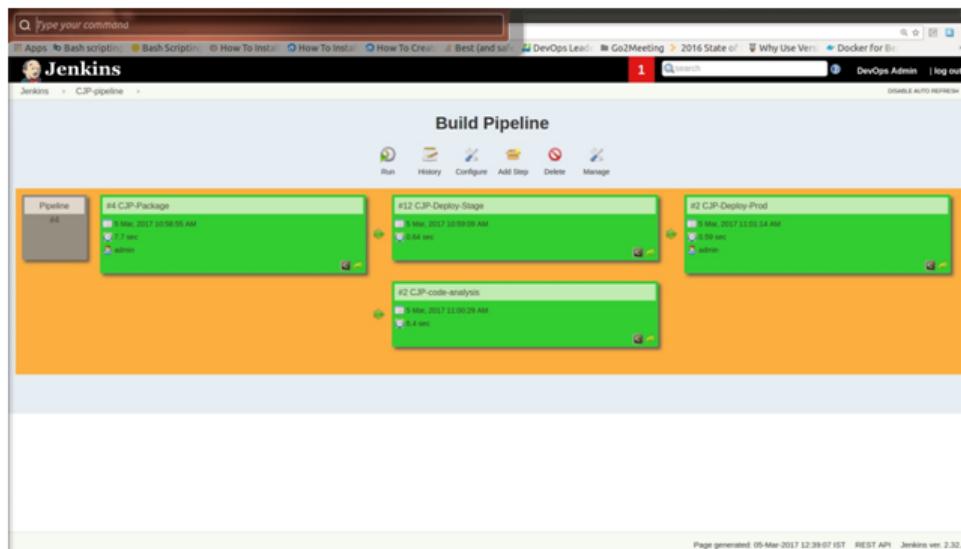
Select initial Job – CJP-Package & click OK

Naren Technologies

Jenkins



Click run and verify if the entire pipeline works



Jenkins Build Triggers

So far we have seen executing Jenkins jobs manually by clicking on the Build Now button.

But there are other ways by which the jobs can be executed those are called as Build

Naren Technologies

Jenkins

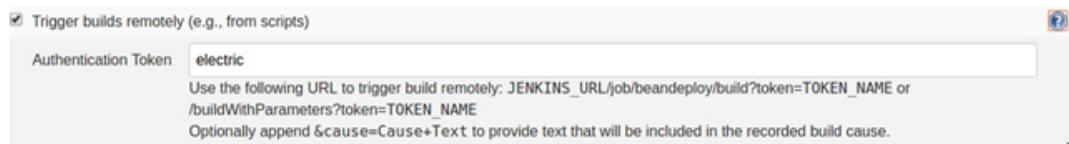
Triggers.

Build Triggers

- Trigger builds remotely (e.g., from scripts) 
- Build after other projects are built 
- Build periodically 
- Poll SCM 

Trigger Build Remotely

We can use this trigger If we want to run the job from a remote location like from a script or from Github or any other tool that can hit the URL. One typical example for this feature would be to trigger new build from the source control system's hook script, when somebody has just committed a change into the repository, or from a script that parses your source control email notifications. You'll need to provide an authorization token in the form of a string so that only those who know it would be able to remotely trigger this project's builds .



Now you can use the URL with the token `JENKINS_URL/job/beandeploy/build?`

`token=TOKEN_NAME` to call this job from

1. CLI like curl command

1. `curl http://username:password@192.168.1.9:8080/job/beandeploy/build?`
`token=electric`

2. Through a script

3. From a browser

4. Github webhooks

1. In github repo go to Settings => Webhooks => Enter Jenkins URL and Token => Add Webhook.

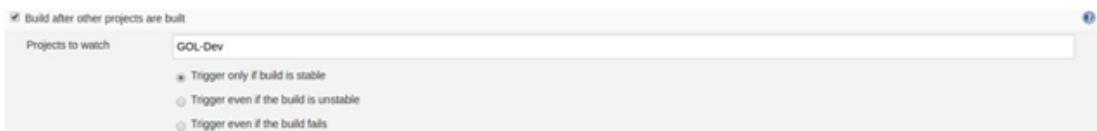
Now whenever there is a commit in the git repo it will trigger the Jenkins job by hitting Jenkins remote URL.

Naren Technologies

Jenkins

Build After Other Projects Are Built.

This is to set the job as downstream for another job. Mention the upstream job name, once the upstream job gets completed successfully our job will get triggered.



Build Periodically

If we want to schedule our job to get executed for example every night 8 pm then we can use a cronjob format to specify the time.



This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0-59)

HOUR The hour of the day (0-23)

DOM The day of the month (1-31)

MONTH The month (1-12)

DOW The day of the week (0-7) where 0 and 7 are Sunday.

Poll SCM

It's similar to cronjob but instead of running the job at the interval it will go and check if there is any new commit in the VCS repo like github repo's and then execute the job.

Sonarqube Integration Jenkins.



Naren Technologies

Jenkins

SonarQube is an open source platform for continuous inspection of code quality. This will reports on duplicated code, coding standards, unit tests, code coverage and code complexity.

SonarQube can Provides a central place to view and define the rules used during analysis of projects. These rulesets are organized in quality profiles. Every member of the organization can see which rules are applied to their project.

Every project administrator can choose which quality profile is used for the project.

Required Plugins

1. SonarQube scanner

2. Maven

SonarCode scanner process in Jenkins

1. Pull code From Git.

2. Build and Run Unit Tests

3. SonarQube analysis with sonarqube scanners.

4. Push SonarQube analysis coverage reports to SonarQube Dashboard.

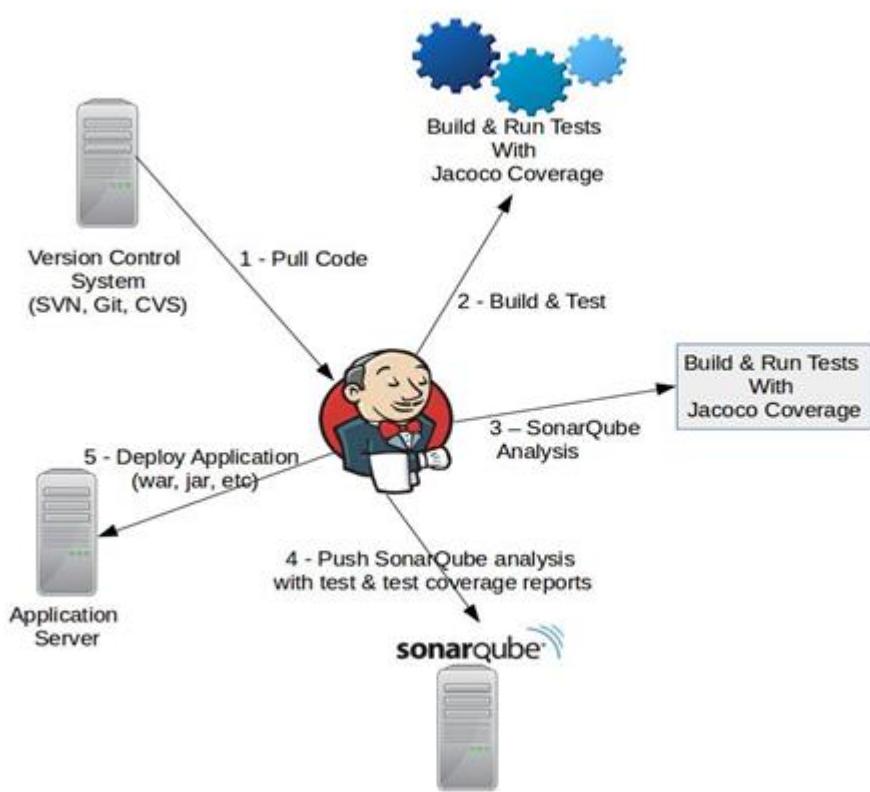
5. We can store the SonarQube data in DataBase (Mysql).

6. Deploy applications to server.

SonarQube Lifecycle

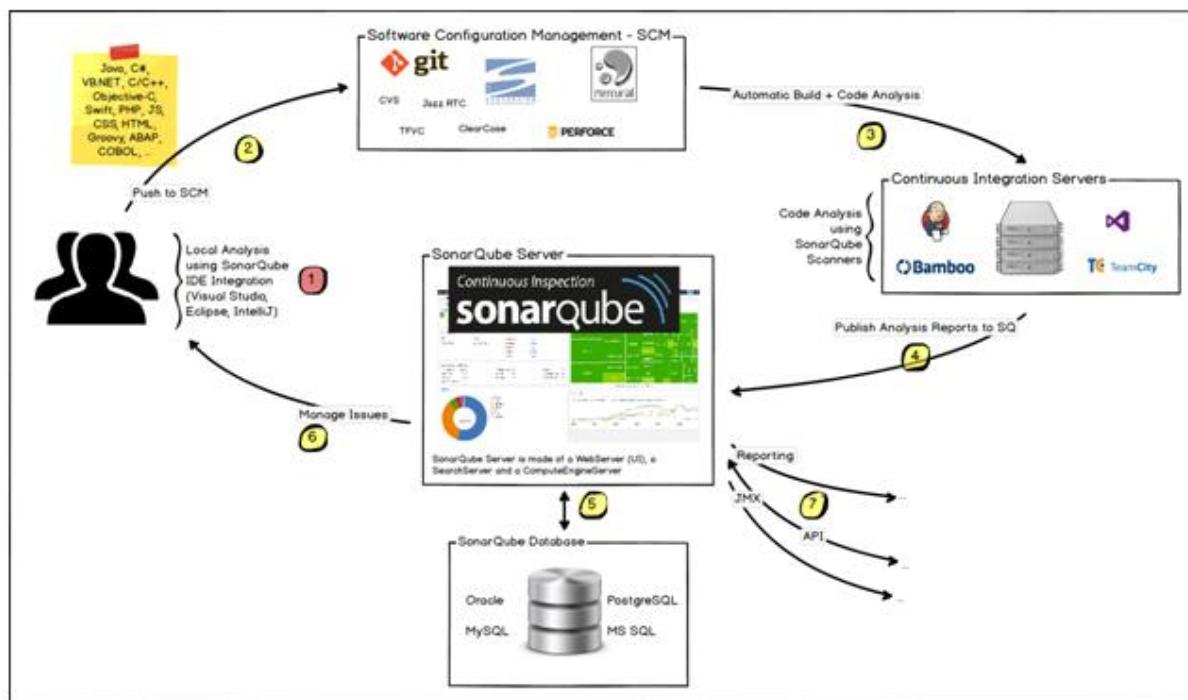
Naren Technologies

Jenkins



Naren Technologies

Jenkins



Prerequisite For Exercise

For the exercise of SonarQube we need two ec2 instances with below details.

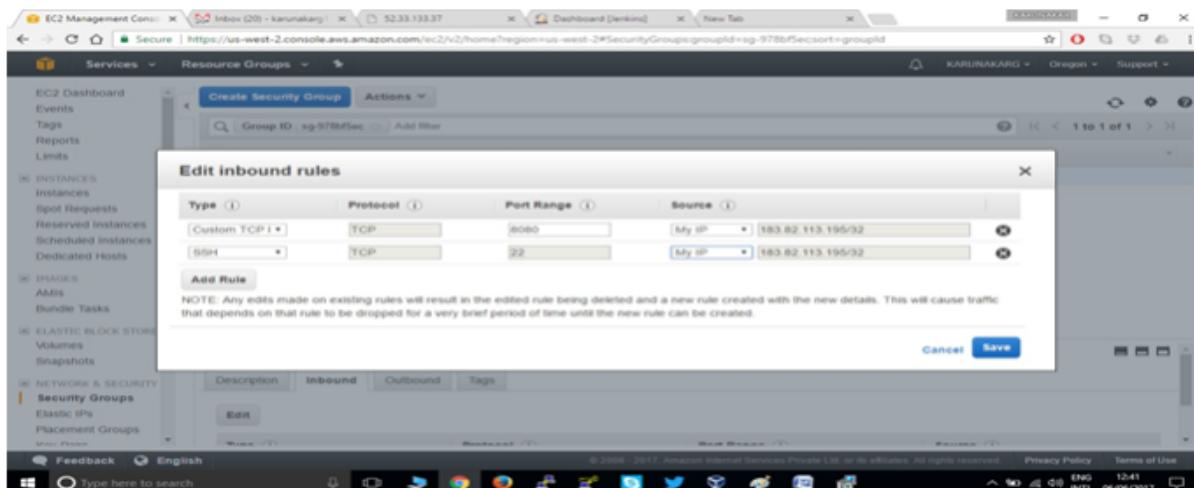
For Jenkins Instance

1. OS – ubuntu 16.10

Naren Technologies

Jenkins

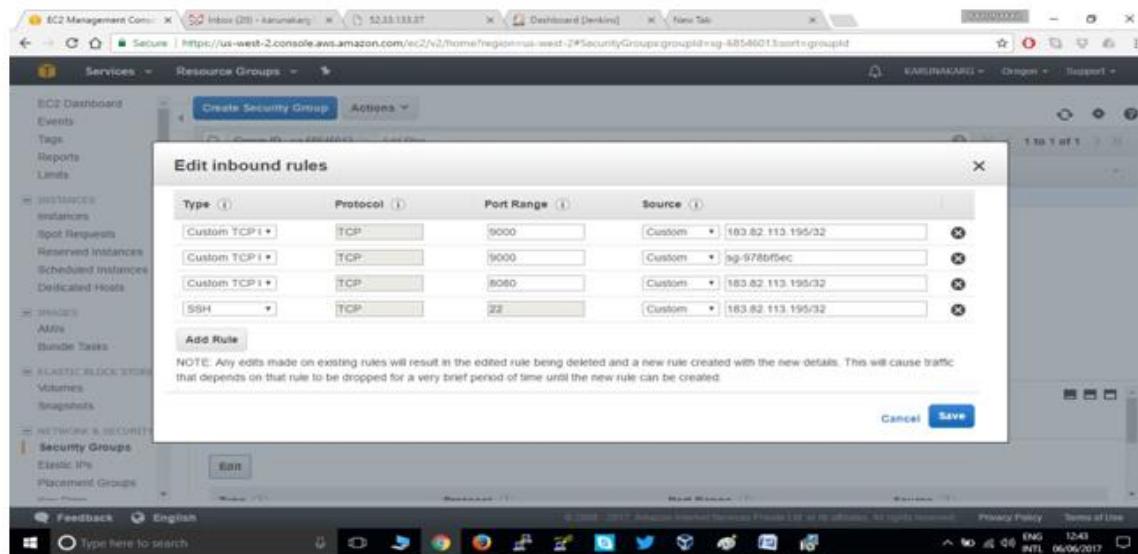
2. Type - t2.micro
3. Securitykey for Jenkins (.pem)
- Security Group for Jenkins
- Type Custom TCP ->Port 8080->source MyIP
- SSH -->Port 22 -> source MyIP



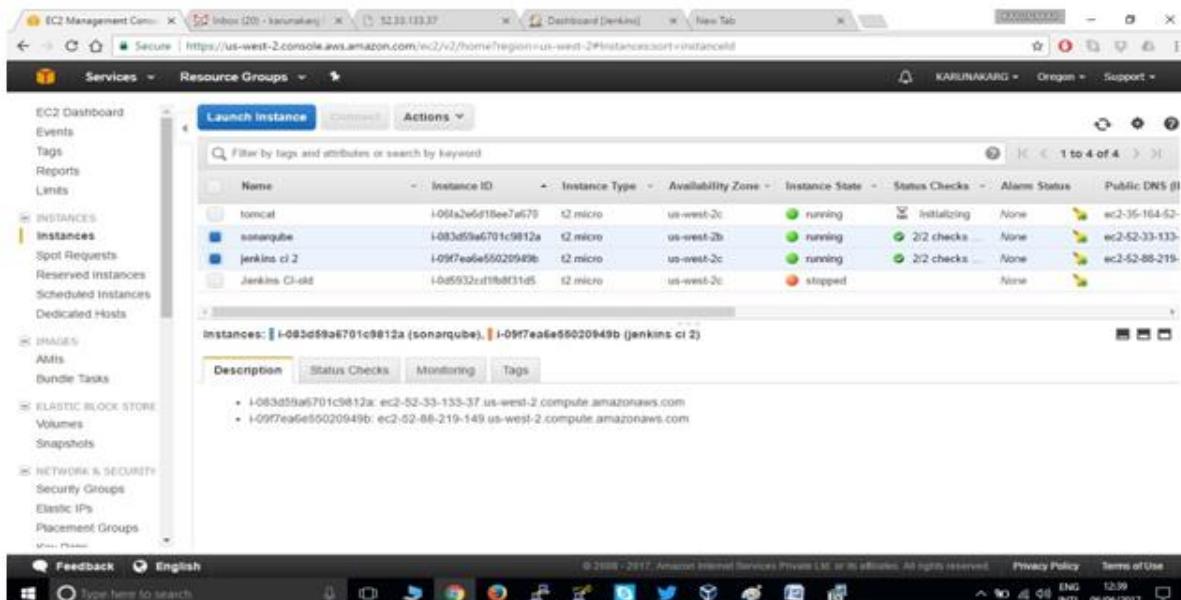
- For SonarQube Instance.
4. OS - ubuntu 16.10
 5. Type - t2.medium
 6. Securitykey for SonarQub (.pem)
 - Security Group for SonarQube
 - Type Custom TCP ->Port 9000->source MyIP
 - Type Custom TCP ->Port 8080->source JenkinsSg
 - Type Custom TCP ->Port 8080->source MyIP
 - SSH -->Port 22 -> source MyIP

Naren Technologies

Jenkins



In Next Slides, we are showing that complete setup of SonarQube



SonarQube Installation Procedure In Ubuntu 16.10 Prerequisites To Setup SonarQube:

EC2 Ubuntu Instance,

1. Java-8-oracle,
2. Sonarqube version 5.6.6,
3. mysql-server 5.7.

Naren Technologies

Jenkins

Adding Java PPA to Our Ubuntu repository

```
add-apt-repository -y ppa:webupd8team/java
```

Updating apt repository

```
apt-get update
```

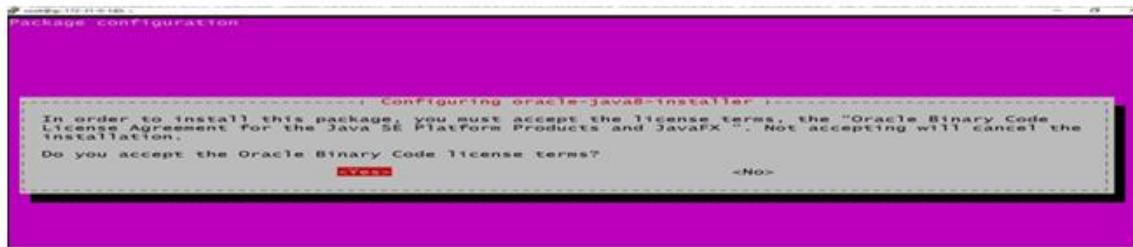
Installing java on EC2 Ubuntu 16.10 Instance

```
apt-get -y install oracle-java8-installer
```

Here Press on “OK”



Here Press on “Yes”



Here We are Creating soft link for java = default java

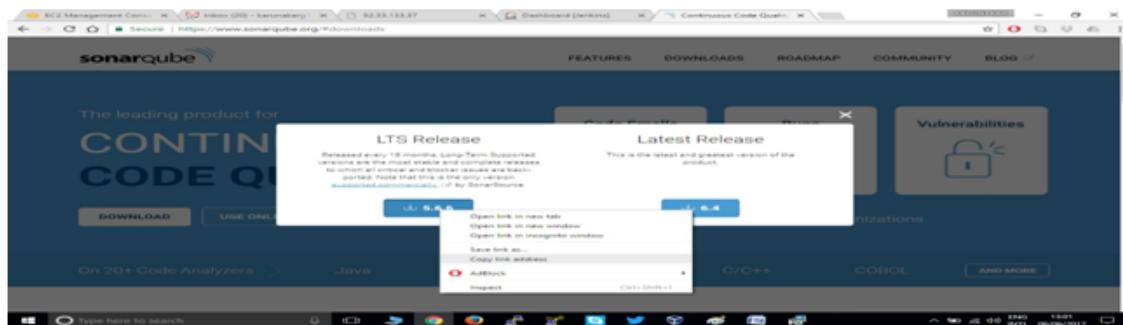
Naren Technologies

Jenkins

```
ln -s /usr/lib/jvm/java-8-oracle/ /usr/lib/jvm/default-java
```

SonarQube Installation

```
sudo apt-get install unzip  
wget https://sonarsource.bintray.com/Distribution.sonarqube/sonarqube-  
5.6.6.zip
```



Here we are installing Unzip

Unzipping the SonarQube Zip file

```
unzip sonarqube-5.6.6.zip cd sonarqube-5.6.6/
```

Mysql Installation Procedure In Sonarqube Server And Configuration

Install MySQL Server.

```
$ sudo apt-get install mysql-server
```

Configure the mysql bind address with sonarqube ip address Start the mysql server

Naren Technologies

Jenkins

```
$ mysql -u user -p  
Mysql>Create the MySQL database.  
Mysql>CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8_general_ci;  
Mysql>CREATE USER 'sonar' IDENTIFIED BY 'sonar';  
Mysql>GRANT ALL ON sonar.* TO 'sonar'@'%' IDENTIFIED BY 'sonar';  
Mysql>GRANT ALL ON sonar.* TO 'sonar'@'localhost' IDENTIFIED BY 'sonar';  
FLUSH PRIVILEGES;
```

Edit the Sonar Configuration in sonarserver.

```
$ sudo vi /sonarqube5.6/conf/sonar.properties  
# User credentials.  
# Permissions to create tables, indices and triggers must be  
granted to JDBC user.  
# The schema must be created first.  
sonar.jdbc.username=sonar  
sonar.jdbc.password=sonar  
#----- MySQL 5.x  
sonar.jdbc.url=jdbc:mysql://sonarip:3306/sonar?  
useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&  
useConfigs=maxPerformance
```

Naren Technologies

Jenkins

Starting The Sonarqube Server

```
bin/linux-x86-64/sonar.sh start
```

stop the firewalls

```
service ufw stop
```

SonarQube DashBoard Mainpage:

The screenshot shows the SonarQube dashboard interface. At the top, there's a navigation bar with links like 'Dashboard', 'Issues', 'Measures', 'Rules', 'Quality Profiles', 'Quality Gates', and 'More'. Below the navigation is a search bar and a user login area. The main content area is titled 'Home' and features a 'Welcome to SonarQube Dashboard' message. It asks if you want to run analysis on a project, start customizing the dashboard, or browse documentation. It also mentions a support page. To the right, there are two sections: 'PROJECTS' and 'PROJECTS'. Both sections have a table with columns: ID, NAME, VERSION, LOC, BUGS, VULNERABILITIES, CODE SMELLS, and LAST ANALYSIS. Under 'PROJECTS', it says 'No data'. Under 'PROJECTS', it also says 'No data'. A note at the bottom states: 'Embedded database should be used for evaluation purpose only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.' At the very bottom, there's a footer with links to 'SonarQube™ technology is powered by SonarSource SA', 'Version 5.6.8 - LGPL v3 - Community - Documentation - Get Support - Plugins - Web API', and system status information like 'CPU: 90% / 40% INTEL 06/06/2017 13:12'.

Jenkins Job Setup

Here we are Installing Sonarscanner Plugin in Jenkins

Naren Technologies

Jenkins

The screenshot shows the Jenkins Plugin Manager interface. The 'Installed' tab is selected, displaying a list of installed Jenkins plugins. One plugin, 'SonarQube', is highlighted. The details for the SonarQube plugin are shown below:

Name	Version	Previously Installed version	Uninstall
SonarQube	2.6.3	2.6.3	[Uninstall]

Manage jenkins ->System Configuration -> assign the sonarqube credentials

The screenshots show the 'Configure System' page under 'SonarQube' settings. Both screenshots show the same configuration for two SonarQube servers:

Server	Name	Server URL	Server version	Server authentication token	SonarQube account login	SonarQube account password
sonarqube	sonarqube	http://172.31.31.230:9000	6.2	[disabled]	admin	[disabled]
sonarqube	sonarqube	http://172.31.31.230:9000	6.2	[disabled]	karunakarg13@gmail.com	[disabled]

Global Tool configuration->JDK name =jdk8->
SonarQubeScanner-> from Maven assign sonar scanner version

Naren Technologies

Jenkins

The screenshot shows the Jenkins Global Tool Configuration page for JDK installations. It displays two entries: 'jdk8' and 'jre11'. Both entries have the 'Name' field set to their respective names and the 'Install automatically' checkbox checked. Under each entry, there is a 'Delete Installer' button. At the bottom, there are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins Global Tool Configuration page for SonarQube Scanner installations. It displays one entry named 'SonarQube'. The 'Name' field is set to 'SonarQube' and the 'Install automatically' checkbox is checked. Below it, there is a 'Delete Installer' button. At the bottom, there are 'Save' and 'Apply' buttons.

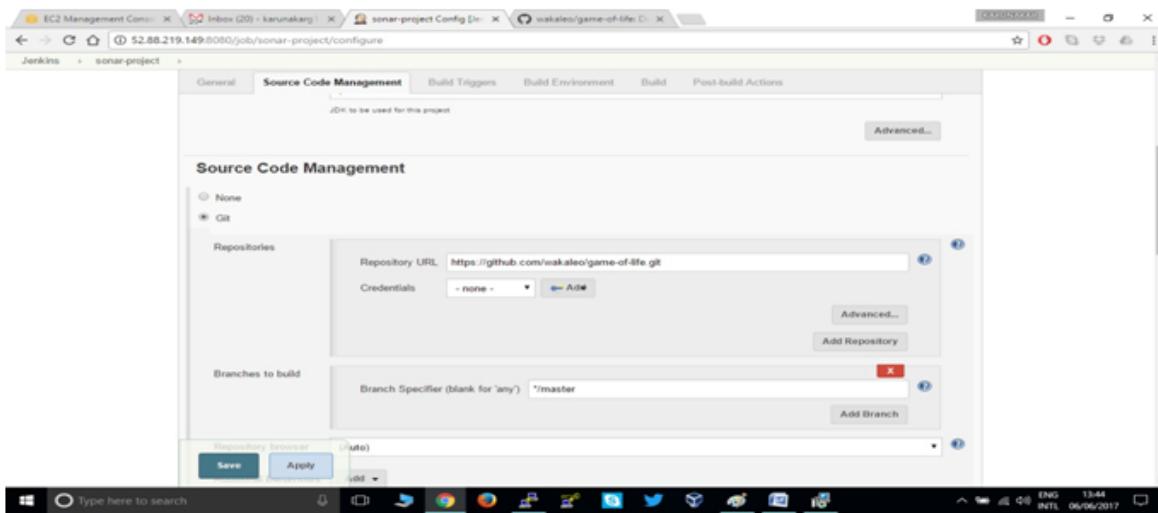
Creating New Project with name of Sonar-project

The screenshot shows the Jenkins 'New Item' creation screen. The 'Item name' field is filled with 'sonar-project'. Below the field, there are several project type options: 'Freestyle project', 'Maven project', 'Pipeline', 'External job', and 'Multi-configuration project'. Each option has a brief description and a 'Configure' link. At the bottom, there is a 'OK' button.

Assigning Git repository in Source code management

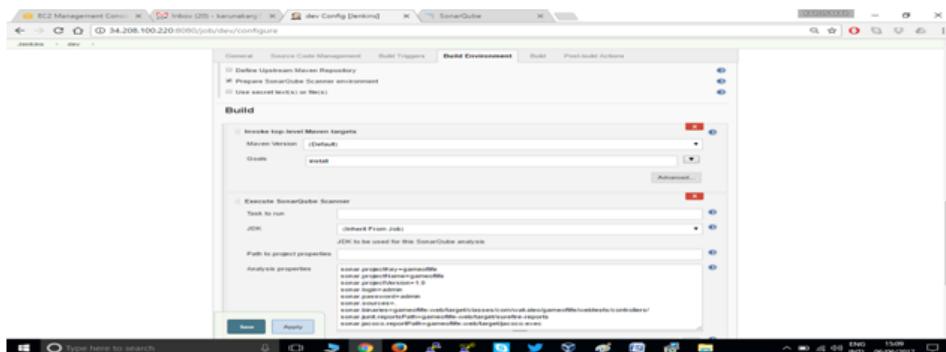
Naren Technologies

Jenkins



=>In Build section Top level maven project
=>And Execute sonarQube scanner
=>Credentials for SonarQube DashBoard and Sonar properties in Sonarscanner Job
sonar.projectKey=gameoflife
sonar.projectName=gameoflife
sonar.projectVersion=1.0
sonar.login=admin
sonar.password=admin
sonar.sources=.
sonar.binaries=gameoflife-
web/target/classes/com/wakaleo/gameoflife/webtests/controllers/
sonar.junit.reportsPath=gameoflife-web/target/surefire-reports
sonar.jacoco.reportPath=gameoflife-web/target/jacoco.exec

These snapshot is showing that sonarproperties and credentials of the sonarqube



Naren Technologies

Jenkins

- =>After entering the sonar properties save the project and go to Build now,
- =>Below snapshots showing that build process as well as sonar scanner tests
- =>After Success The job that sonartest files and reports are stored in Var/lib/jenkins /workspace directory so these files and reports we have to give in sonarDashboard to see the output.

The screenshots show the Jenkins console output for four different builds (dev #28, dev #29, dev #30, and dev #31) of the 'gamenoflife-web' project. Each screenshot displays the build log, which includes the SonarQube scanner configuration, repository details, and the execution of the SonarQube scanner. The logs indicate a successful build with various metrics like total time, final memory usage, and the number of source files analyzed. The logs also mention Java bytecode initialization and the absence of certain Java libraries.

```

[INFO] [INFO] gameoflife-web ..... SUCCESS [ 8.004 s]
[INFO] [INFO] BUILD SUCCESS
[INFO] [INFO] -----
[INFO] [INFO] Total time: 20:006 s
[INFO] [INFO] Finished at: 2017-06-06T06:10:08+00:00
[INFO] [INFO] Final Memory: 28M/68M
[INFO] [INFO] -----
[INFO] [INFO] SonarScanner 3.0.3.1778
[INFO] [INFO] Java 1.8.0_131 Oracle Corporation (64-bit)
[INFO] [INFO] User cache: /var/lib/jenkins/.sonar/cache
[INFO] [INFO] Load global repositories
[INFO] [INFO] Load global repositories (done) | Time=30ms
[INFO] [INFO] Load local repository (done) | Time=0ms
[INFO] [INFO] Load plugin index
[INFO] [INFO] Load plugin index (done) | Time=6ms
[INFO] [INFO] SonarQube server 5.6.6
[INFO] [INFO] SonarQube scanner 3.0.6
[INFO] [INFO] Process project properties
[INFO] [INFO] Load project repositories
[INFO] [INFO] Load project repositories (done) | Time=264ms
[INFO] [INFO] Load quality profiles (done) | Time=11ms
[INFO] [INFO] Load active rules
[INFO] [INFO] 
[INFO] [INFO] gameoflife-web ..... SUCCESS [ 8.004 s]
[INFO] [INFO] BUILD SUCCESS
[INFO] [INFO] -----
[INFO] [INFO] Total time: 20:006 s
[INFO] [INFO] Finished at: 2017-06-06T06:10:08+00:00
[INFO] [INFO] Final Memory: 28M/68M
[INFO] [INFO] -----
[INFO] [INFO] 100 files indexed
[INFO] [INFO] Quality profile for java: Sonar way
[INFO] [INFO] Quality profile for jax: Sonar way
[INFO] [INFO] JaCoCo IT report not found: /var/lib/jenkins/workspace/dev/target/jacoco-it.xml
[INFO] [INFO] Sensor JavaSquidSensor
[INFO] [INFO] Configured Java source version (sonar.java.source): none
[INFO] [INFO] JavaClasspath initialization...
[INFO] [INFO] Bytecode of dependencies was not provided for analysis of source files, you might end up with less precise results. Bytecode can be provided using sonar.java.binaries property
[INFO] [INFO] sonar.binaries and sonar.libraries are deprecated since version 2.5 of sonar-java-plugin, please use sonar.java.binaries and sonar.java.libraries instead
[INFO] [INFO] JavaClasspath initialization done: 32 ms
[INFO] [INFO] JavaTestClasspath initialization...
[INFO] [INFO] Bytecode of dependencies was not provided for analysis of test files, you might end up with less precise results. Bytecode can be provided using sonar.java.test.libraries property
[INFO] [INFO] JavaTestClasspath initialization done: 0 ms
[INFO] [INFO] Java Main Files AST scan...
[INFO] [INFO] 30 source files to be analyzed
[INFO] [INFO] 30/30 source files have been analyzed
[INFO] [INFO] Java Main Files AST scan done: 4229 ms
[INFO] [INFO] Java bytecode scan...
[INFO] [INFO] Class 'com/wakaleo/gamenoflife/domain/Cell' is not accessible through the ClassLoader.
[INFO] [INFO] Class 'com/wakaleo/gamenoflife/domain/Grid' is not accessible through the ClassLoader.
[INFO] [INFO] Class 'com/wakaleo/gamenoflife/domain/GridHeader' is not accessible through the ClassLoader.
[INFO] [INFO] Class 'com/wakaleo/gamenoflife/domain/GridHeader$1' is not accessible through the ClassLoader.
[INFO] [INFO] 
[INFO] [INFO] gameoflife-web ..... SUCCESS [ 8.004 s]
[INFO] [INFO] BUILD SUCCESS
[INFO] [INFO] -----
[INFO] [INFO] Total time: 20:006 s
[INFO] [INFO] Finished at: 2017-06-06T06:10:08+00:00
[INFO] [INFO] Final Memory: 28M/68M
[INFO] [INFO] -----

```

Naren Technologies

Jenkins

The image displays three separate windows of a Jenkins build console, each showing the output of a build process. The logs are as follows:

Top Window:

```
INFO: [INFO] -----  
[INFO] gameoflife-web ..... SUCCESS [ 8.504 s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 23.686 s  
[INFO] Finished at: 2017-06-06T08:10:00+00:00  
[INFO] Final Memory: 28M/68M
```

Middle Window:

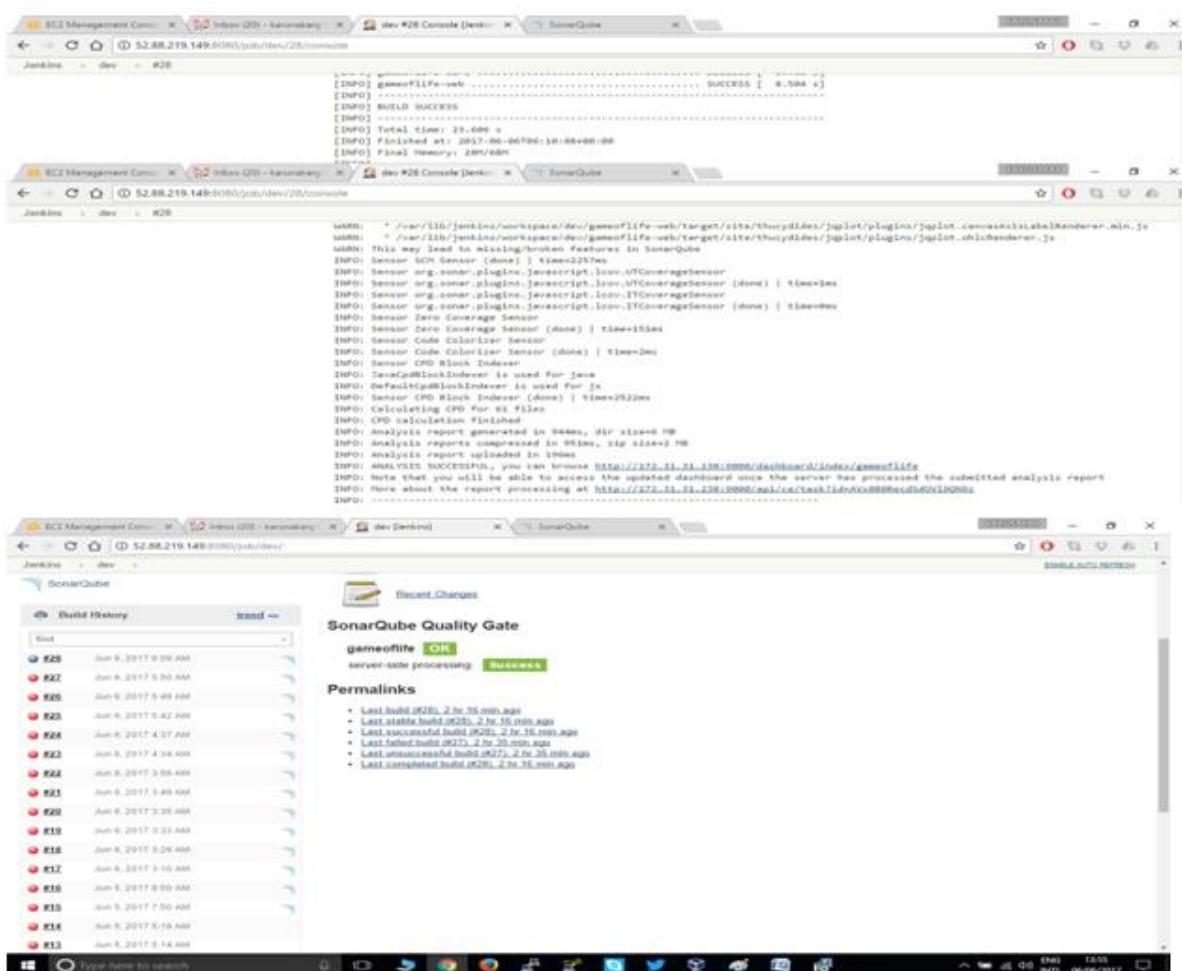
```
INFO: 0/78 files analyzed  
WARNING: Missing blame information for the following files:  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.clParser.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.cursor.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.dragable.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.json2.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.prototype.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.canvasPixelTickRenderer.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/datatables/media/js/jquery.dataTables.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplotBubbleRenderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.cipher.ser.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.cookie.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.logarithmicRenderer.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.mekkoAreaRenderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.highlighter.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.dataTableRenderer.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.pieRenderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.needleRenderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.pyramidBubbleRenderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/plugins/jqplot.nivo_slider.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/escanove.wm.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/funnel2Renderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/jellyfishCurveRenderer.min.js  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/htmlUnit/htmlUnitOverlay.js
```

Bottom Window:

```
INFO: 0/78 files analyzed  
WARNING: Missing blame information for the following files:  
INFO:   * /var/lib/jenkins/workspace/dev/gameoflife-web/target/site/thucydides/jqplot/htmlUnitOverlay.js
```

Naren Technologies

Jenkins



After a SonarQube Scanner has finished analysing your code, the result of the analysis (Sources, Issues, Metrics) - the Analysis Report - is sent to SonarQube Server for final processing by the Compute Engine.

In this SonarDashborad contains

=>code coverage

=>Bugs

=>code smells

=> vulnerabilities

=>unit tests

=> Duplications

Naren Technologies

Jenkins

The screenshot shows the SonarQube dashboard for the 'gameoflife' project. The top navigation bar includes links for Dashboards, Issues, Measures, Roles, Quality Profiles, Quality Gates, Administrations, and More. A search bar and a 'Configure widgets' button are also present. The main content area is divided into several sections:

- Welcome to SonarQube Dashboard:** A message indicating successful server startup and a call to action to start analyzing projects.
- PROJECTS:** A table showing project details: gameoflife (Version 1.0, 41,306 LOC, 2,734 bugs, 21 vulnerabilities, 2,722 code smells, last analysis 06/10).
- PROJECTS:** A section showing SonarQube's Code Coverage, with a large green bar labeled 'gameoflife'.
- MY FAVOURITES:** A section showing no data.
- Issues:** A detailed list of findings:
 - Remove this unused import 'com.badlogic.gdx.graphics.g2d.SpriteBatch requirements GameOrTheApplication': 0 hours ago
 - Code Smell: Minor • Open • Not assigned • 2min effort • Comment
 - Remove this unused import 'net.mycodes.com.annotations.FadeIn': 0 hours ago
 - Code Smell: Minor • Open • Not assigned • 2min effort • Comment
 - Remove this unused import 'net.mycodes.com.annotations.Sterby': 0 hours ago
 - Code Smell: Minor • Open • Not assigned • 2min effort • Comment
 - Make this array "private": 0 hours ago
 - Vulnerability: Critical • Open • Not assigned • 1min effort • Comment
 - Reorder the modifiers to comply with the Java Language Specification: 0 hours ago
 - Code Smell: Minor • Open • Not assigned • 2min effort • Comment
 - 2 duplicated blocks of code must be reviewed: 0 hours ago
 - Code Smell: Major • Open • Not assigned • Multilevel effort • Comment
 - Remove this unused import 'com.badlogic.gdx.scenes.scene2d.actions.SequenceAction requirements GameOrTheApplication': 0 hours ago
 - Code Smell: Minor • Open • Not assigned • 2min effort • Comment

Naren Technologies

Jenkins

The screenshot displays two windows of the SonarQube interface. The top window shows the 'gameoflife' project's dashboard with metrics for Reliability, Security, Maintainability, and Coverage. The bottom window shows a detailed code analysis report for the same project.

Reliability Metrics:

- Bugs: 2,734 (0 New Bugs)
- Reliability Rating: F
- Reliability Remediation Effort: 51h
- Reliability Remediation Effort on New Code: 8h

Security Metrics:

- Vulnerabilities: 21 (0 New Vulnerabilities)
- Security Rating: D
- Security Remediation Effort: 3h 40m
- Security Remediation Effort on New Code: 8h

Maintainability Metrics:

- Code Smells: 2,722 (0 New Code Smells)
- Maintainability Rating: A
- Technical Debt: 57h
- Average Technical Debt: 9h
- Technical Debt Ratio: 2.2%
- Technical Debt Ratio on New Code: 0.0%
- Effort to Reach Maintainability Rating A: 0h

Coverage Metrics:

- Coverage: 90.7%
- Line Coverage: 92.8%
- Condition Coverage: 83.2%
- Uncovered Lines: 0
- Uncovered Conditions: 2

Code Analysis Report (Bottom Window):

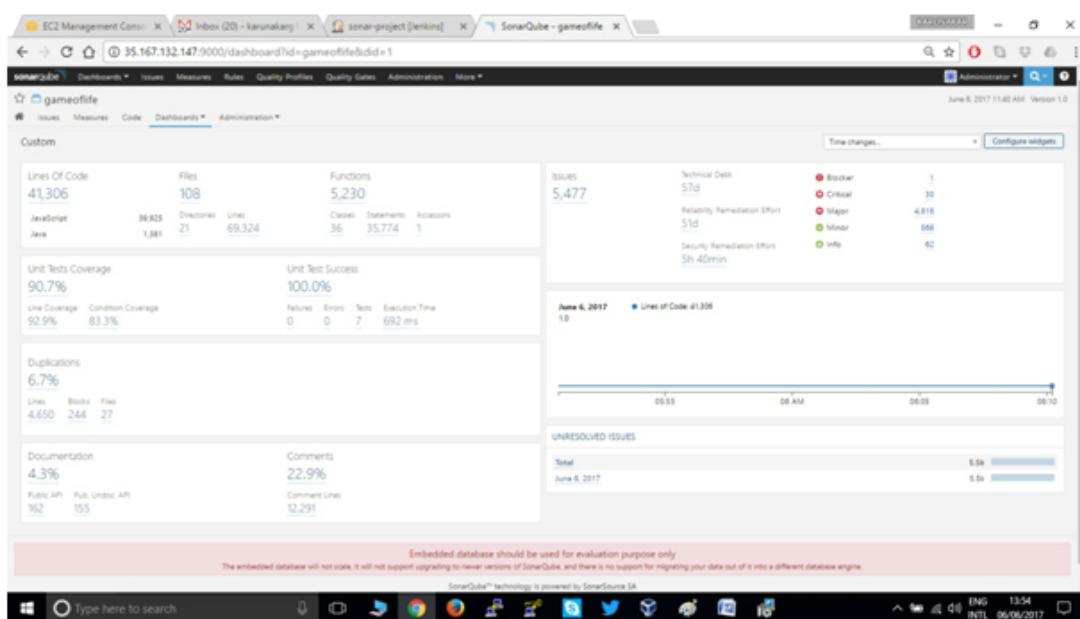
LOCs OF CODE	BUGS	VULNERABILITIES	CODE SMELLS	COVERAGE	DUPLICATES
474	0	2	17	80.7%	8.7%
100	0	0	1	0.0%	0.0%
18	0	0	4	0.0%	0.0%
14	0	0	4	0.0%	0.0%
237	0	0	7	0.0%	0.0%
628	0	0	9	0.0%	0.0%
16	0	0	4	0.0%	0.0%
4	0	0	0	0.0%	0.0%
69	1	0	2	60.7%	0.0%
78	0	0	4	0.0%	0.0%
5.40	42	0	148	0.0%	0.0%
4.89	236	0	400	0.0%	0.0%
8.24	703	0	489	7.2%	0.0%
1.74	31	0	41	25.8%	0.0%
1.18	91	4	110	14.0%	0.0%
2.46	10	0	45	3.9%	0.0%
0	0	0	0	0.0%	0.0%
8.07	198	0	99	0.0%	0.0%
8.46	238	0	470	4.3%	0.0%
8.46	233	4	374	0.0%	0.0%

Estimated database should be used for evaluation purpose only.
The embedded database will not scale. It will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube Dashboard

Naren Technologies

Jenkins



Pass/Fail Notification:

Once an analysis is done, a report is sent to the SonarQube server to be integrated. At the end of this integration, a standard web hook mechanism lets you notify any external system to do whatever you want: trigger an alarm, update a wallboard, and notify a chat room.

Naren Technologies

Jenkins

Artifactory

Documented By Pavan Kumar Ranjit.

As the first, and only, universal Artefact Repository Manager on the market, JFrog Artifactory fully supports software packages created by any language or technology.

Artifactory is the only enterprise-ready repository manager available today, supporting secure, clustered, High Availability Docker registries.

Integrating with all major CI/CD and DevOps tools, Artifactory provides an end-to-end, automated and bullet-proof solution for tracking artefacts from development to production.

What Is A Binary Repository Manager?

It's a single gateway through which you access external artefacts, and store your own build artefacts. By centralizing the management of all binary artefacts, it overcomes the complexity arising from the diversity of binary types, their position in the workflow and the dependencies between them.

Top 10 Reasons You Should Use A Binary Repository Manager

Naren Technologies

Jenkins

Top 10 Reasons You Should Use A Binary Repository Manager

1. Increase Build Speed and Proxy Remote Artefacts. Eliminate redundant downloads by automatically caching artefacts for the entire development team.
2. Manage Artefact Lifecycle. Promote artefacts from development through QA to production and distribution.
3. Avoid License Violations. Receive alerts on potential violations. Control all licenses used in your software by managing third party artefacts in one central location.
4. Keep Builds Reproducible. Integrate with leading CI servers and keep builds reproducible with exhaustive build information to track and protect all artefacts used by your CI builds.
5. Enforce Module Security. Control access and track all changes made to artefacts. Block unwanted external requests, and define who can create, delete or overwrite artefacts.
6. Control Module Consumption. Define access rules for users and groups that consume artefacts.

Naren Technologies

Jenkins

7. Share Artefacts Across Teams. Easily distribute artefacts produced by different teams in your organization, and share them as dependencies between the teams as needed.
8. Clean Up CI Artefacts Clutter. Automatically clean up integration and snapshot artefacts produced during the CI build process.
9. Locate Artefacts Instantly. Easily find artefacts with advanced search capabilities, including class search, and run bulk operations on the result set.
10. Automate While You “REST”. Automate and integrate all aspects of artefact management within your organization using a powerful REST API.

The Most Powerful Repository Around Download Blocking

We all know that prevention is better than cure. The same is true of your software systems. Removing potentially harmful dependencies once they are already deeply entrenched in your production software can require finding an alternative component, refactoring your software, and running several cycles of QA before you’re finally back where you were. Through Artifactory’s integration with Xray, you can

Naren Technologies

Jenkins

avoid this kind of scenario by preventing artefacts that X-ray has detected to have issues or vulnerabilities from being downloaded from your repositories and used in the first place.

Chef

The concept of “Infrastructure as Code” has been widely adopted by most enterprise IT organizations. Chef provides IT and DevOps with the tools they need to manage the different environments they need to spin up. Through support for Chef Cookbook repositories, Artifactory brings a new dimension to Infrastructure as Code. By managing configuration packages through a binary repository, IT and DevOps organizations working hard on configuration management with Chef now have many more capabilities at their fingertips.

Puppet

The concept of “Infrastructure as Code” has been widely adopted by most enterprise IT organizations. Puppet provides IT and DevOps with the tools they need to manage

Naren Technologies

Jenkins

the different environments they need to spin up. Through support for Puppet repositories, Artifactory brings a new dimension to Infrastructure as Code. By managing configuration packages through a binary repository, IT and DevOps organizations working hard on configuration management with Puppet now have many more capabilities at their fingertips.

Docker

Use Artifactory to manage your in-house Docker images. Distribute and share your images among teams across your organization, whether on-site or at remote locations, just like using Docker Hub Enterprise. Control access to your images using secure “docker pull”, and never have to rely on the internet to access them. Once your images are stored in your repository, find them easily with smart search.

Distribution Repository

Artifactory takes its integration with JFrog Bintray to the next step with Distribution Repositories streaming liquid software from Artifactory to Bintray. Distribution

Naren Technologies

Jenkins

repositories provide an easy way to move artefacts from Artifactory to Bintray, for distribution to end users. As opposed to other repositories in Artifactory, distribution repositories are not typed to a particular package format, but rather, are governed by a set of rules that give fine-grained control over how to specify exactly where an artefact in the distribution repository should be routed to in its corresponding repository in Bintray.

Google Cloud Storage

Upload artefacts indefinitely to Google's secure and highly available storage and let Artifactory manage them for you through full support for Google Cloud Storage (GCS). When used with Artifactory HA your whole system is highly available with no single-point-of-failure. Ready to start using GCS? Just configure an XML file and do a simple migration of your current filestore. GCS is available on Artifactory with an Enterprise license.

Build Integration

Naren Technologies

Jenkins

Jenkins/Hudson, TeamCity And Bamboo

Stream your builds of liquid software into Artifactory from your favorite CI Server together with exhaustive build environment information captured during deployment to enable fully reproducible builds that continuously update computer systems and devices. Promote builds and use the build's Bill of Materials to view deployed modules with their published artefacts and dependencies in all scopes. See where specific artefacts are used and receive warnings when required build dependencies are removed. Link back to the build information in the CI server and vice versa. Currently, Jenkins, Hudson, JetBrains TeamCity and Atlassian Bamboo are supported.

Git LFS

Do you use Git for source code control? So do many others. But what about the binary assets that go along with your source code? Git is not the best solution for that.

“GitHub LFS,” you say? Well, there is a better solution still. Artifactory is a fully-fledged Git LFS (Large File Storage) repository and can optimize your workflow when working with large media files and other binary resources. Artifactory fully supports the Git LFS

Naren Technologies

Jenkins

API, so all you need to do is configure your Git client to point to Artifactory as the Large File Storage repository.

Artifactory Vs. Nexus The Integration Matrix

The evolution of Continuous Integration, build tools and build servers in the past years has been very impressive. The amount of projects (open source, or not) and tools that were launched and later adopted by the community (Maven, Gradle, Jenkins/Hudson, TeamCity, Bamboo, and more...) shows that this is the future of software development and this is how people will build and package their applications. One of the greatest advantages, especially in the Java world, is that development teams have the freedom to choose and build modular environments by integrating the tools that they like, need and that were adopted by their organization. This is why our users build their projects using Maven, TeamCity, Ivy, Hudson, Gradle, Bamboo and recently Jenkins. Bottom line is, you should not be concerned about the ability of the tools to integrate while designing your CI stack! BUT, things are changing. Now even more with the fork of Hudson CI (currently owned by Oracle),

Naren Technologies

Jenkins

Artifactory has come to be the only Binary Repository Manager that gives you the real freedom to choose. During the past year, JFrog's team has been developing for the Artifactory users a number of open source plug-ins that are tightly integrated with the world's leading tools and vendors. The following comparison tables are more than just a list of features – it is what we envision when we think about the software development tooling, and how we ensure that our users keep their freedom of choice:

Build tools integration

Build Servers Integration

4. Why Should I Use Jcenter Over Maven Central For Downloading Dependencies?

jcenter is the public repository hosted at bintray that is free to use for open source library publishers. There are load of good reasons to use jcenter over Maven Central. Here are some of the major ones.

1. jcenter delivers library through CDN which means improvements in CI and developer builds.
2. Avoid trojan codes
3. jcenter is the largest Java Repository on earth, so whatever is available on Maven Central is available on jcenter as well.
4. It is incredibly easy to upload your own library to bintray. No need to sign them or do any complex things like you have to on Maven Central.
5. Friendly-UI
6. If you want to upload your library to Maven Central you could do it easily with a single click on the bintray site.