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.

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

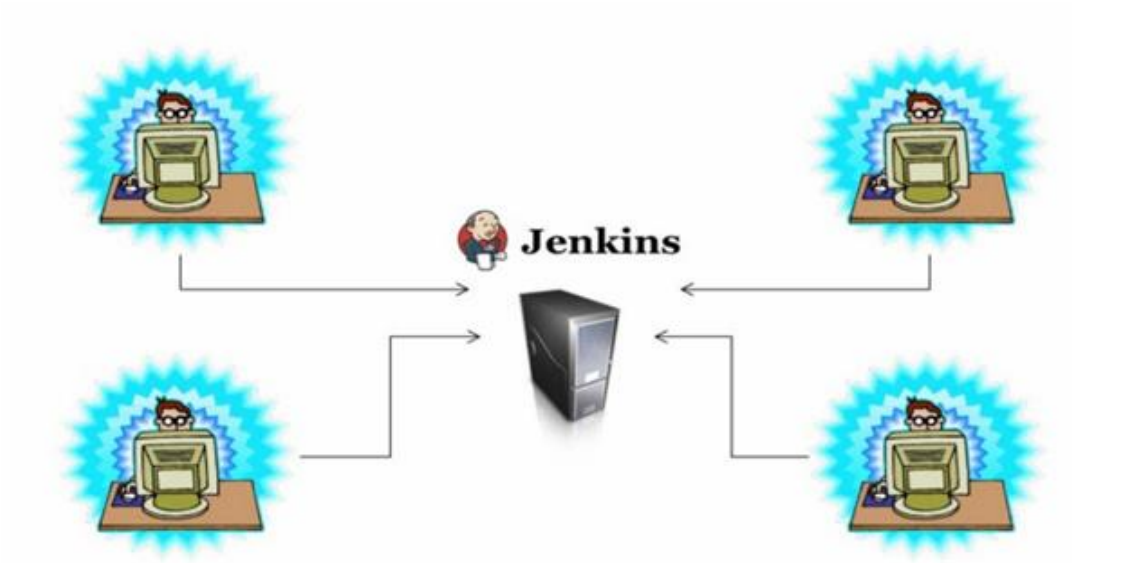
Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control. CI encourages developers to share their code and unit tests by merging their changes into a shared version control repository after every small task completion. 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.

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 their is any issue in the build process the developers will get notified through email and they will fix the problem.

Jenkins

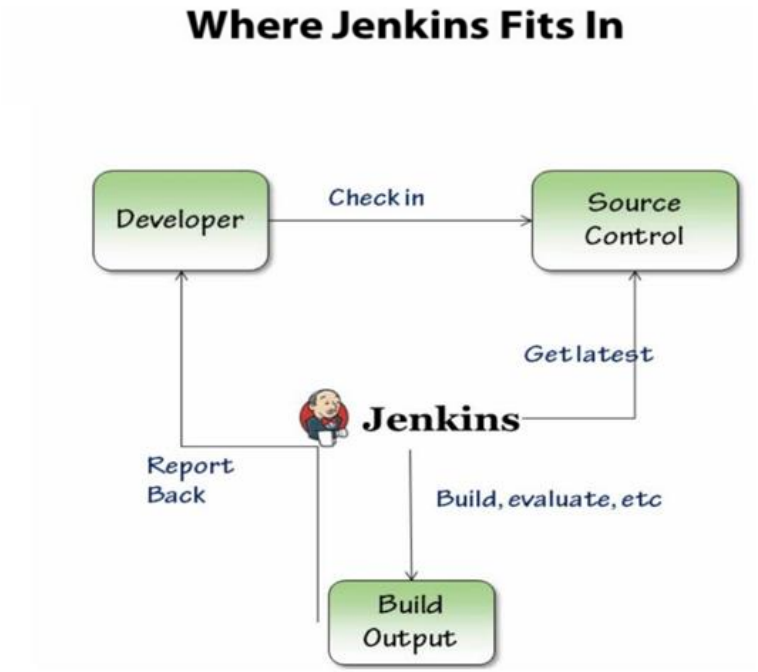
What is Jenkins

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

OpenSorce

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.



Extensible

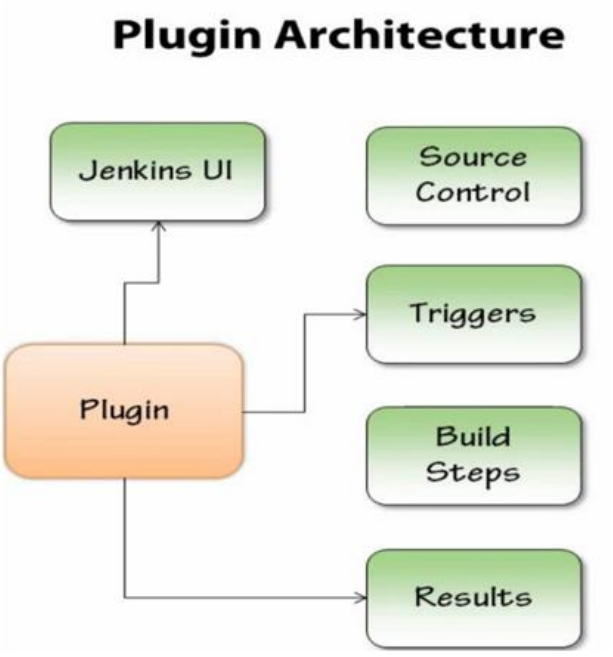
Jenkins comes with lot of goodies but its just not limited by that, Jenkins main power is its extensibility that can be achieved by installing plugins into it.

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".



Jenkins Setup

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

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 requirment to run Jenkins if you are not using git and maven.

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

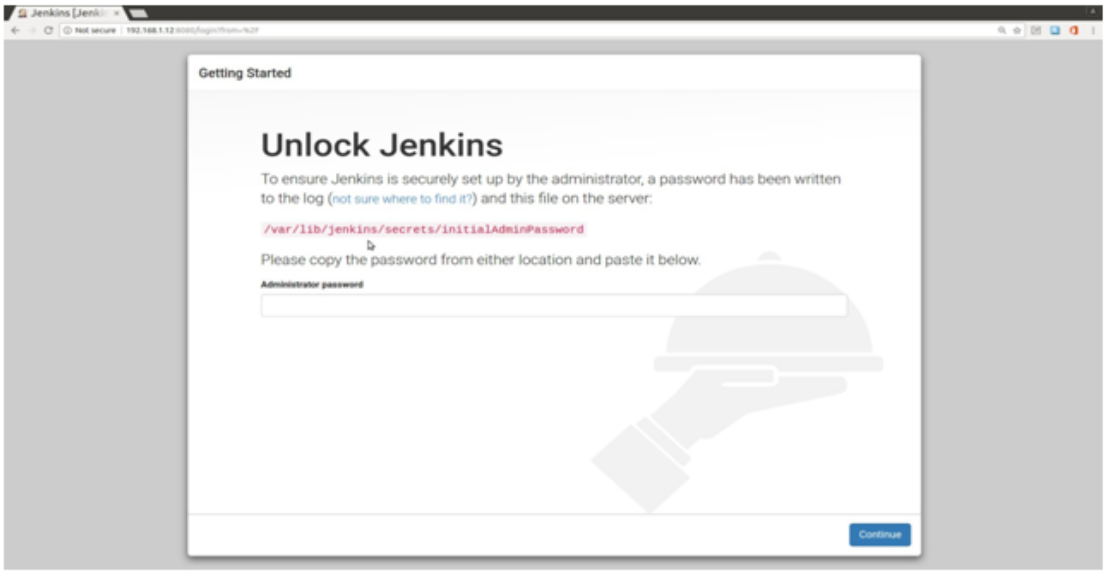
Accessing Jenkins

Jenkins runs on port 8080 by default

Open up a browser and use below url.

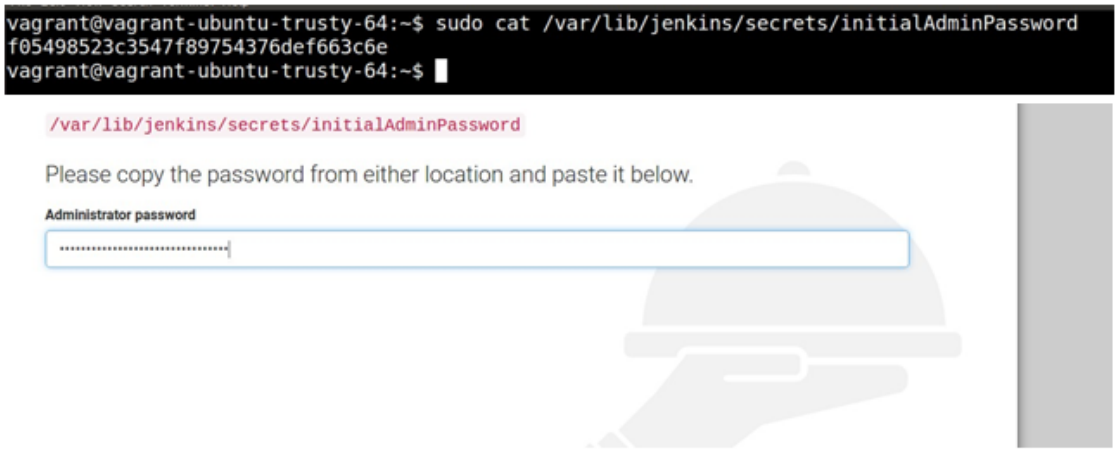
http://jenkinsIP:8080

Jenkins will set a random password to unlock jenkins setup.

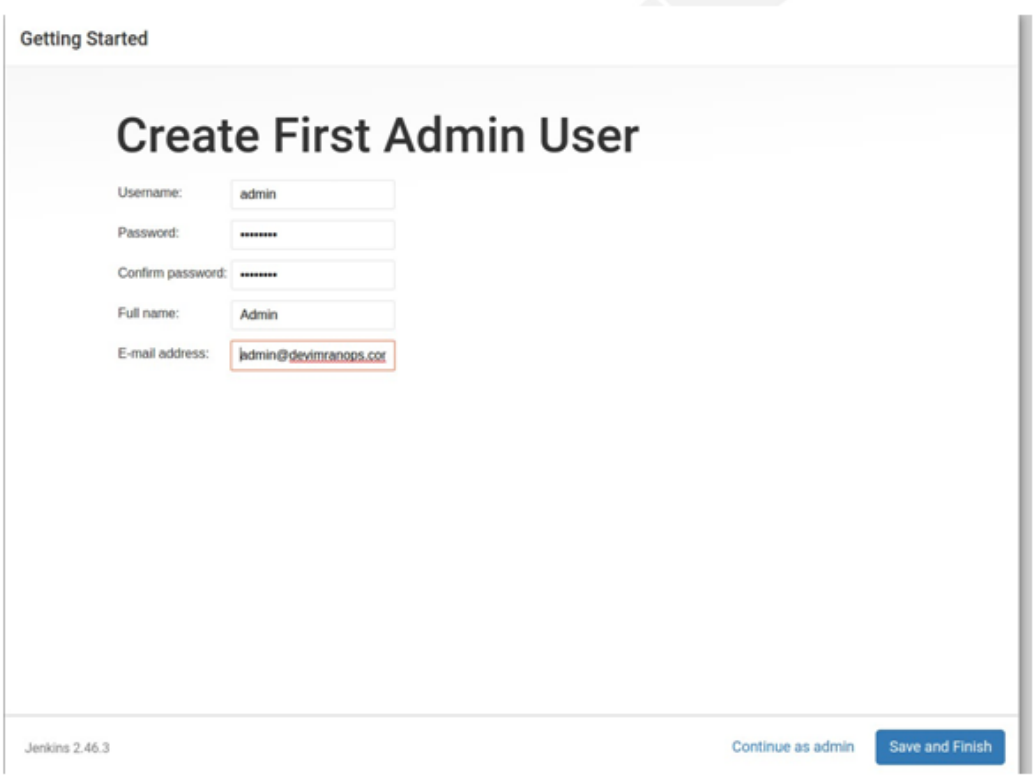


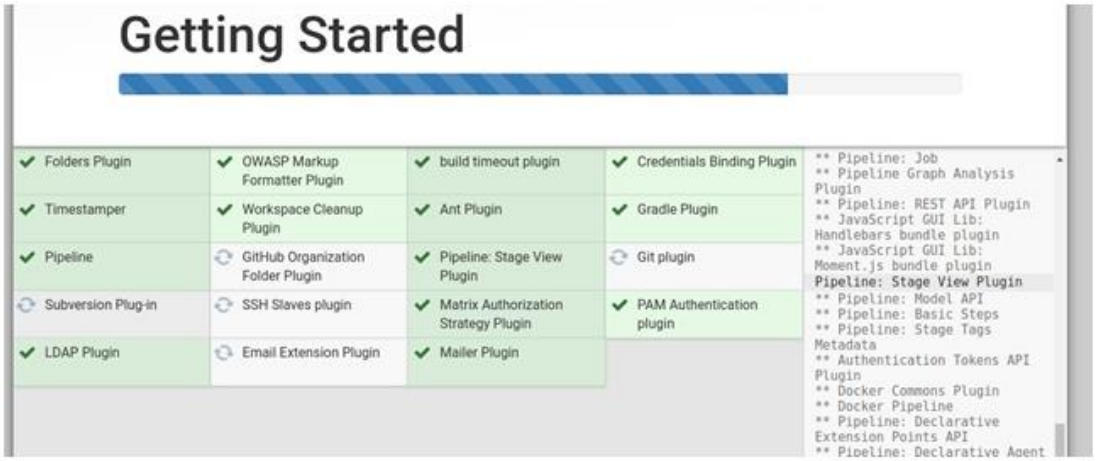
The password would be stored in /var/lib/jenkins/secrets/initialAdminPassword file.

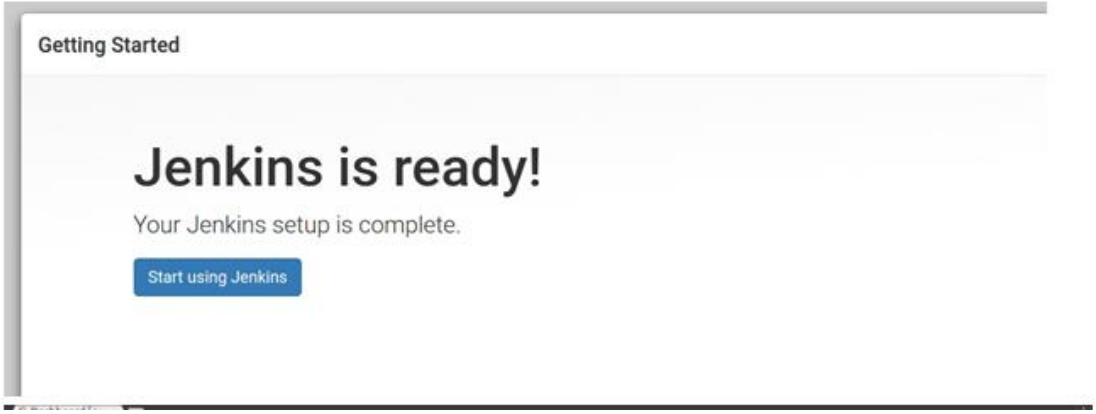
Read that file and get the password. Use that password to unlock jenkins.



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.

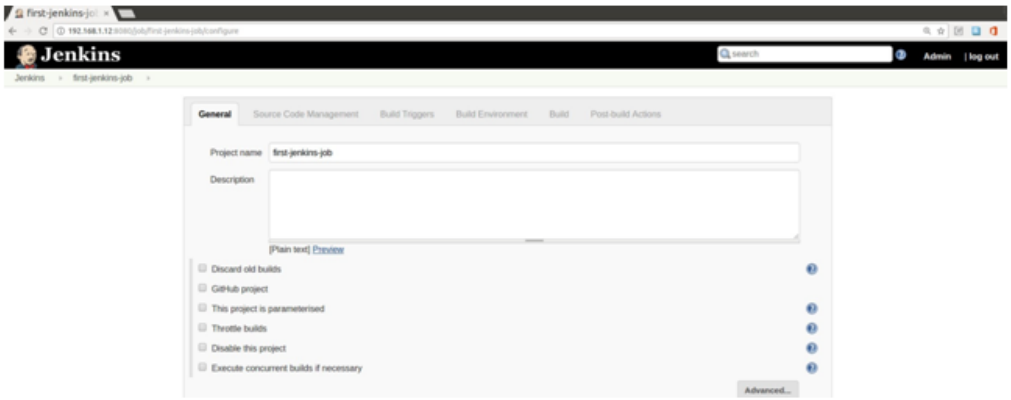

Getting Started
Create First Admin User
Username:
admin
Password:
Confirm password:
Full name:
Admin
E-mail address:
admin@devimranops.cor
Jenkins 2.46.3
Continue as admin
Save and Finish


Getting Started
Folders Plugin
OWASP Markup Formatter Plugin
build timeout plugin
Credentials Binding Plugin
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
** 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


Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins

Jenkins

Main Dash Board Of Jenkins.



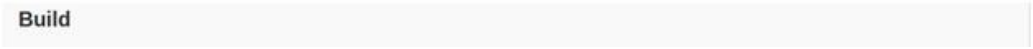
Creating first jenkins job.

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

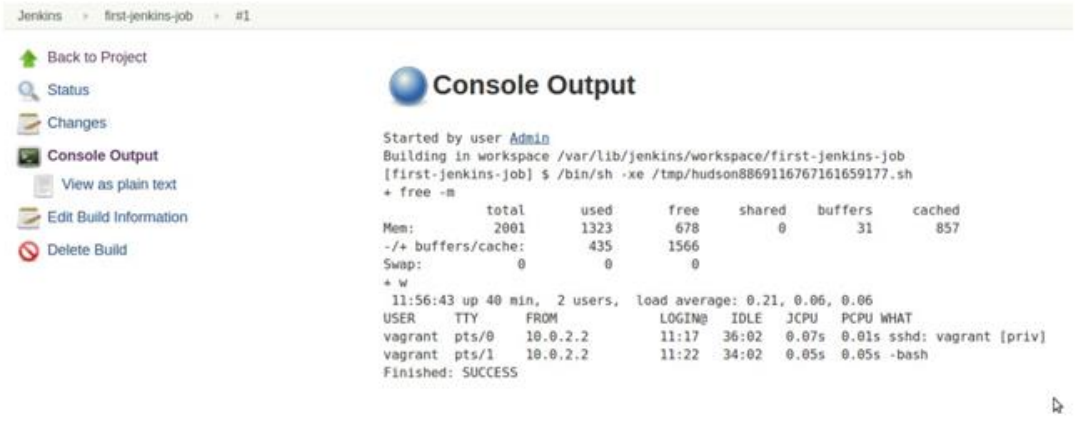
Go to Build section => From drop down select shell => Enter some Linux commands “free -m” ; w => Save.



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.



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

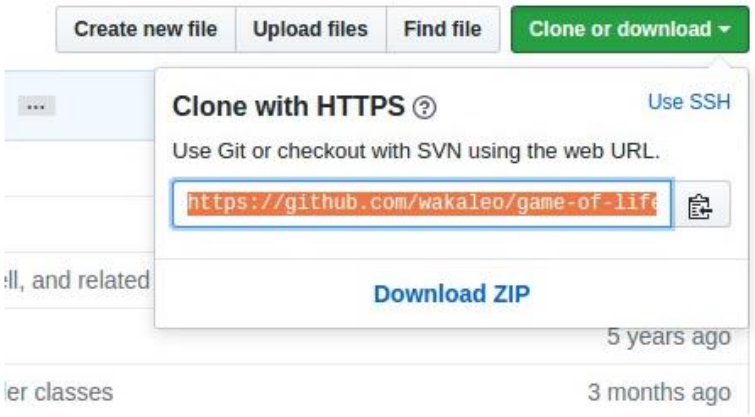
Source Code Managment

Copy the URL of gameoflife source code from github.

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

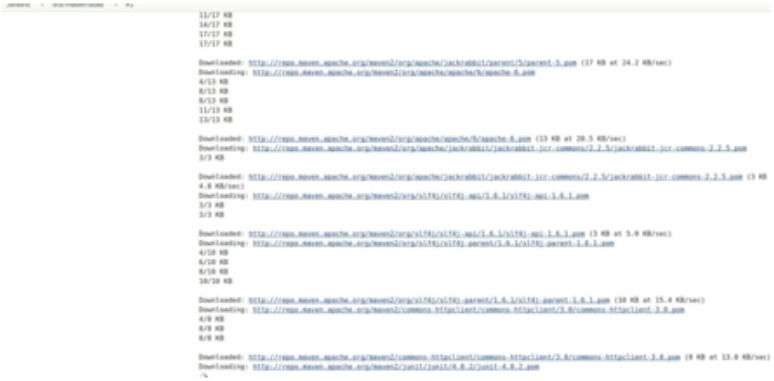
Enter the game of life github url.

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



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

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



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 artefact
4. Archive artefact
5. Run unit test cases in the source code.

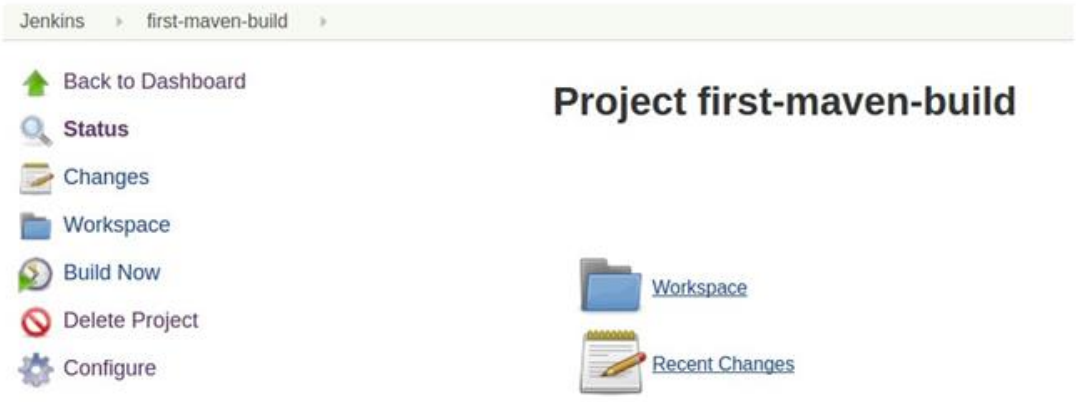
```
[INFO] Reactor Summary:
[INFO]
[INFO] gameoflife ......................................... SUCCESS [2:52.094s]
[INFO] gameoflife-build ................................... SUCCESS [1:04.176s]
[INFO] gameoflife-core .................................... SUCCESS [8.782s]
[INFO] gameoflife-web ..................................... 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.



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.

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


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 m
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.

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.

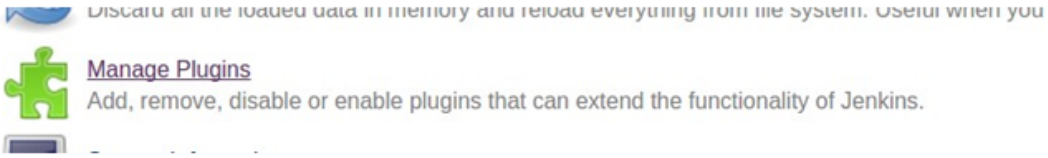
http://<JenkinsIP>:8080/restart

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

Click Manage plugins



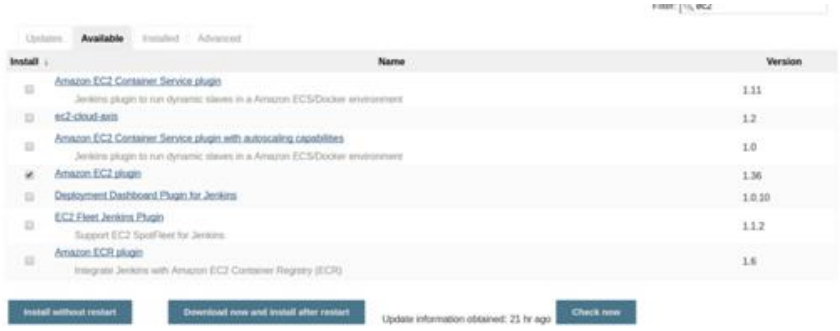
There are four tabs

1. Update

If any plugin is outdated or a never 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.



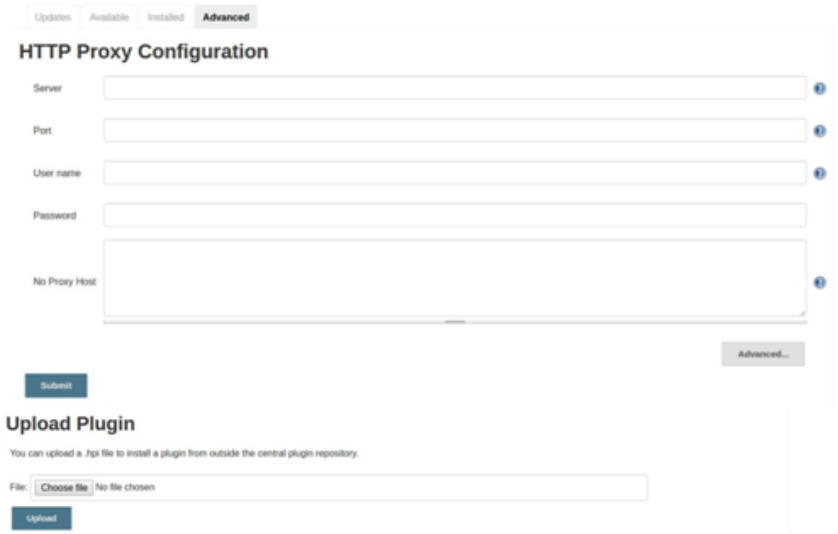
3. Installed

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

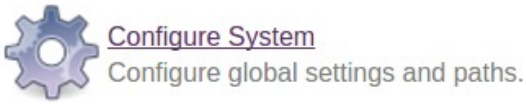
4. Advanced

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.



Configure System



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.

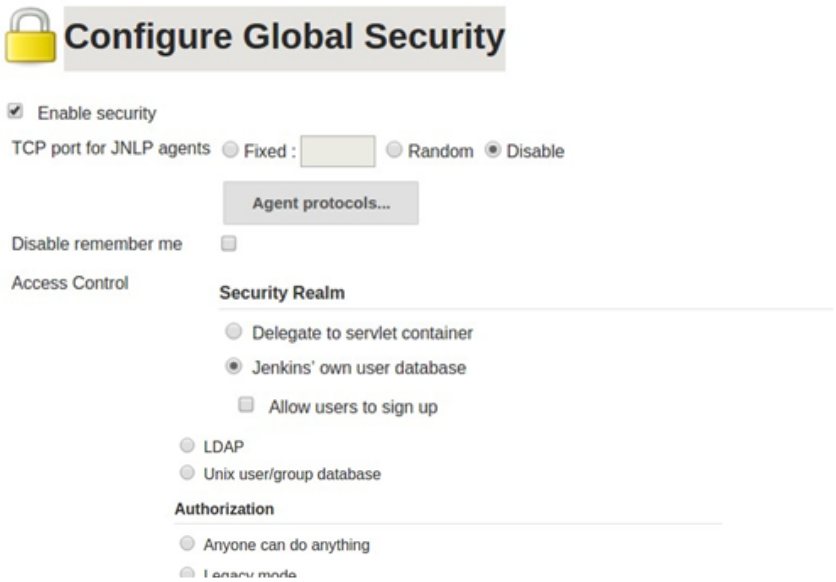
Environment Variables & Tools Path Can Also Be Set.

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.



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

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	*mydomaincontroller.mycompnay*.com:389
Root DN	dc=*mycompnay*,dc=com
User Search Filter	sAMAccountName={0}
Manager DN	cn=*mymanageruser*,ou=users,ou=na,ou=*mycompany*,dc=*mycompany*,dc=com

Manager DN	cn=*mymanageruser*,ou=users,ou=na,ou=*mycompany*,dc=*mycompany*,dc=com
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

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.

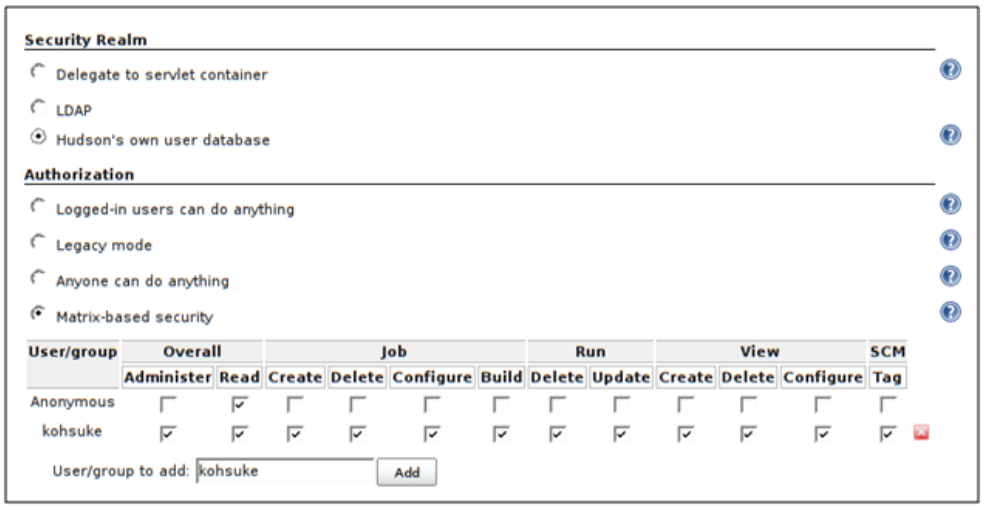
Authorization

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.

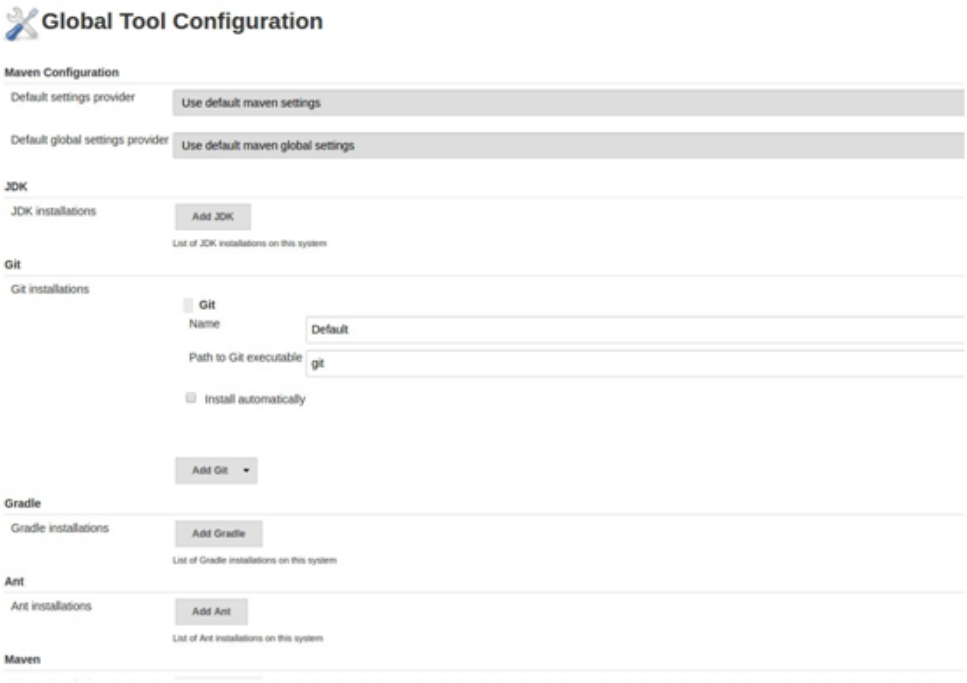
1. Select Matrix-based security as the Authorization
2. Give the Anonymous user only Overall Read access
3. In the text box below the matrix, type your user name (or the user name you plan to use when you register as a new Jenkins user) and click Add
4. Give yourself full access by checking the entire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



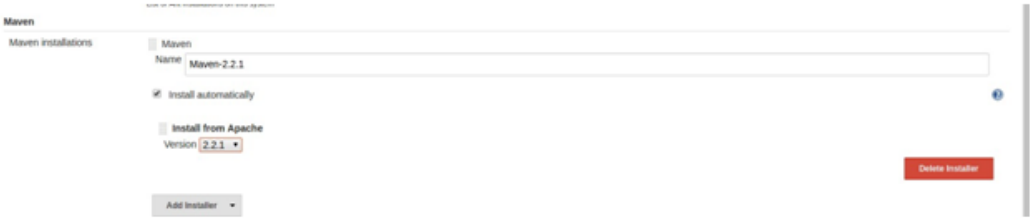
6. Click Save at the bottom of the page. You will be taken back to the top page. Now Jenkins is successfully secured.
7. Restart Jenkins (service Jenkins restart on Linux) If you set up a service like NIS, Active Directory or LDAP, you can now log in to Jenkins using your network credentials. If you 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

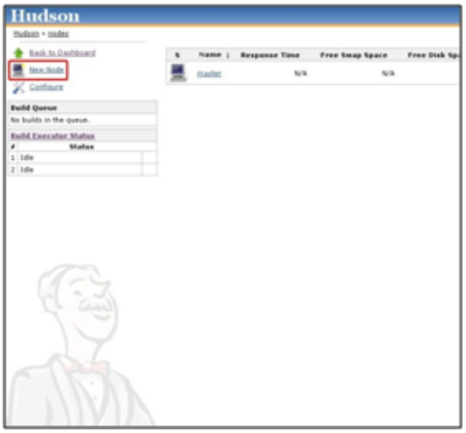
Jenkins gets integrated with variety of tools. In the build job, we have seen it gets integrated with git and maven. Similarly, you can have other tools in your OS installed or you can install it from this page. Also, sometimes we need different versions of Java or Maven or Git etc. We can manage multiple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



Installing Maven 2.2.1

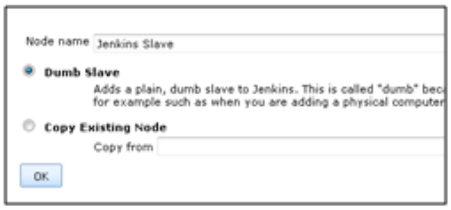


Adding A Windows Slave Node.



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

>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)

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.

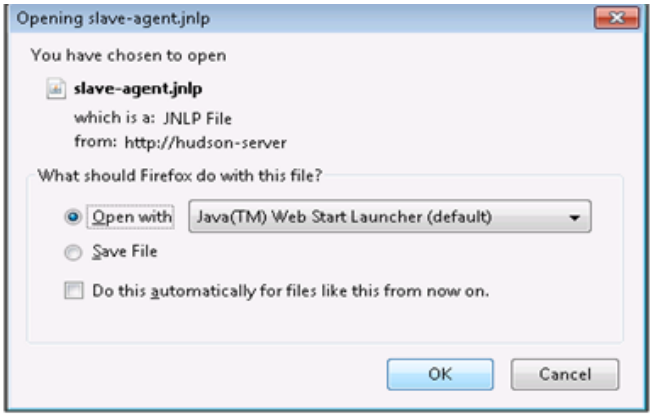
5. Now you need to connect your slave machine to the master using the following steps.

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.

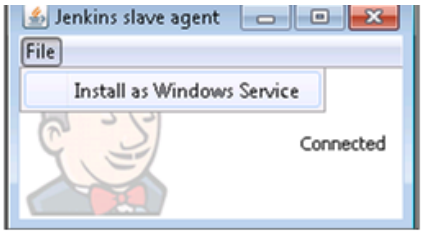


If you encounter connection issue, then you could enlarge the popup windows to see the master port used and check your network configuration (firewall, port forward, ...)

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.



Note that this feature requires ".Net Framework 3.5" .NET Framework 3.5 Features

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.

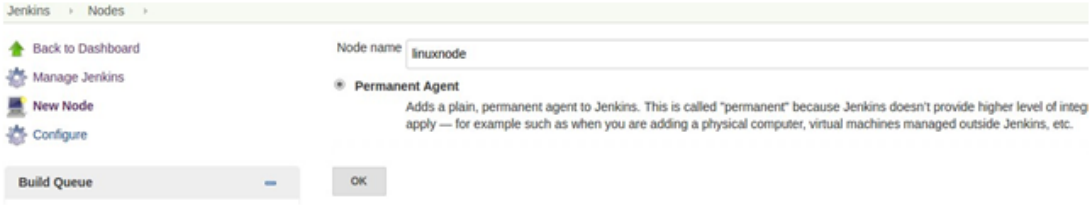
First setup the node.

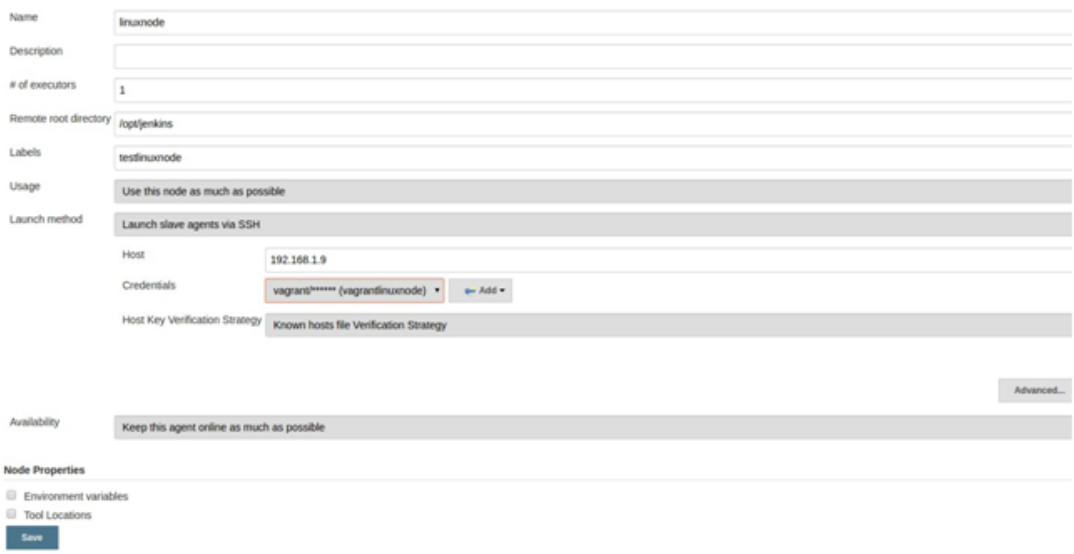
Adding A Linux Slave Node.

First setup the node.

```
sudo spt-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





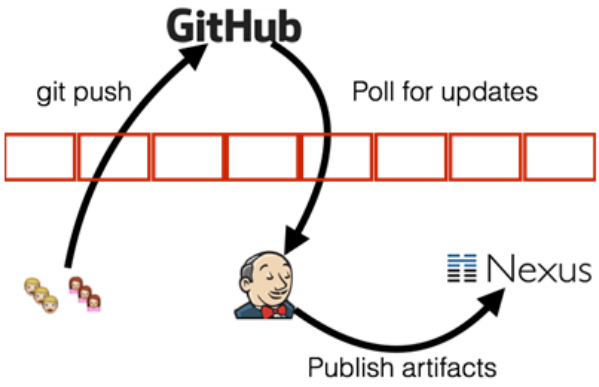
Continuous Integration Project.

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.



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.

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

Create New Build Job

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

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

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

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.

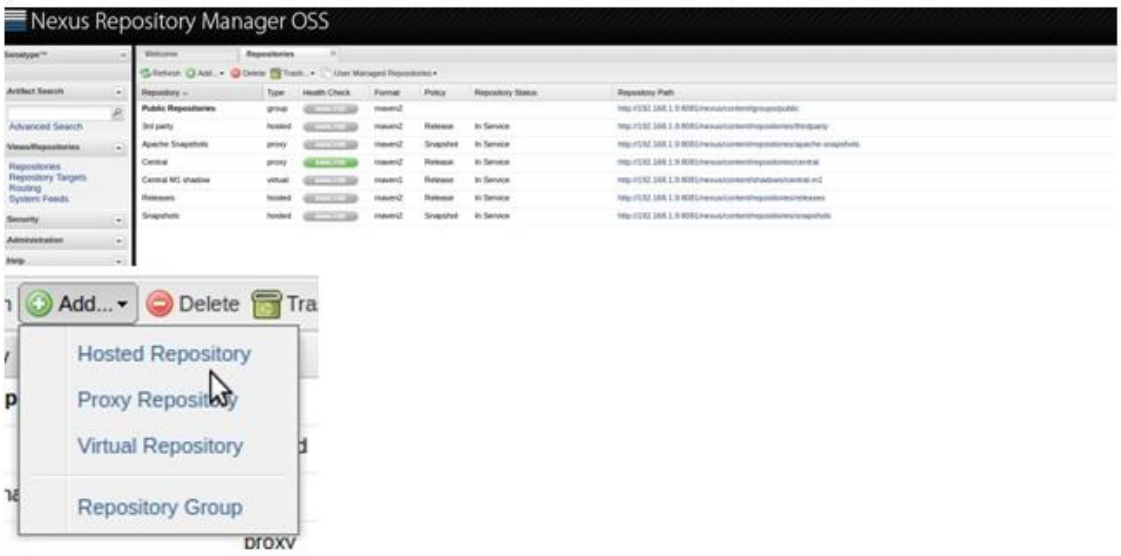
Create a centos vm or cloud instance and login to it.

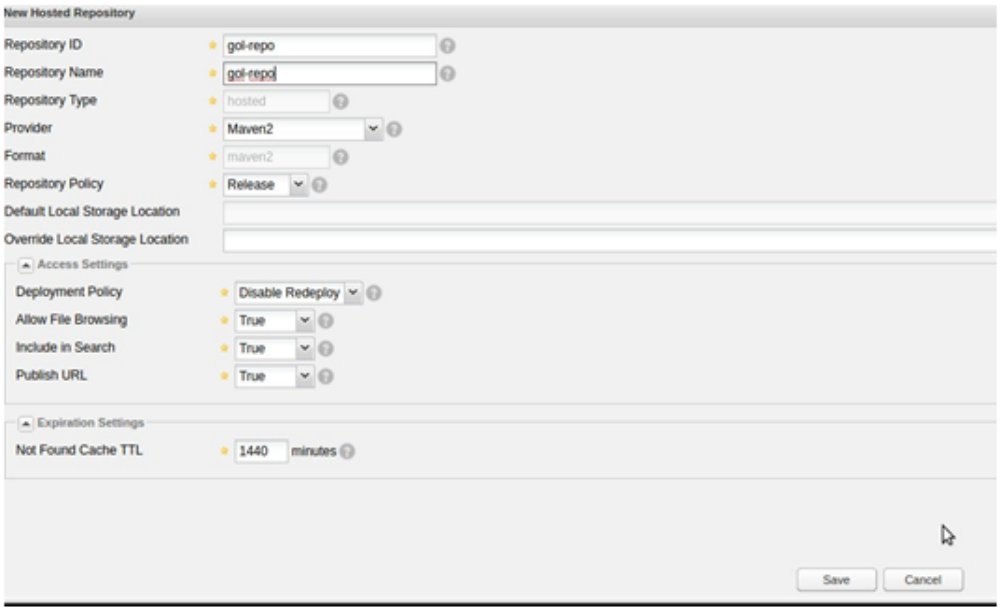
Follow below steps to setup Nexus

```
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

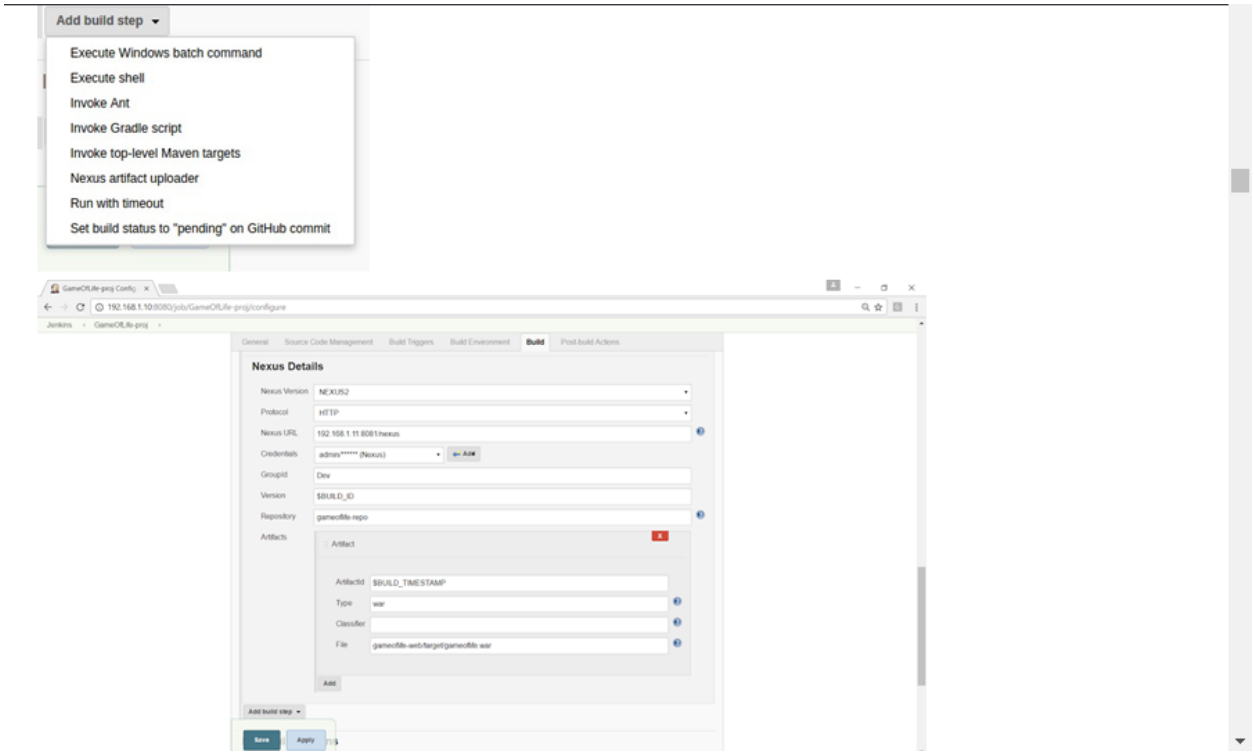
1. From browser hit URL <Nexus server IP>:8081/nexus.
2. Click login button and enter the credentials. (admin/admin123)
3. Create hosted repository named "gol-repo" with all default settings

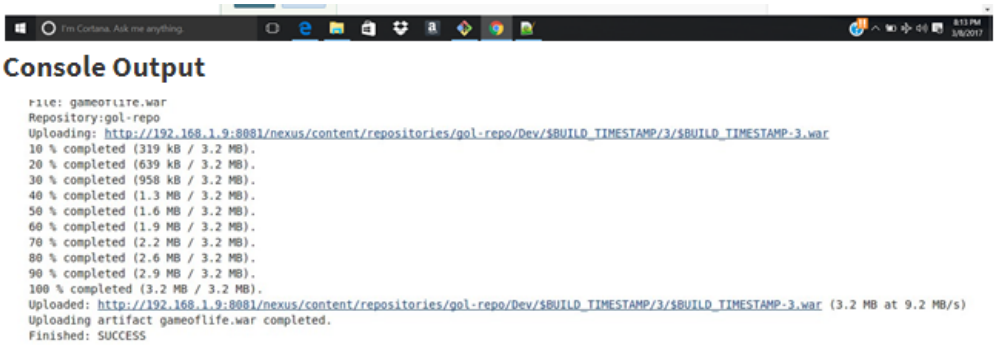




Configure Nexus Plugin To Push The Artefact To Nexus Repository.

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



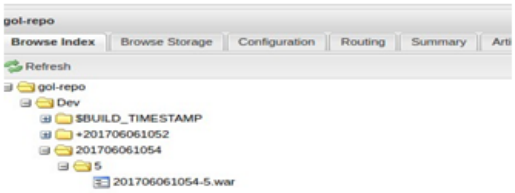


Console Output

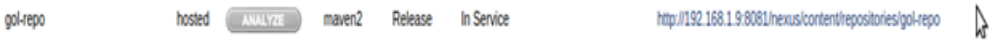
```
File: gameoflife.war
Repository:gol-repo
Uploading: http://192.168.1.9:8081/nexus/content/repositories/gol-repo/Dev/$BUILD_TIMESTAMP/3/$BUILD_TIMESTAMP-3.war
10 % completed (319 kB / 3.2 MB).
20 % completed (639 kB / 3.2 MB).
30 % completed (958 kB / 3.2 MB).
40 % completed (1.3 MB / 3.2 MB).
50 % 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 (2.6 MB / 3.2 MB).
90 % completed (2.9 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.



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

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.

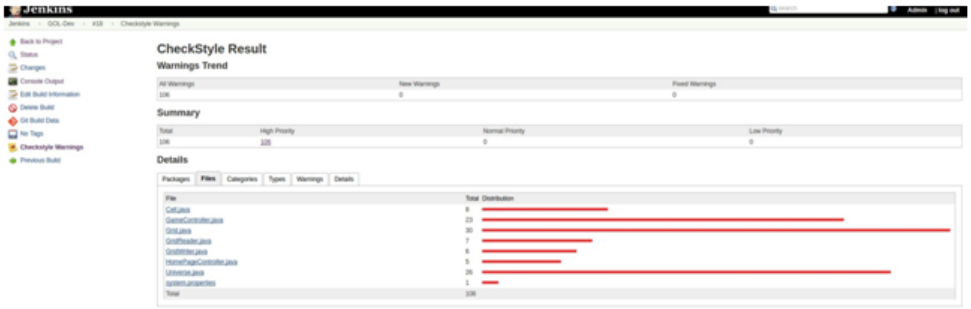
Steps:

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



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

```
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

```
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" 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.

```
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|0:0:0:0:0:0:0:1" />
</Context>
```

With

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

Stop And Start Tomcat.

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

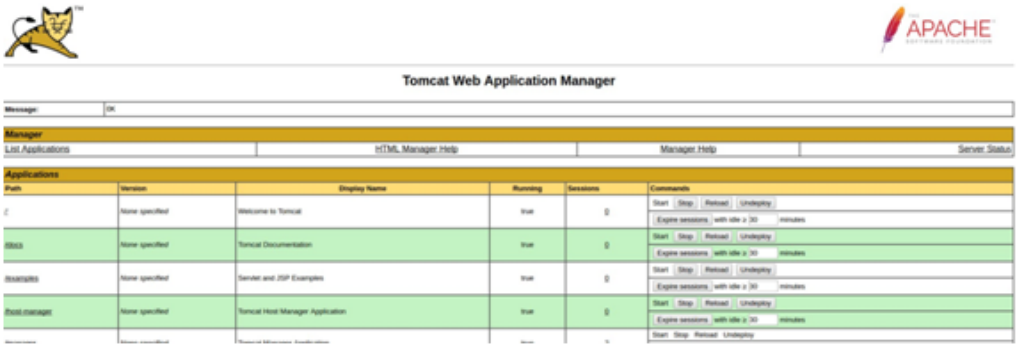
Verify The Settings

http://tomcatIP:8080/manager/

Username: tomcat

password: tomcat

You should see below mentioned screen.



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.

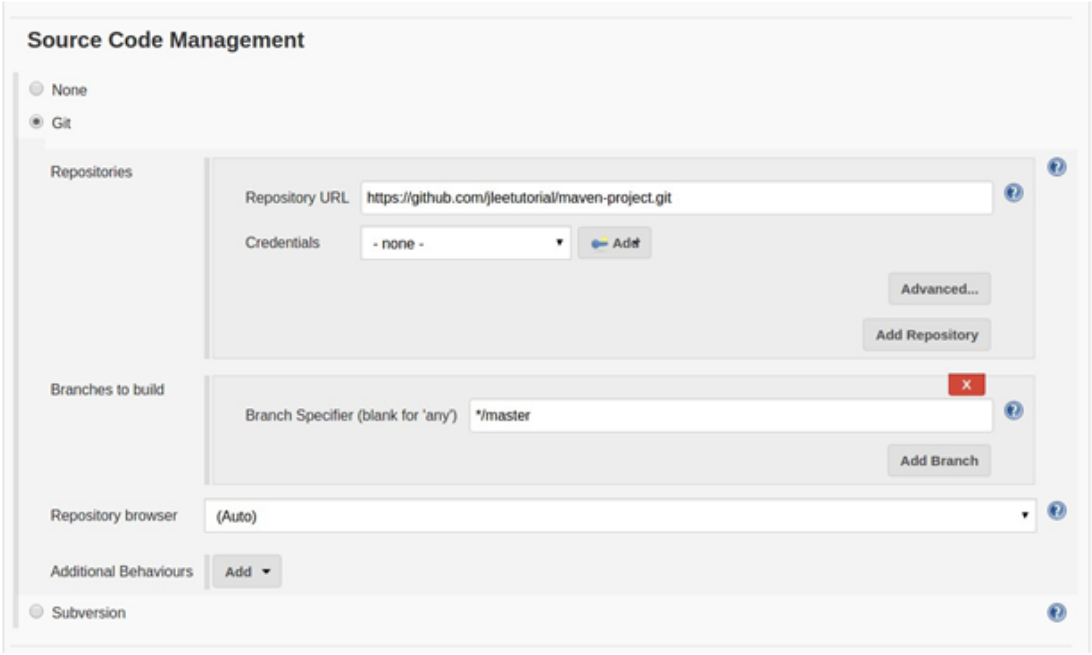
1. Package job – This job will checkout source code from git repository, use Maven to build the code, archive the artefacts, trigger Static code analysis job & staging deploy job
2. Static code analysis – This job will checkout source code from git repository, use Maven to run checktyle code analysis 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.

Prerequisites:

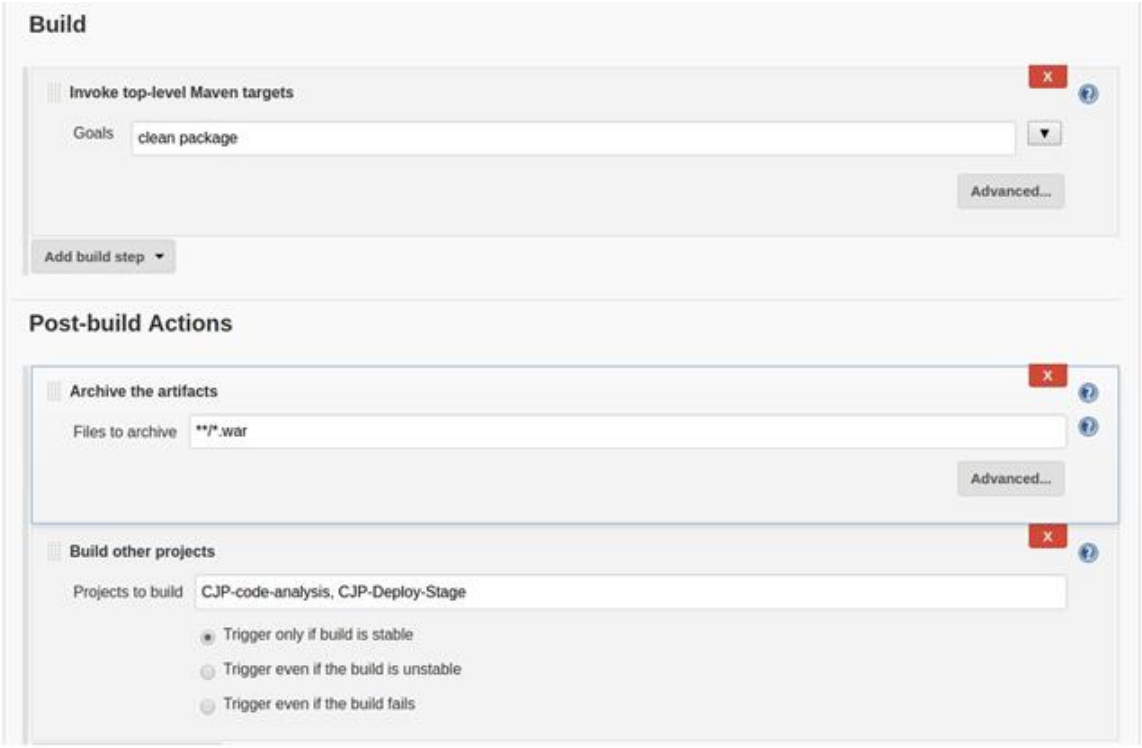
1. Jenkins server: - Jenkins server should have openjdk 1.7, Git, Maven (Follow Jenkins setup doc) 2. Plugins: - git, maven, copy artifacts, deploy to container, checkstyle, build pipeline 3. Staging tomcat server: - Create a vm or ec2 instance with ubuntu OS & Follow tomcat setup doc. 4. Prod tomcat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

Create Jenkins Jobs:

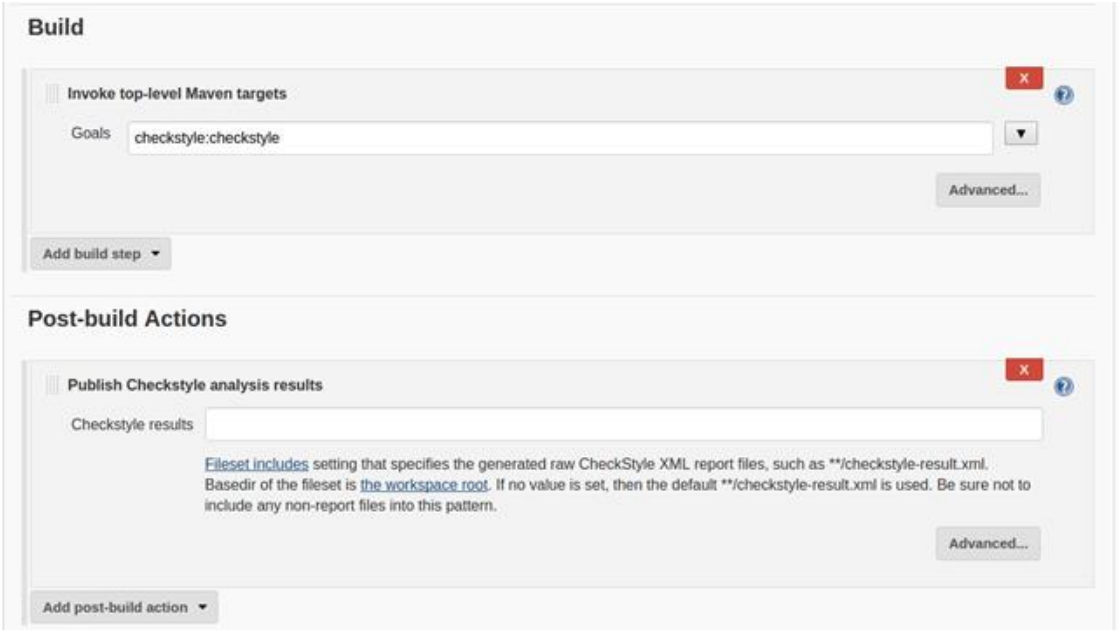
1. CJP-package: - Create a new job with CJP-package name, freestyle project. Refer below mentioned screenshots to setup the job

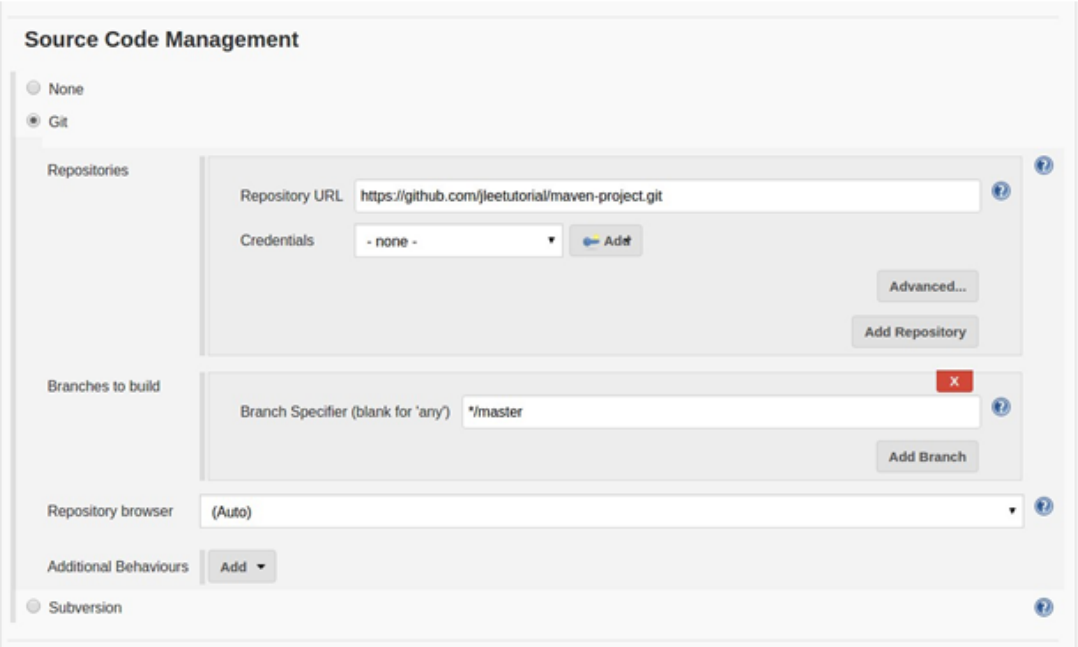


2. CJP-code-analysis: - Create a new job with CJP-code-analysis name, freestyle project.

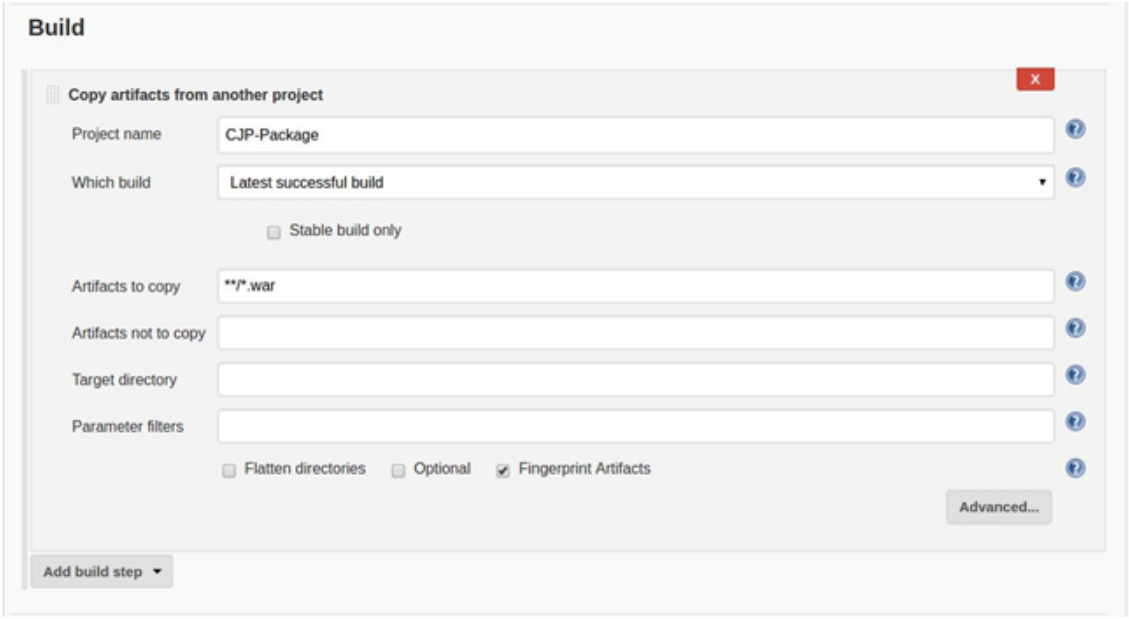


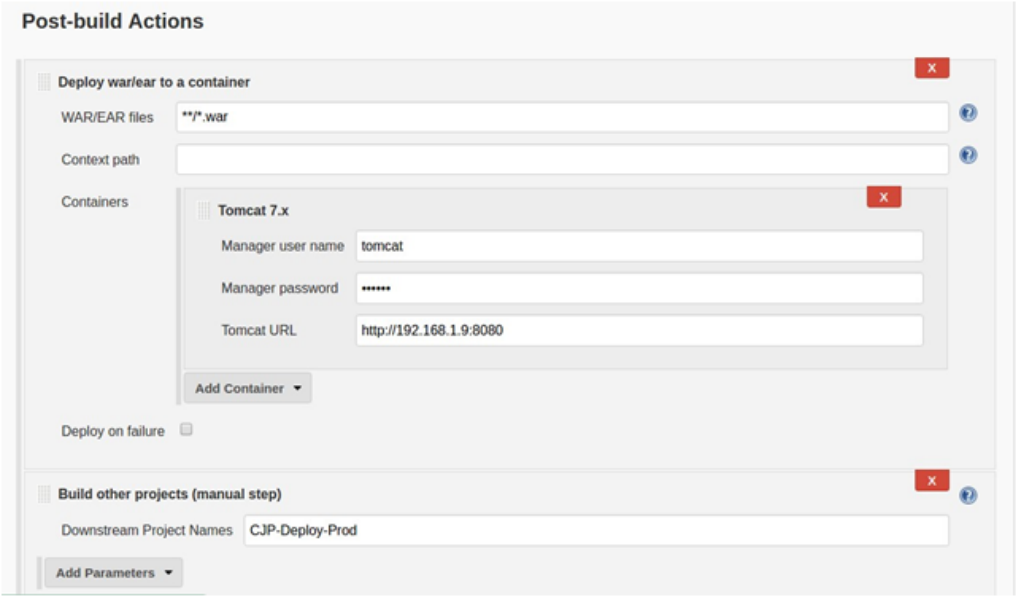
Refer below mentioned screenshots to setup the job





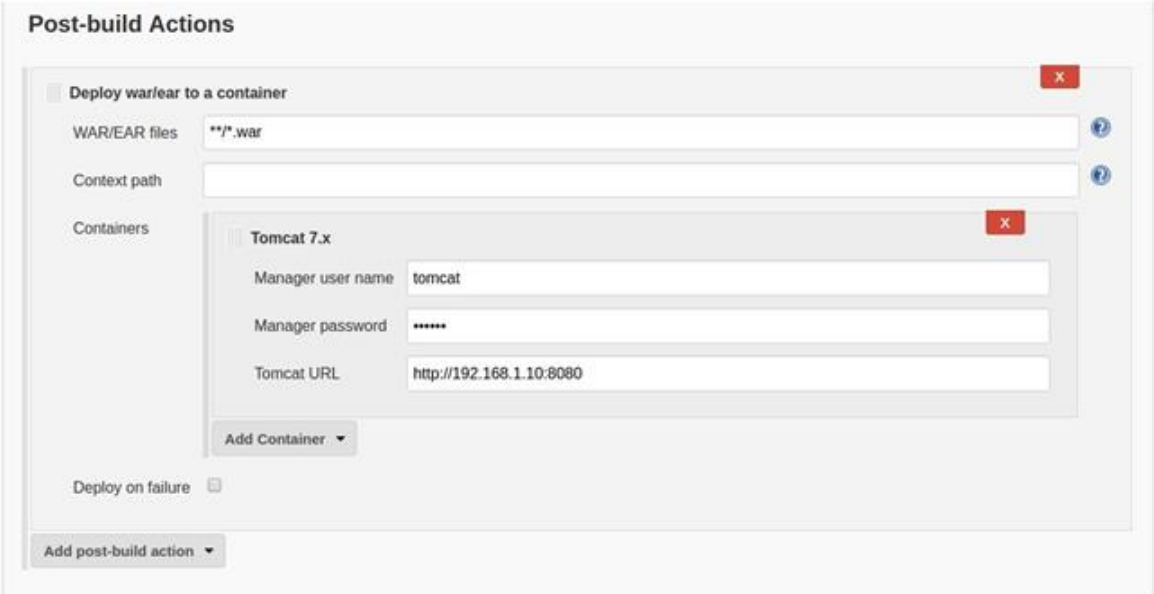
3. CJP-Deploy-Stage:- Create a new job with CJP-Deploy-Stage name, freestyle project. Refer below mentioned screenshots to setup the job



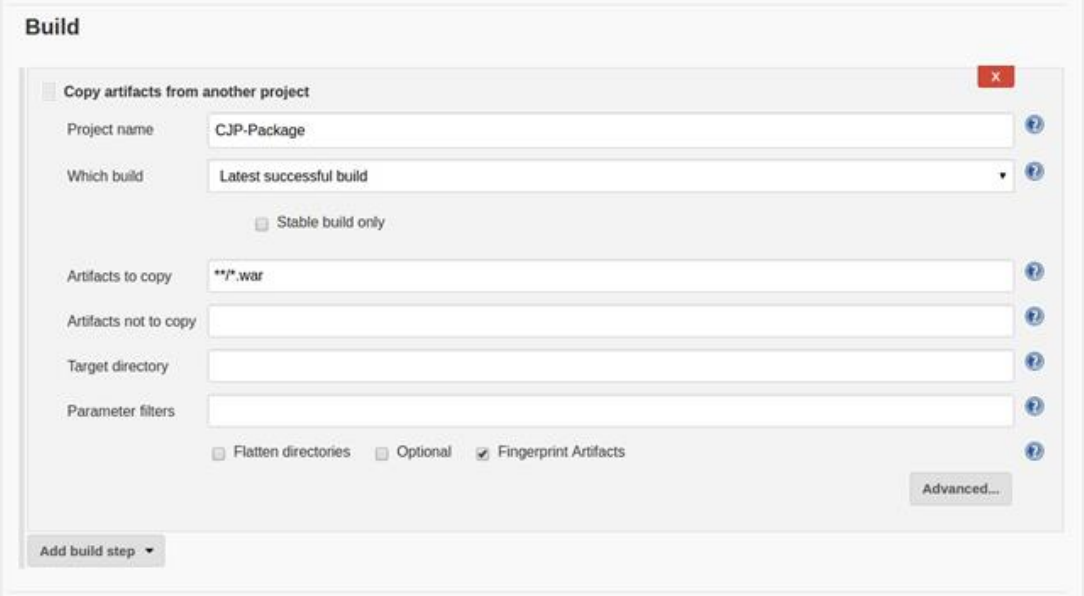


Deploy-Prod: - Create a new job with CJP-Deploy-Prod name, freestyle project.

Refer below mentioned screenshots to setup the job



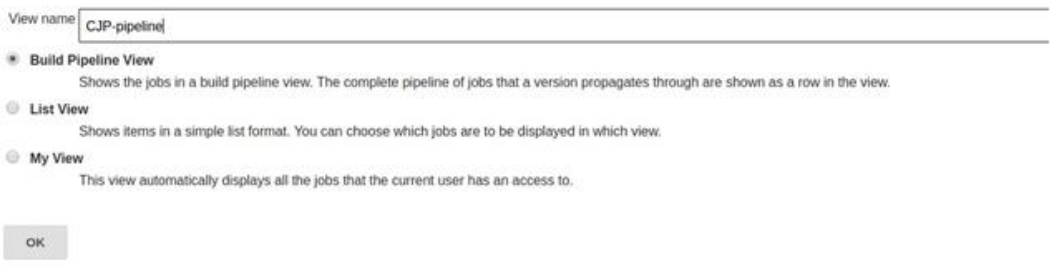
Setup Build Pipeline View



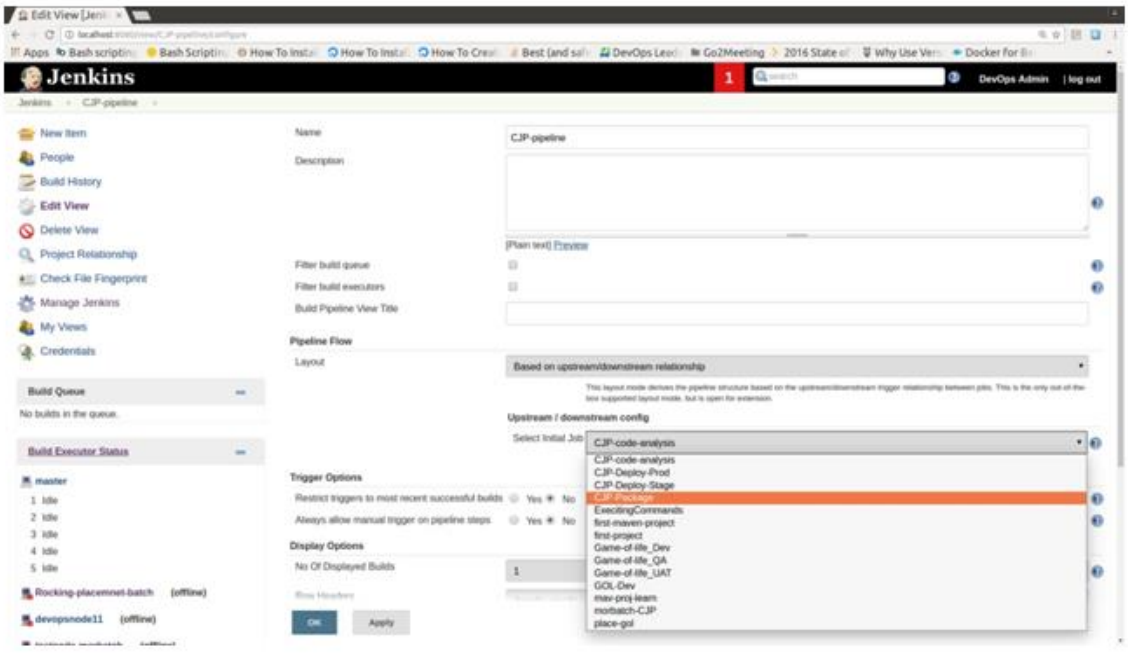
Go to Jenkins main dashboard

Click on the plus symbol

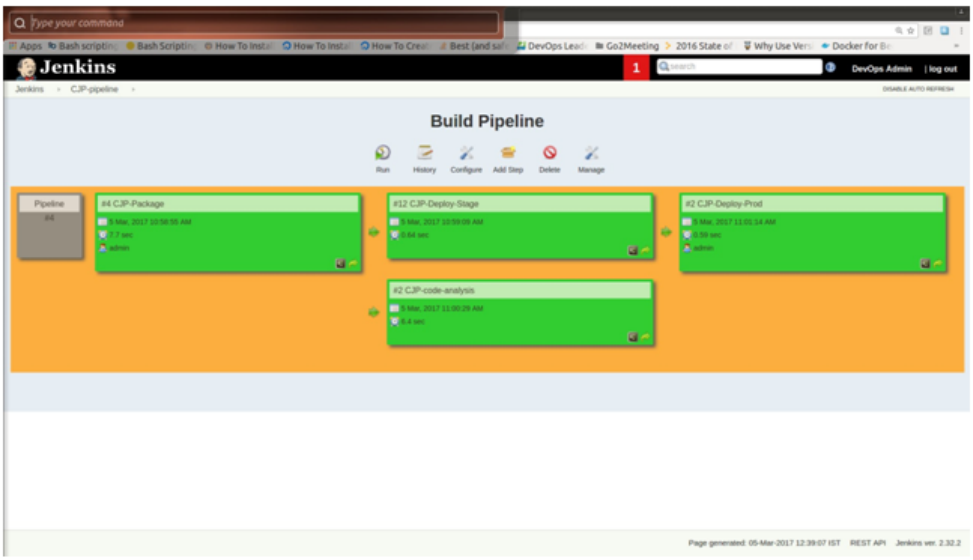
Select Build Pipeline View, give pipeline view a name and click OK



Select initial Job – CJP-Package & click OK



Click run and verify if the entire pipeline works

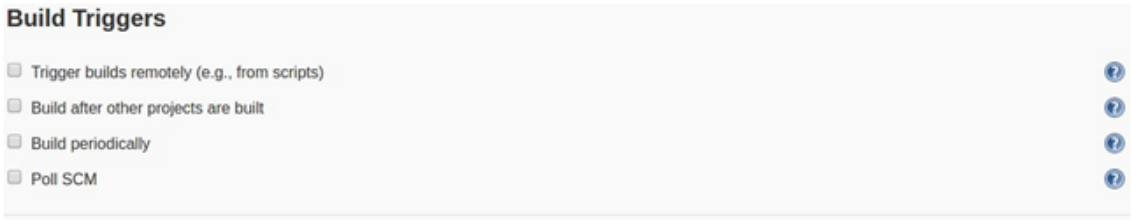


Jenkins Build Triggers

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

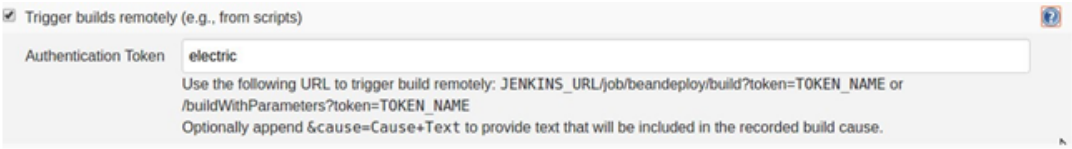
But there are other ways by which the jobs can be executed those are called as Build

Triggers.



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

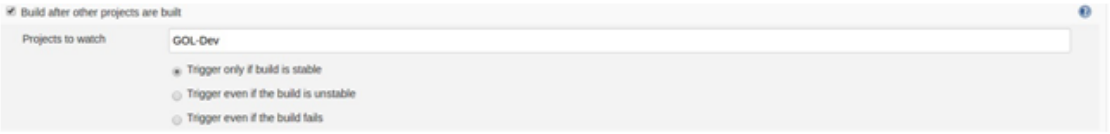
4. Github webhooks

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.

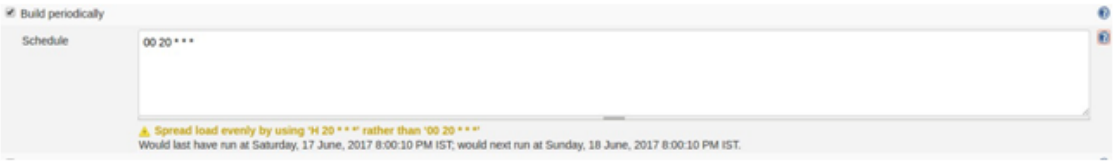
Build After Other Projects Are Build.

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.

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

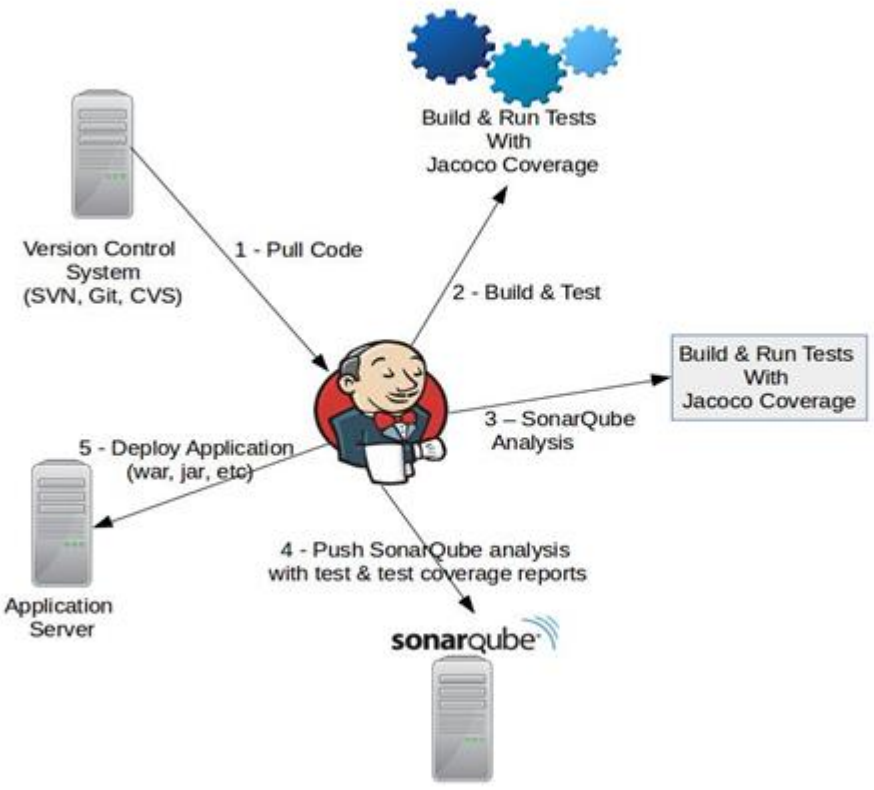
1. SonarQube scanner
2. Maven

SonarCode scanner process in Jenkins

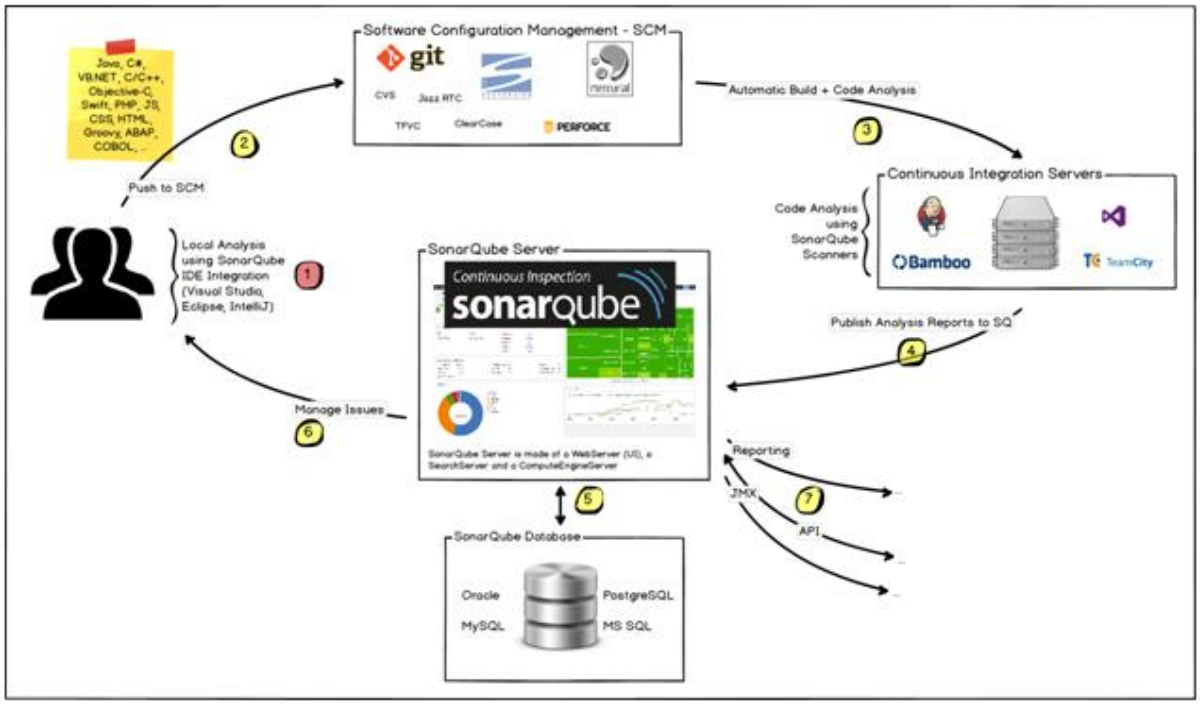
1. Pull code From Git.
2. Build and Run Unit Tests
3. SonarQube analysis with sonarqube scanners.
4. Push SonarQube analysis coverage reports to SonarQube Dashboard.
5. We can store the SonarQube data in DataBase (Mysql).
6. Deploy applications to server.

SonarQube Lifecycle

Jenkins


Build & Run Tests
With
Jacoco Coverage
Version Control
System
(SVN, Git, CVS)
1 - Pull Code
2 - Build & Test
Build & Run Tests
With
Jacoco Coverage
3 – SonarQube
Analysis
5 - Deploy Application
(war, jar, etc)
4 - Push SonarQube analysis
with test & test coverage reports
Application
Server
sonarqube

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

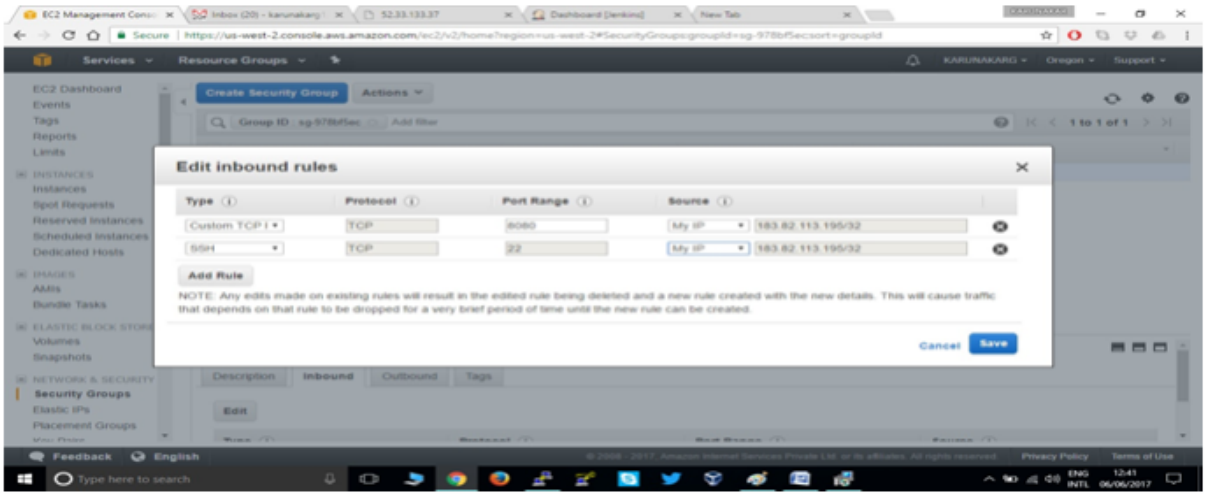
2. Type – t2.micro

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)

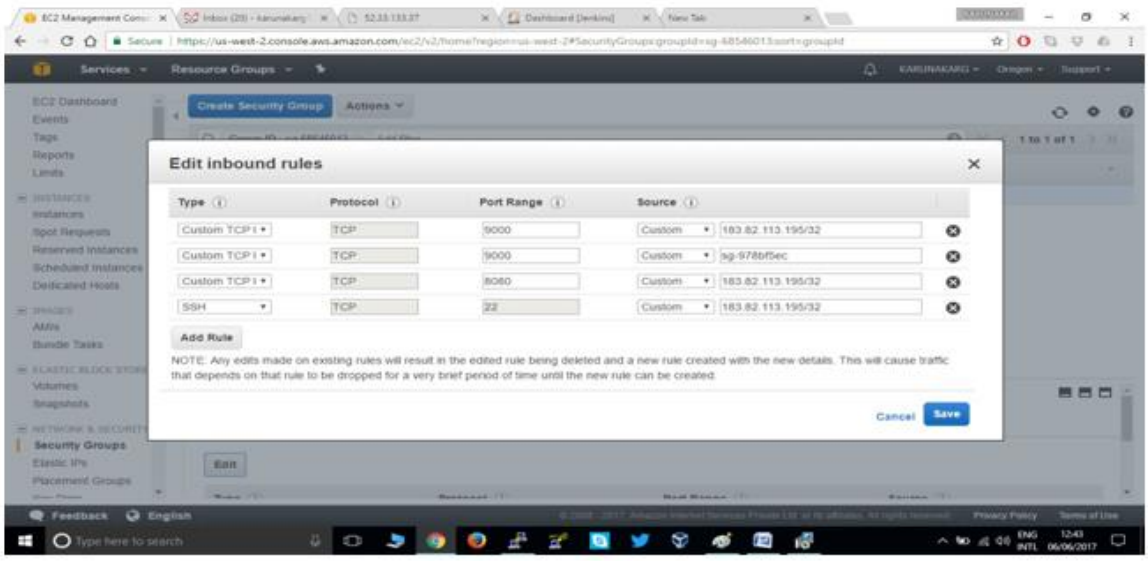
Security Group for SonarQube

Type Custom TCP ->Port 9000->source MyIP

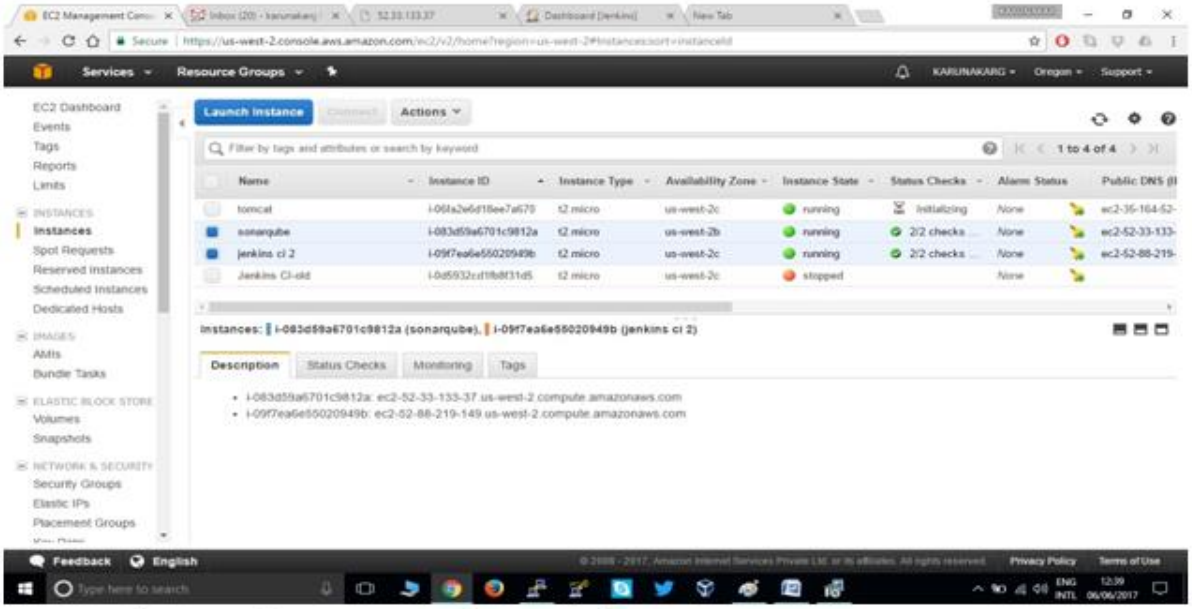
Type Custom TCP ->Port 8080->source JenkinsSg

Type Custom TCP ->Port 8080->source MyIP

SSH -->Port 22 -> source MyIP



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



SonarQube Instalation 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.

Adding Java PPA to Our Ubuntu repository

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

Updating apt repository

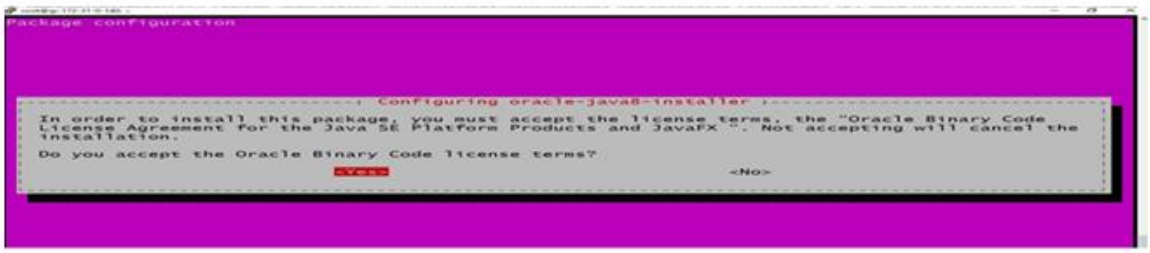
```
apt-get update
```

Installing java on EC2 Ubuntu 16.10 Instance

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

Here Press on “OK”

Here Press on “Yes “

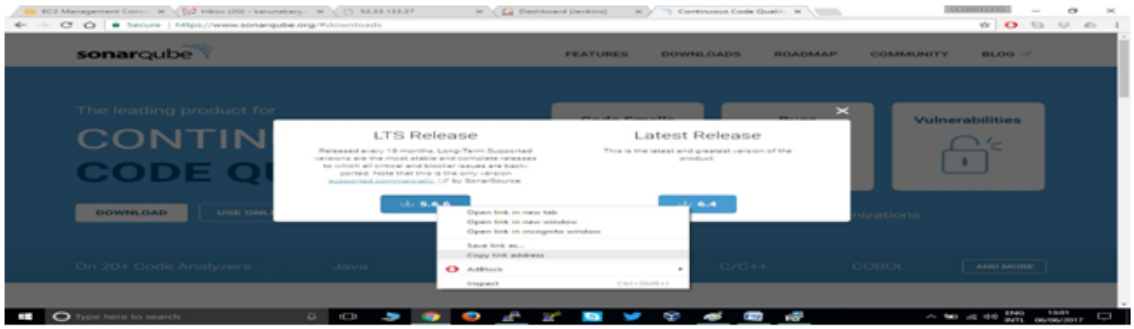


Here We are Creating soft link for java = default java

```
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 Satart the mysql server

```
$ 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
```

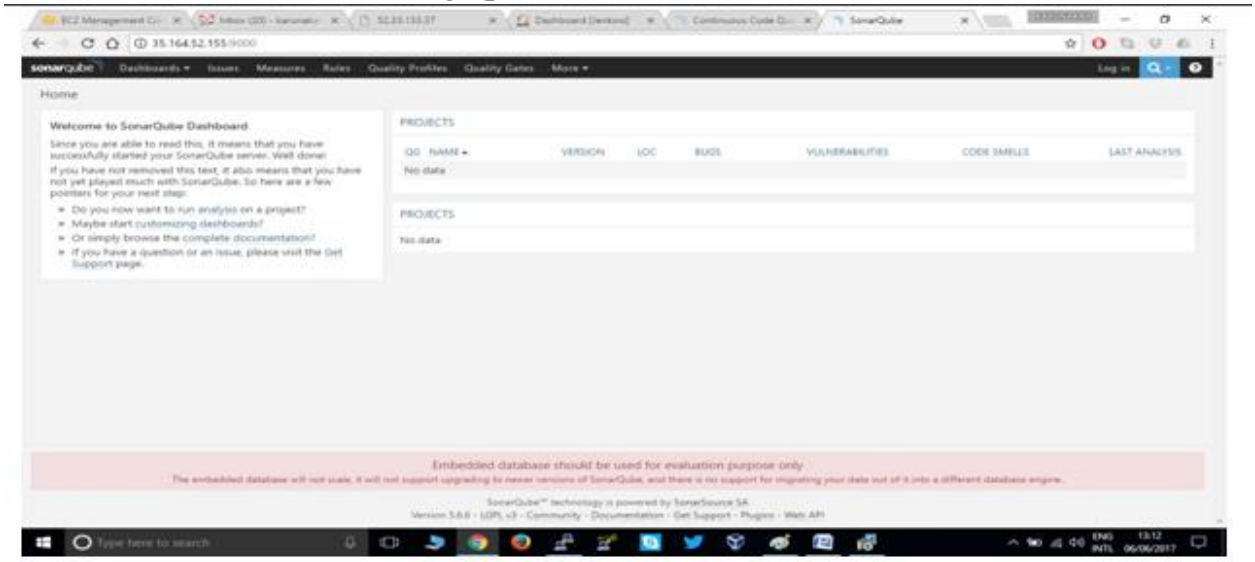
Starting The Sonarqube Server

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

stop the firewalls

```
service ufw stop
```

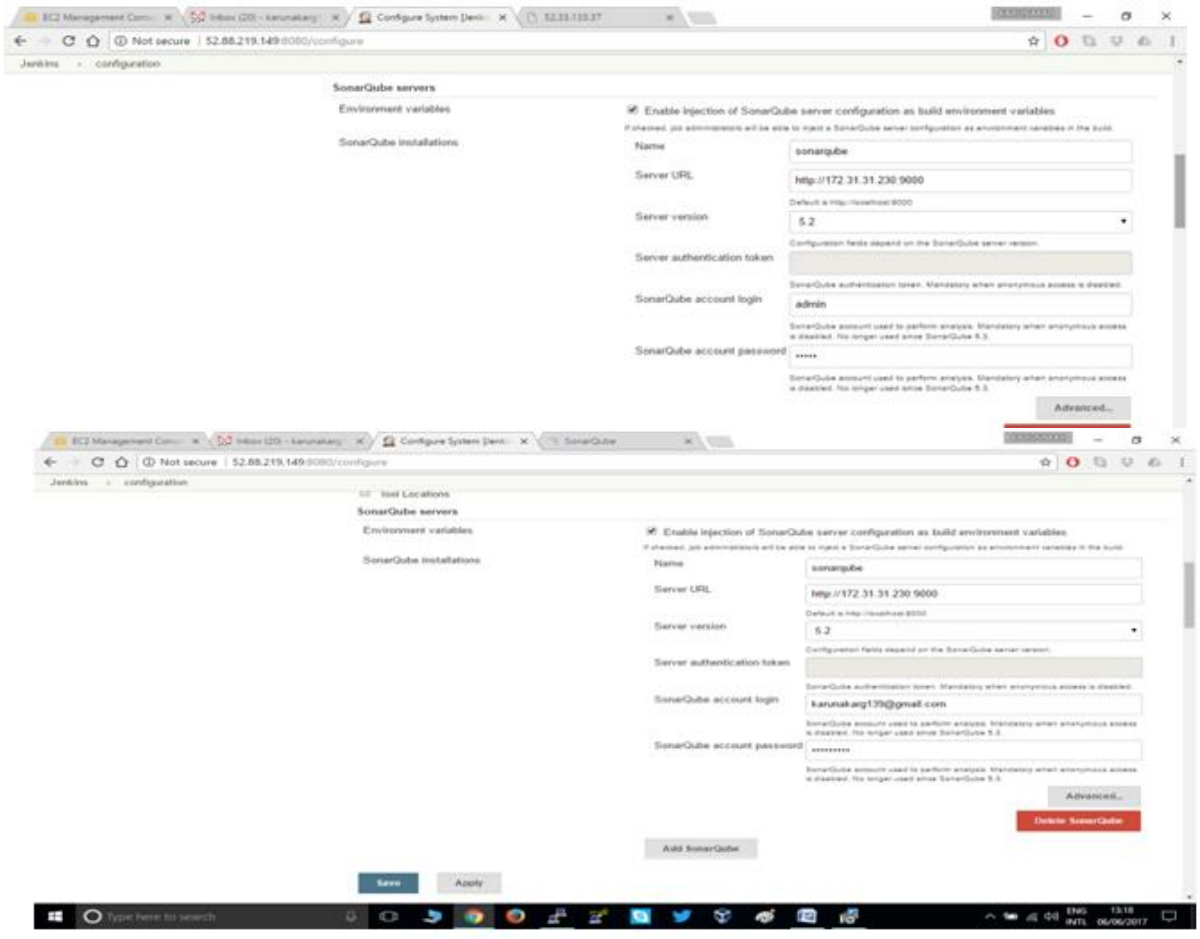
SonarQube DashBoard Mainpage:



Jenkins Job Setup

Here we are Installing Sonarscanner Plugin in Jenkins

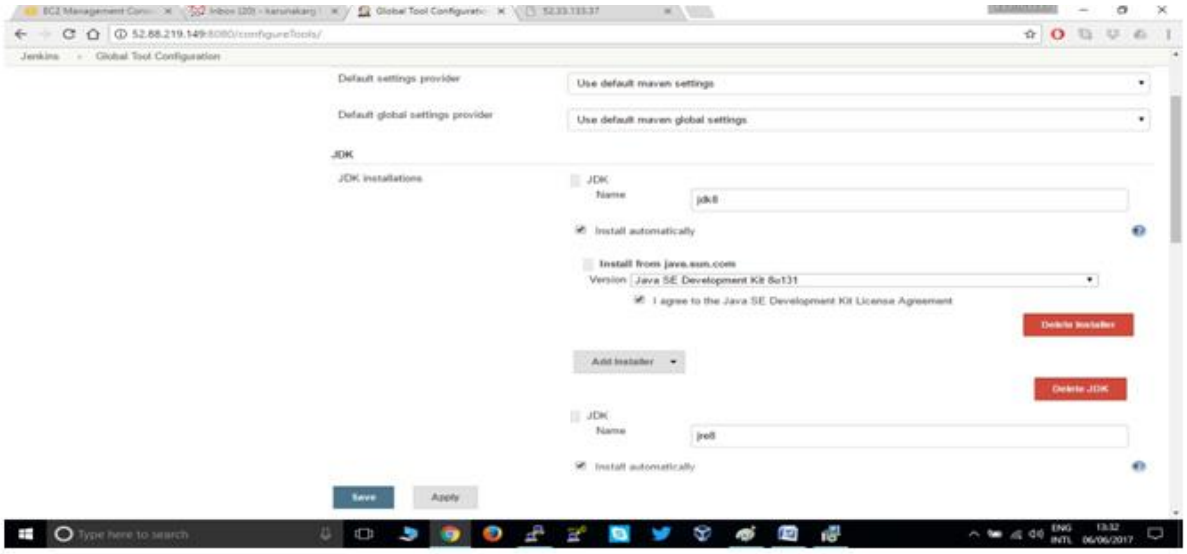
Manage jenkins ->System Configuration -> assign the sonarqube credentials

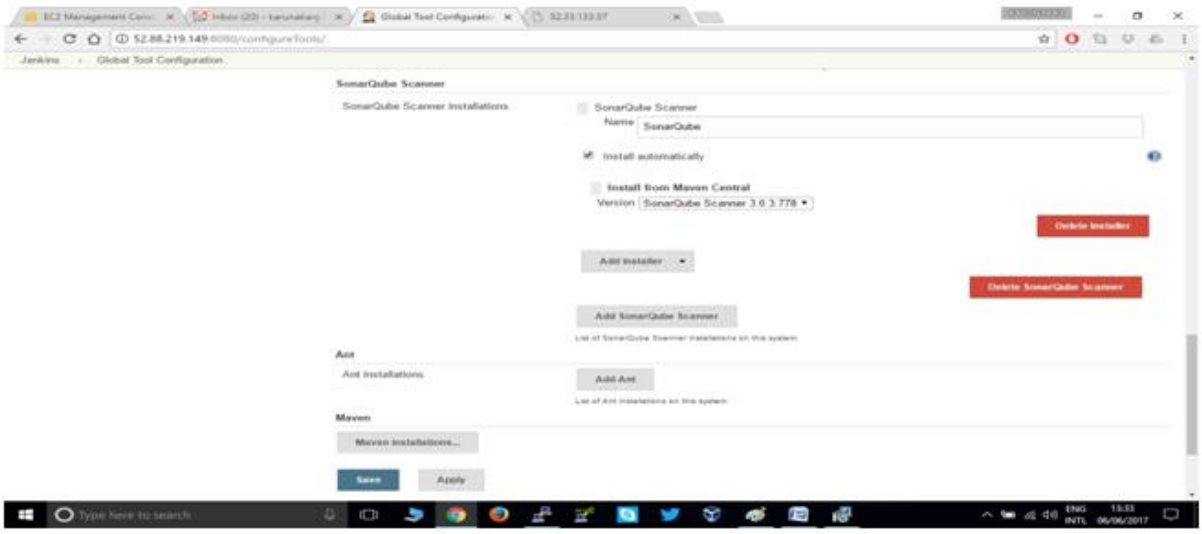


Global Toll configuration->JDK name =jdk8->

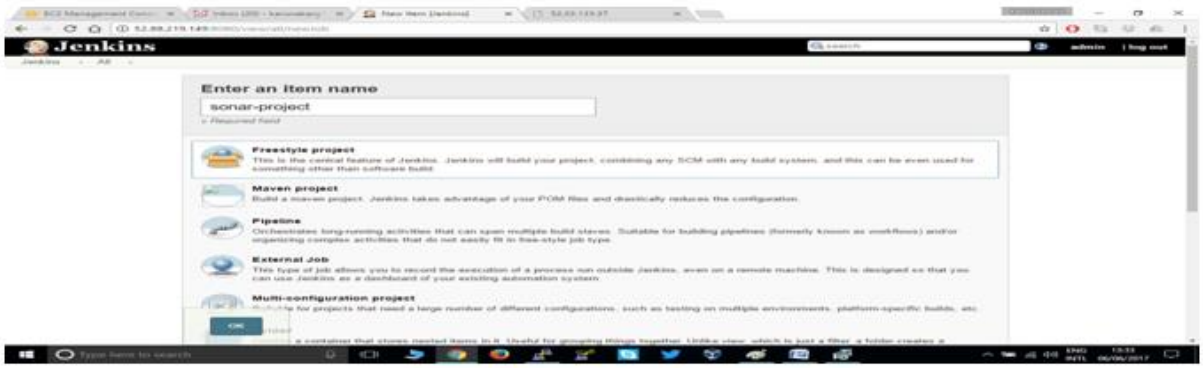
SonarQubeScaner-> from Maven assign sonar scanner version

Jenkins

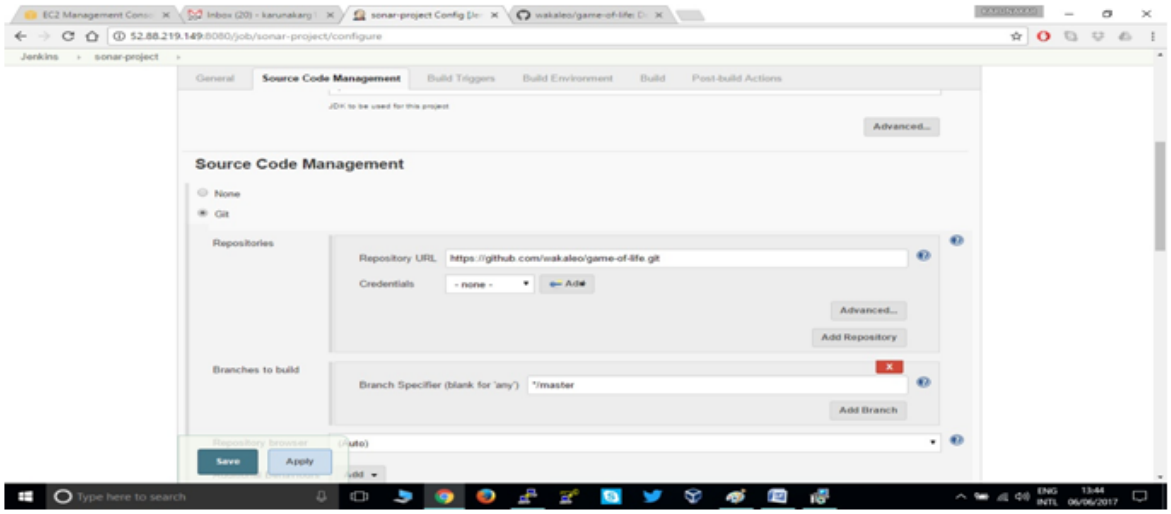




Creating New Project with name of Sonar-project



Assigning Git repository in Source code management



=>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

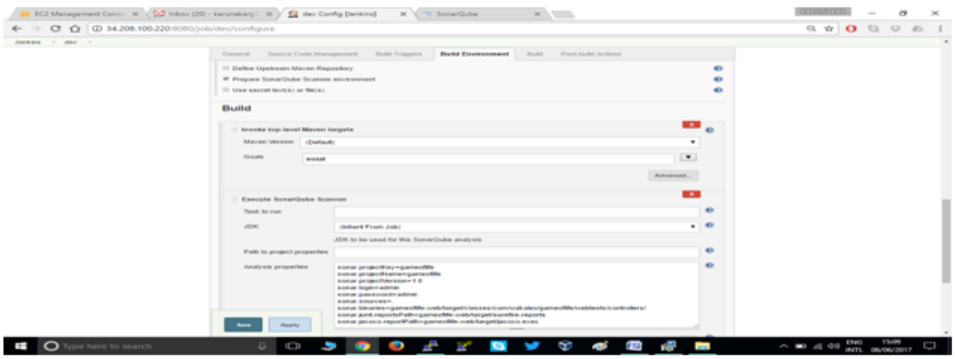
sonar.sources=.

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 snopshot is showing that sonarproperties and credentials of the sonarqube

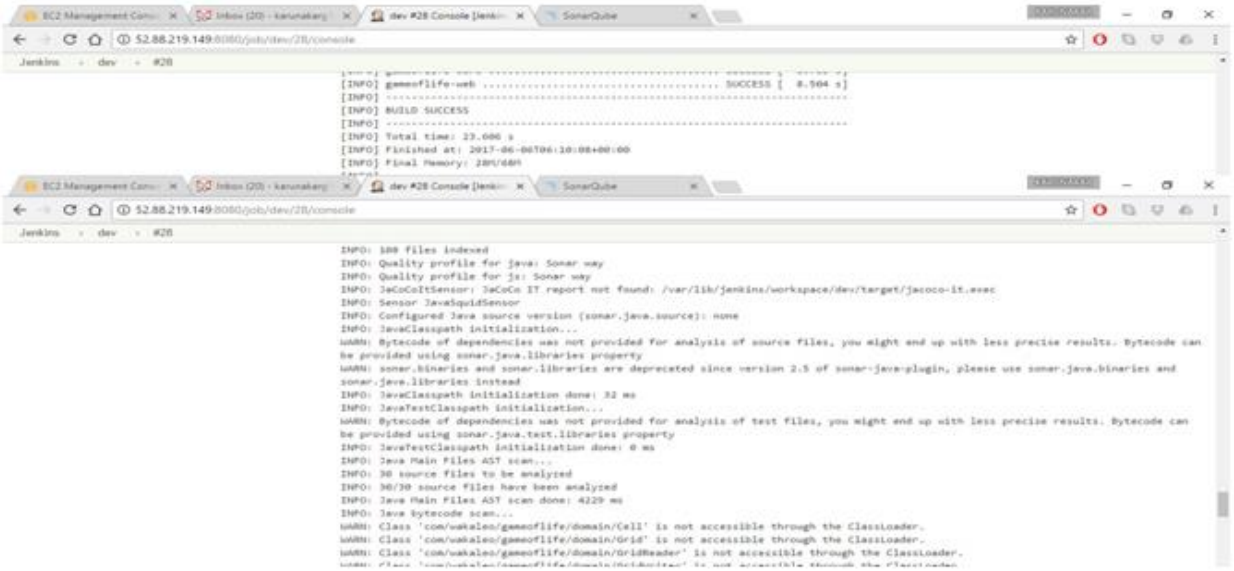


=>After entering the sonar properties save the project and go to Build now,

=>Below snapshots showing that build process as well as sonar scanner tests

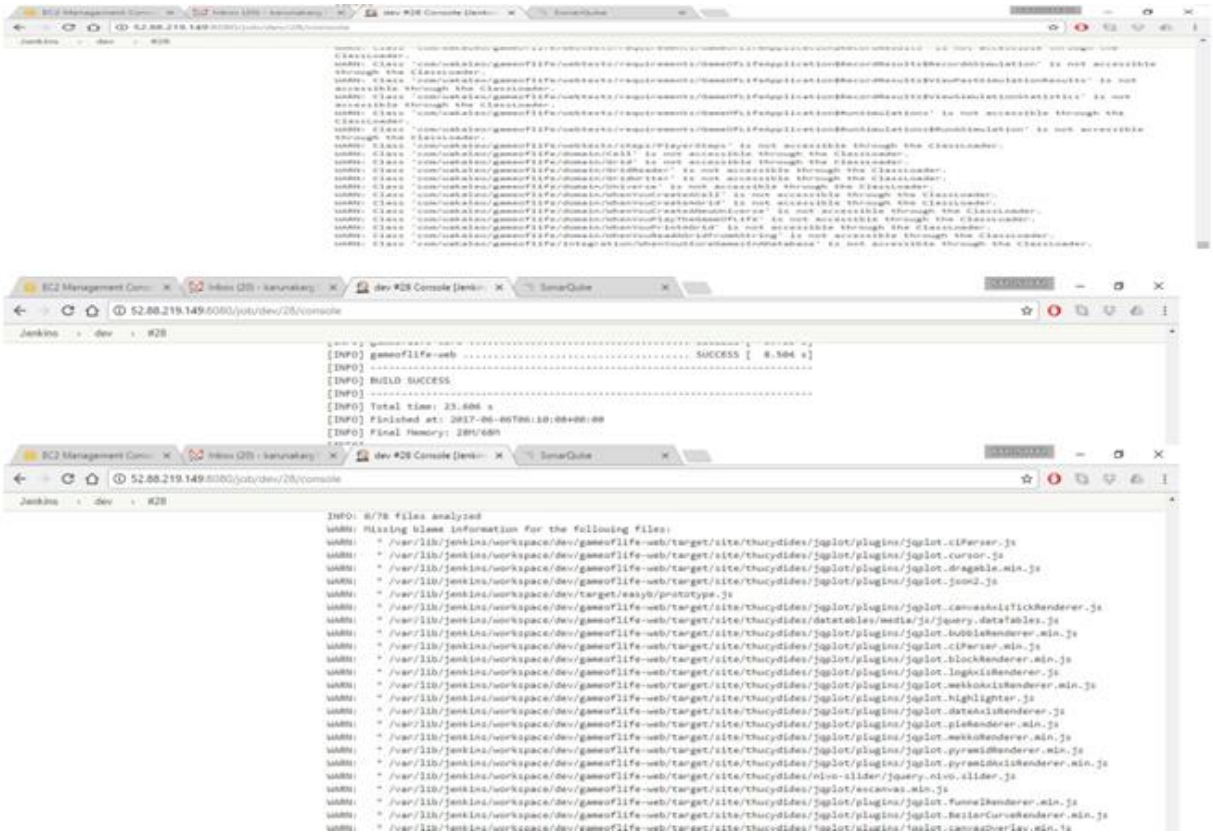
=>After Success The job that sonartest files and reports are stored in Var/lib/jenkins /workspace directory so these files and

reports we have to give in sonarDashboard to see the output.

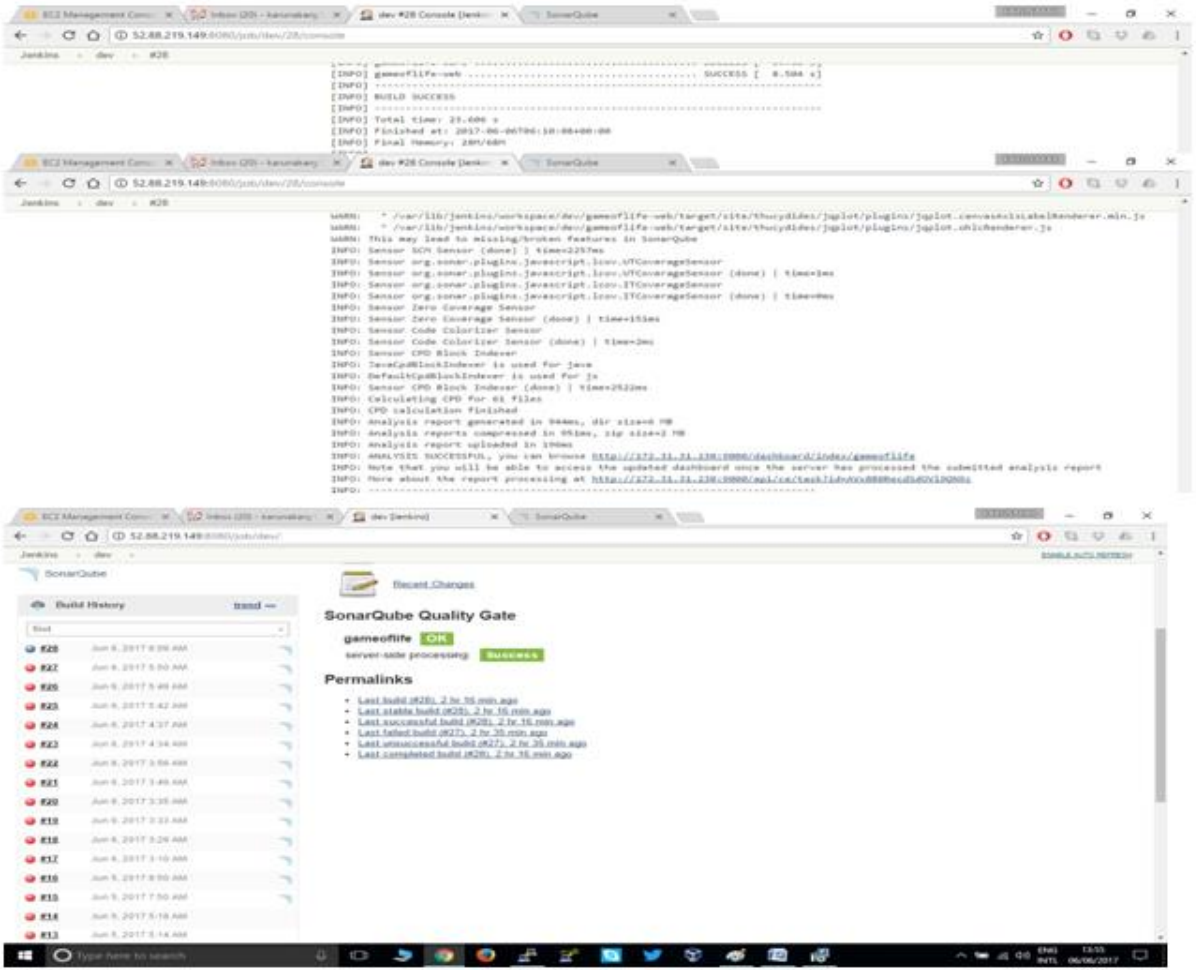


Naren Technologies

Jenkins



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

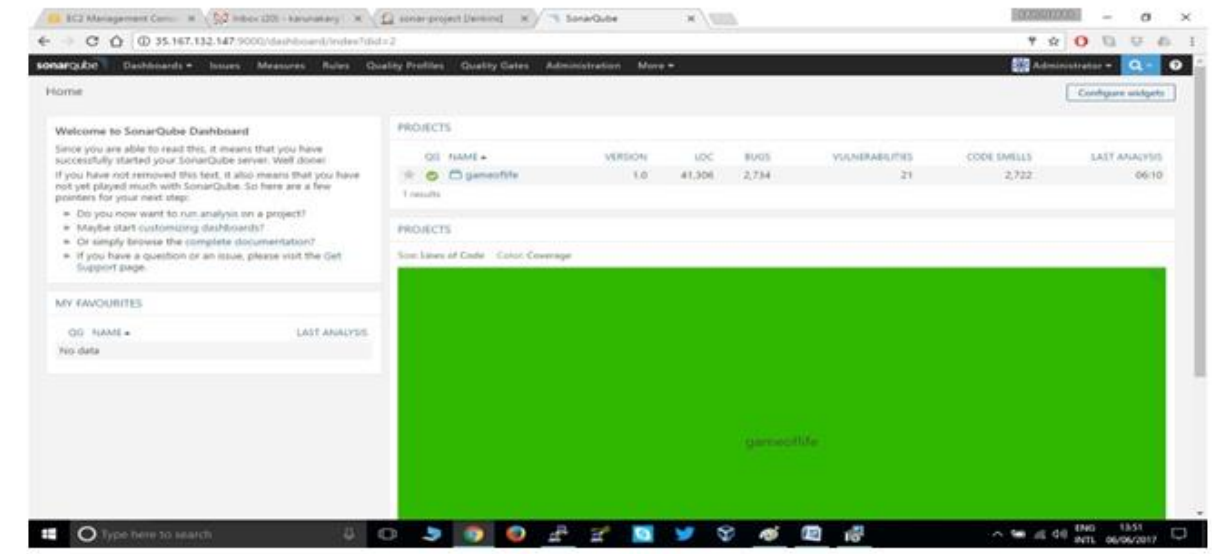
=>code smells

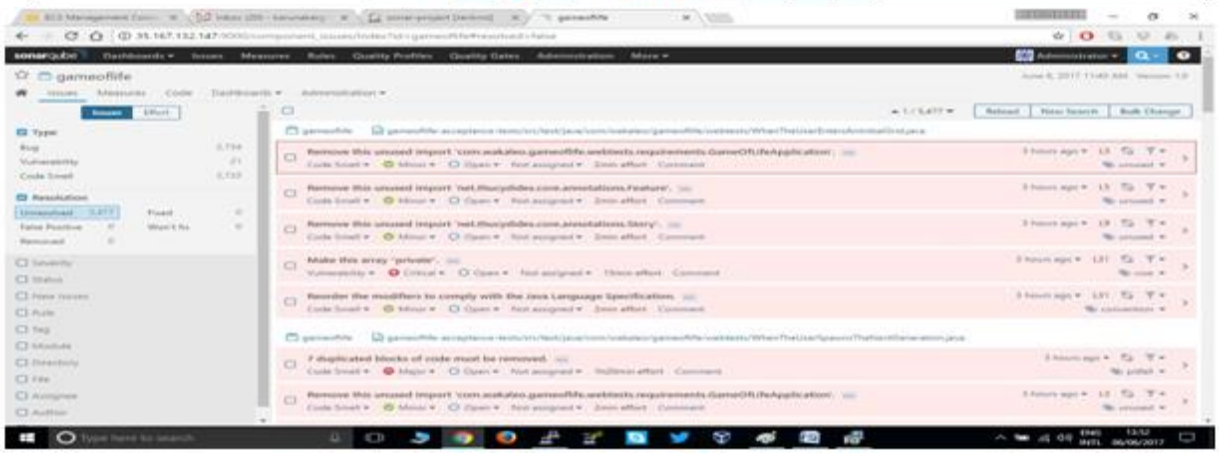
=> vulnerabilities

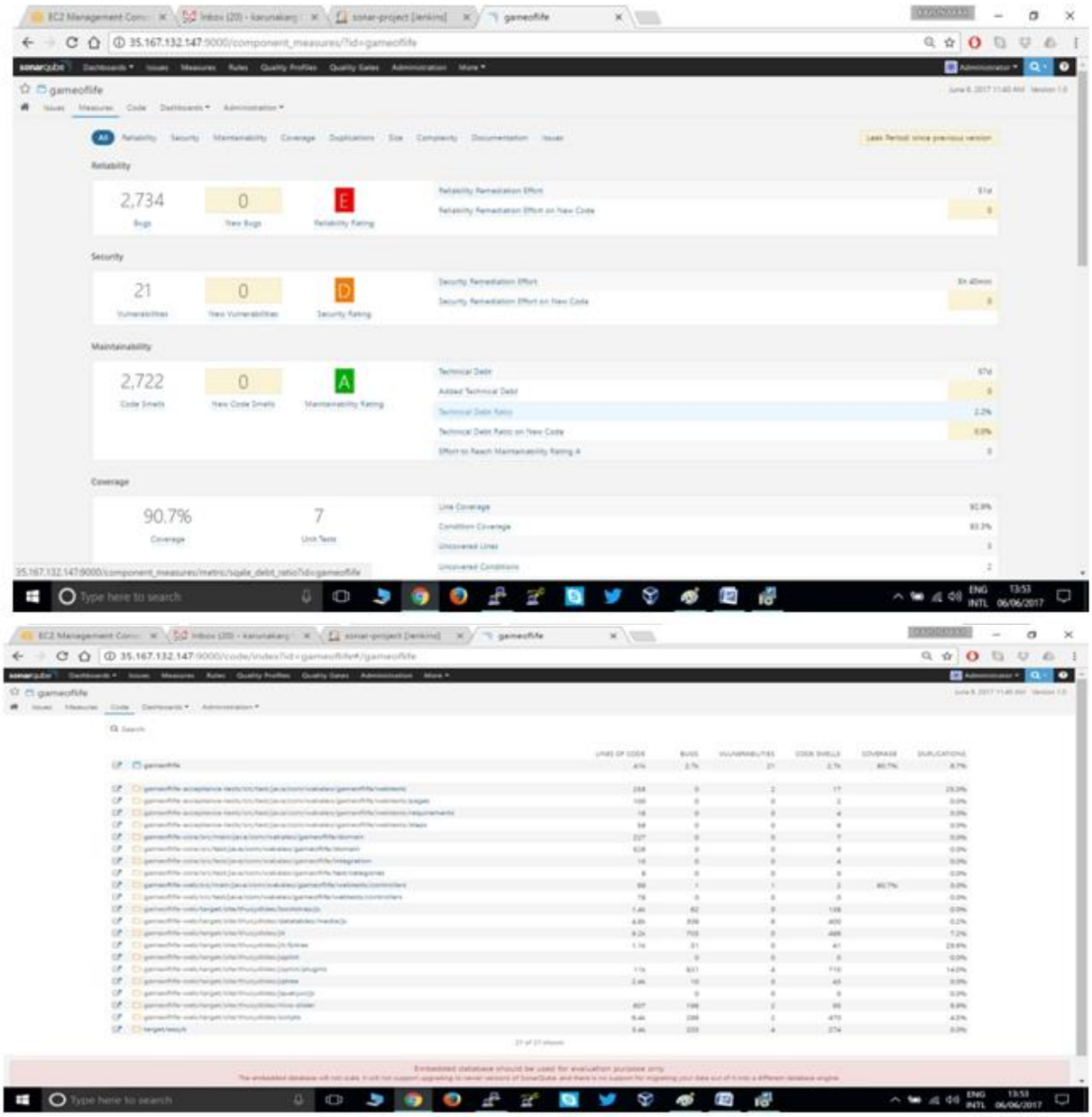
=>unit tests

=>Duplications

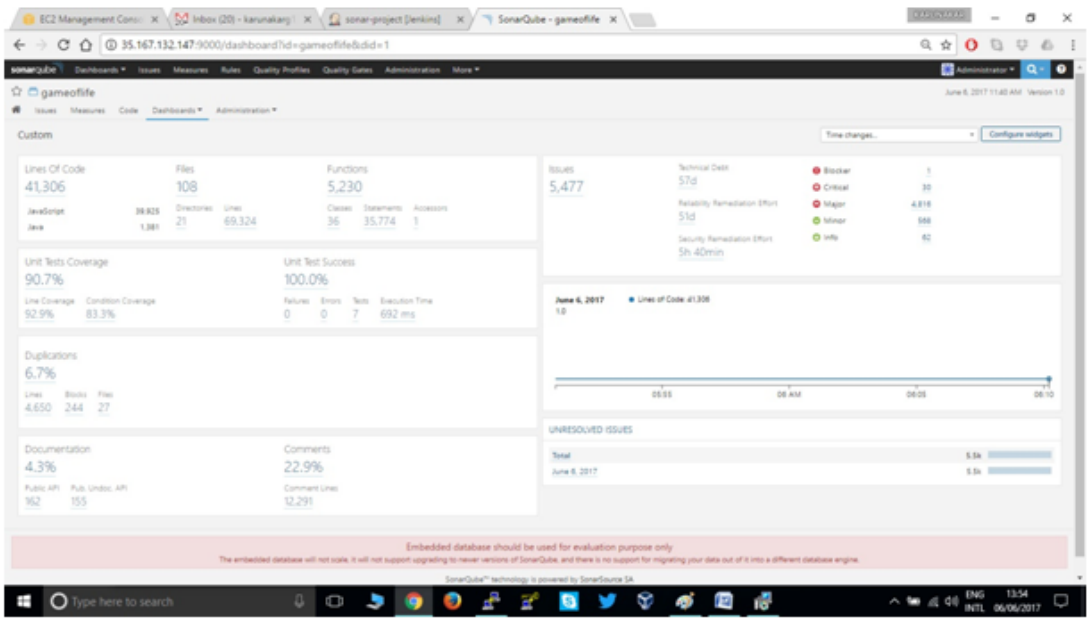
Jenkins


Welcome to SonarQube Dashboard
PROJECTS
MY FAVOURITES
No data
gameoflife


gameoflife



SonarQube Dashboard



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.

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

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.

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

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

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

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

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

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),

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.