

INtroduction:	3
Continuous Integration :	3
Benefits of continuous integration :	3
Steps in Continuous Integration:	3
Continuous delivery :	4
Hardware Required for Server :	4
Tools :	4
Jenkins :	5
System Requirements :	5
Advantages of Jenkins:	5
What Jenkins can Do?	5
Standard installation steps :	6
Create Job on Jenkins :	6
Manage Jenkins Plugins :	17
Installing a plugin :	17
From the web UI :	18
Using the Jenkins CLI:	18
Advanced installation :	18
Jenkins Master Slave :	19
Jenkins Theme Customization :	19
Jenkins Theme Customization Steps :	20
Add Simple Theme plugin :	20
Customize Theme :	20
Custom theme :	20
Existing themes :	20
Jenkins Backup & Restore :	21
Upgrade Jenkins :	26
Jenkins CLI :	27
Jenkins Security :	27
Troubleshooting:	28
Configuration management:	29
Introduction To Ansible :	29
Virtualbox :	29

Introduction :	29
Why is virtualization useful?	30
Install Virtualbox :	30
Vagrant :	31
What is Vagrant?	31
Why Vagrant and What are the Advantages of using it?	31
Installation Steps :	31
Docker :	32
Introduction :	32
Docker Architecture :	33
The Docker daemon :	33
Docker Registries :	33
Docker Images :	33
Docker Container :	34
Docker Installation :	34
Prerequisites:	34
Installation steps:	34
GitLab :	35
How to pull Jenkins Jobs metadata from GitLab:	36
Repository Structure of Jenkins Job Metadata on Gitlab :	36
Deployer :	37
What is deployer ?	37
Standard installation steps :	37
Benefits of deployer :	37
Using Deployer:	38
Setting up Servers :	38
Defining your Repository :	40
The Deploy Task in deployer :	40
Sample Deployer File :	42
SonarLint for Command Line :	43
Installation Steps:	43
Configuration For SonarLint :	43
Global configuration :	43
Per-project configuration :	44
Sonarqube:	44
What is Sonarqube?	44

Hardware Requirement :	45
SonarQube Rules :	45
Introduction:	45
Quality Profiles :	46
Benefits of Using SonarQube :	46

INtroduction:

Continuous Integration :

Continuous Integration is a software development practice where team member integrate their work frequently. Usually every person integrates at least daily leading to multiple integration per day. The end goal of continuous integration is to make integration a simple, repeatable process that is part of the everyday development workflow in order to reduce integration costs and respond to defects early.

Benefits of continuous integration :

- Minimize integration risk.
- Everyone commits to the mainline everyday.
- Code stability.
- At all time you know where you are, what works , what doesn't , the bugs you have in your system.
- Unit test had to pass for deploy.
- Continuous detection and fixing of integration problems.

Steps in Continuous Integration:

- Developers check out code into their private workspaces.
- When done, commit the changes to the repository.
- The CI server monitors the repository and checks out changes when they occurs.
- The CI server builds the system and runs unit and integration tests.
- The CI server releases deployable artifacts for testing.
- The CI server assigns a build label to the version of the code it just built.
- The CI server informs the team of the successful build.

- If the build or tests fail, the CI server alerts the team.
- The team fixes the issue and commit the changes to the repository.
- Continue to continually integrate and test throughout the project.

Continuous delivery :

Continuous delivery is an extension of continuous integration. It focuses on automating the software delivery process so that teams can easily and confidently deploy their code to production at any time.

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

Hardware Required for Server :

Processor	Quad core
Memory	8 GB
Disk Space	258 GB
Operating System	Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS

Note : Above configurations may vary as per requirements.

Tools :

Some of the commonly used CI tools are :

- Travis :
Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub.
- Bamboo :
Bamboo is a continuous integration server developed by Atlassian. Bamboo can run multiple builds in parallel for faster completion .In case of build failures, Bamboo provides an analysis of the failure

- CircleCI :
CircleCI enables you to automatically deploy after green builds.
CircleCI automatically infers the settings from your code to get your team building faster.
You can also setup a project manually or add a circle.yml file to customize your workflow.
- Scrutinizer :
scrutinizer ci is a hosted continuous inspection service for open-source as well as private code. It runs static code analysis tools, runtime inspectors, and can also run your checks in your favorite language.
- Jenkins :

Jenkins :

Jenkins is one of the most popular tools for doing continuous integration on any platform or technology.

Jenkins will be installed on a server where the central build will take place. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

System Requirements :

JDK	JDK 1.7 or above
Memory	2 GB RAM (recommended)
Disk Space	No minimum requirement. Note that since all builds will be stored on the Jenkins machines, it has to be ensured that sufficient disk space is available for build storage.
Operating System	Jenkins can be installed on Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS

Advantages of Jenkins:

- Its an open source tool with great community support.
- Easy to install and It has a simple configuration through a web-based GUI .
- It has around 900+ plugins to ease your work.
- Jenkins tool is developed in java , It is portable on all platforms.
- If a plugin does not exist, Write custom groovy script to fulfill requirement.

What Jenkins can Do?

- Associate jenkins with a version control server.
- Trigger builds by polling, Periodic etc.
- Execute bash scripts, shell scripts, ANT scripts.

- Publish results and send email notifications.
- Customize Jenkins using groovy script.
- Execute Builds on multiple servers using master slave.

Standard installation steps :

- `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`

Note : By default, Jenkins listen on port 8080. Access this port with your browser to start configuration.

Create Job on Jenkins :

Step 1 : Before Start configuration install ant , composer , phpunit, phpcpd, phpmd, phpcs, Xdebug as per requirements.

Step 2 : Create build.xml (ant script) and save it on application root directory.

```

<?xml version="1.0" encoding="UTF-8"?>
<project name="TestProject" default="build">
  <property name="workspace" value="${basedir}" />
  <property name="sourcedir" value="${basedir}/src" />
  <property name="builddir" value="${workspace}/app/build" />
  <target name="build" depends="prepare,vendors,parameters,lint,phpcpd,phpunit"/>
  <target name="build-parallel" depends="prepare,lint,tools-parallel,phpunit"/>
  <target name="tools-parallel" description="Run tools in parallel">
    <parallel threadCount="2">
      <antcall target="phpcpd"/>
    </parallel>
  </target>
  <target name="clean" description="Cleanup build artifacts">
    <delete dir="${builddir}/api"/>
    <delete dir="${builddir}/code-browser"/>
    <delete dir="${builddir}/coverage"/>
    <delete dir="${builddir}/logs"/>
    <delete dir="${builddir}/pdepend"/>
    <delete dir="${builddir}/docs/**"/>
  </target>
  <target name="prepare" depends="clean" description="Prepare for build">
    <mkdir dir="${builddir}/api"/>
    <mkdir dir="${builddir}/code-browser"/>
    <mkdir dir="${builddir}/coverage"/>
    <mkdir dir="${builddir}/logs"/>
    <mkdir dir="${builddir}/pdepend"/>
  </target>
  <target name="lint" description="Perform syntax check of sourcecode files">
    <apply executable="php" failonerror="true">
      <arg value="-l" />
      <fileset dir="${sourcedir}">
        <include name="**/*.php" />
      </fileset>
      <fileset dir="${basedir}/src/">
        <include name="**/*Test.php" />
      </fileset>
    </apply>
  </target>
  <target name="phpcs" description="Find coding standard violations using PHP_CodeSniffer and print human readable output. Intended for usage on the command line before committing.">
    <exec executable="phpcs">
      <arg value="--standard=Symfony2" />
      <arg path="${sourcedir}" />
    </exec>
  </target>
  <target name="phpcpd" description="Find duplicate code using PHPCPD">
    <exec executable="phpcpd">
      <arg value="--log-pmd" />
      <arg value="${builddir}/logs/pmd-cpd.xml" />
      <arg path="${sourcedir}" />
    </exec>
  </target>
  <target name="phpunit" description="Run unit tests with PHPUnit">
    <exec executable="phpunit" failonerror="true">
      <arg value="-c" />
      <arg path="${basedir}/app/phpunit.xml" />
    </exec>
  </target>
  <target name="vendors" description="Update vendors">
    <exec executable="php" failonerror="true">
      <arg value="composer.phar" />
      <arg value="update" />
    </exec>
  </target>
  <target name="parameters" description="Copy parameters">
    <exec executable="cp" failonerror="true">
      <arg path="app/config/parameters.yml.dist" />
      <arg path="app/config/parameters.yml" />
    </exec>
  </target>
</project>

```

Now create a directory “build” in the application directory , all command output will go inside, we’re gonna create configuration files for PHP libraries. Let’s start with a basic configuration for PHPMD (PHP Mess Detector)

Note : The phpmd.xml and phpunit.xml are optional, PHPMD is used to detect violations from defined standards and PHPUnit is used to execute test cases .

```
<?xml version="1.0"?>
<ruleset name="Symfony2 ruleset" xmlns="http://pmd.sf.net/ruleset/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://pmd.sf.net/ruleset/1.0.0 http://pmd.sf.net/ruleset_xml_schema.xsd" xsi:noNamespaceSchemaLocation="
  http://pmd.sf.net/ruleset_xml_schema.xsd">
  <description>
    Custom ruleset.
  </description>

  <rule ref="rulesets/design.xml" />
  <rule ref="rulesets/unusedcode.xml" />
  <rule ref="rulesets/codesize.xml" />
  <rule ref="rulesets/naming.xml" />

</ruleset>
```

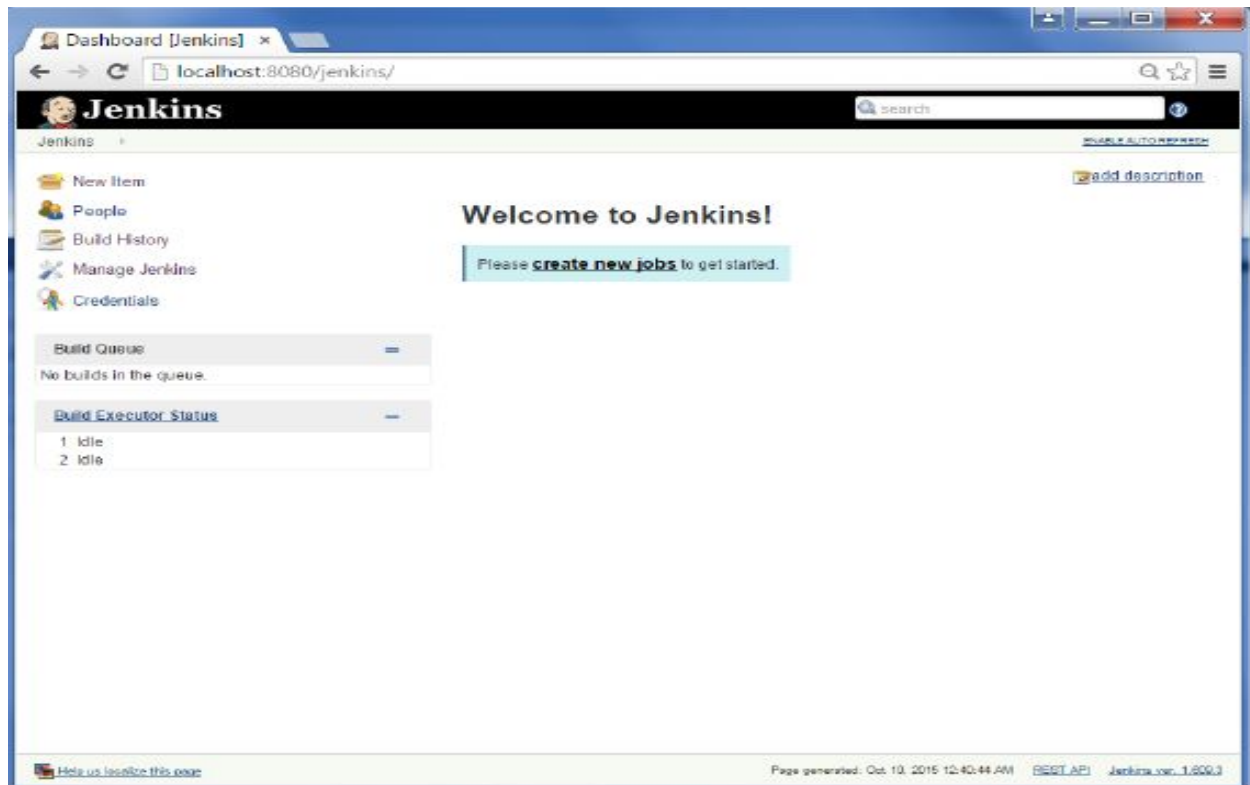
Add the above content to phpmd.xml in root of the project.

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit
  backupGlobals          = "false"
  backupStaticAttributes = "false"
  colors                 = "true"
  convertErrorsToExceptions = "true"
  convertNoticesToExceptions = "true"
  convertWarningsToExceptions = "true"
  processIsolation       = "false"
  stopOnFailure           = "false"
  syntaxCheck             = "false"
  bootstrap               = "bootstrap.php" >

  <testsuites>
    <testsuite name="Project Test Suite">
      <directory>../src/*/*Bundle/Tests</directory>
      <directory>../src/*/*Bundle/*Bundle/Tests</directory>
    </testsuite>
  </testsuites>
  <logging>
    <log type="coverage-html" target="build/coverage" title="TestProject" charset="UTF-8" yui="true" highlight="true"
    lowUpperBound="35" highLowerBound="70"/>
    <log type="coverage-clover" target="build/logs/clover.xml"/>
    <log type="junit" target="build/logs/junit.xml" logIncompleteSkipped="false"/>
  </logging>
  <filter>
    <whitelist>
      <directory>../src</directory>
      <exclude>
        <directory>../src/*/*Bundle/Resources</directory>
        <directory>../src/*/*Bundle/Tests</directory>
        <directory>../src/*/*Bundle/*Bundle/Resources</directory>
        <directory>../src/*/*Bundle/*Bundle/Tests</directory>
      </exclude>
    </whitelist>
  </filter>
</phpunit>
```

Add the above code to phpunit.xml.dist

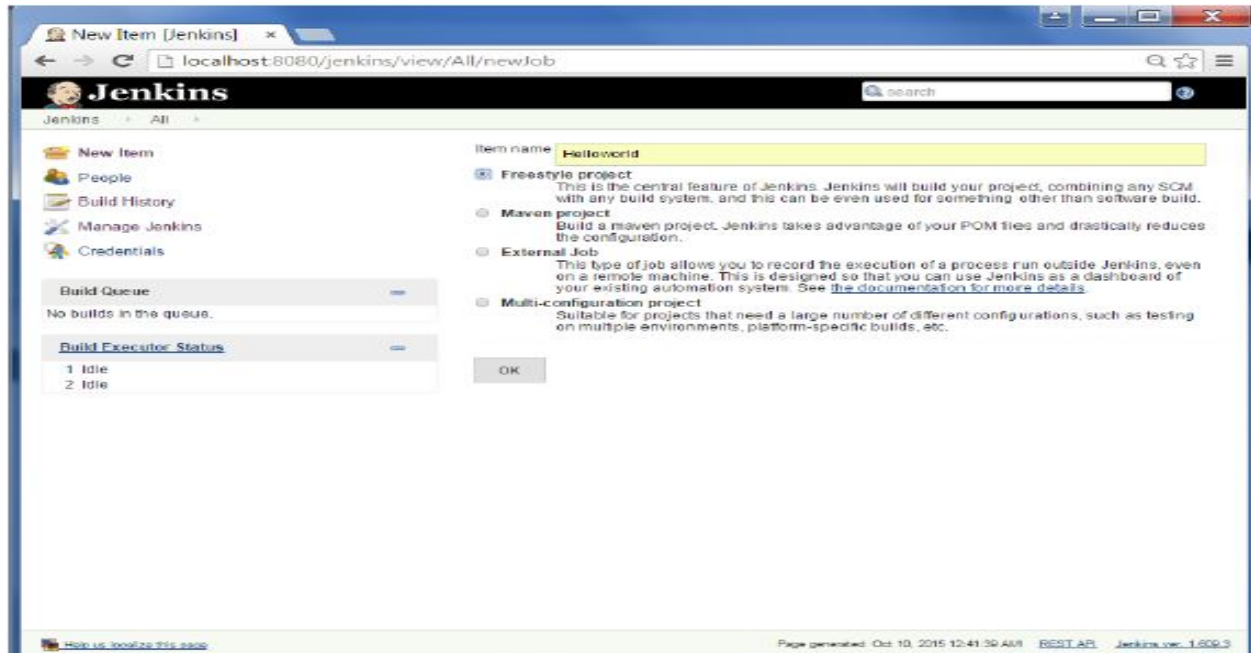
Step 3 : Go to the Jenkins dashboard and Click on New Item.



Step 4 : In the next screen , enter the Item name, in this case we have named it “Helloworld”. Choose the “Freestyle project option”

We can create different types of jobs such as :

- Maven project
- External Job
- Multi-configuration project
- Pipeline



Step 5 : The following screen will come up in which you can specify the details of the job.

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

Description

[Safe HTML]
Preview

☐ Enable project-based security
☐ Configure Dry Run
☐ Discard old builds
☐ GitHub project

GitLab connection

GitLab

GitLab Repository Name

☐ Delivery Pipeline configuration
☐ Sidebar Links
☐ This project is parameterized
☐ Throttle builds
☐ Prepare an environment for the run
☐ Disable this project
☐ Execute concurrent builds if necessary
☐ Restrict where this project can be run

Advanced...

Source Code Management

☒ None
☐ AWS CodePipeline
☐ CVS
☐ CVS Projectset
☐ Git
☐ OpenShift ImageStreams
☐ Subversion

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)
☐ Build after other projects are built
☐ Build periodically
☐ Build when a change is pushed to GitLab. GitLab CI Service URL: http://127.0.0.1:8080/project/Test%20Job
☐ GitHub Branches
☐ GitHub Pull Request Builder
☐ GitHub Pull Requests
☐ GitHub hook trigger for GITScm polling
☐ Gitlab Merge Requests Builder
☐ Poll SCM

Build Environment

☐ Start Xvfb before the build, and shut it down after.
☐ Delete workspace before build starts
☐ Provide Configuration files
☐ Abort the build if it's stuck
☐ Add timestamps to the Console Output
☐ Automatically record and report code coverage using OpenClover. Currently for Ant builds only.
☐ Copy files into the job's workspace before building
☐ Create Delivery Pipeline version
☐ Inject environment variables to the build process
☐ Inject passwords to the build as environment variables
☐ Logentries Forwarder
☐ Run a PHP built-in web server
☐ SSH Agent
☐ Set GitHub commit status with custom context and message (Must configure upstream job using GHPRB trigger)
☐ Setup Kubernetes CLI (kubectl)
☐ Use secret text(s) or file(s)

Save

Apply

In the above screen configure the “General” , “Source Code Management” ,” Build Triggers” ,” Build Environment” , “build” ,” Post Build Actions” sections according to the requirement .

General:

Add general information : name, description (Can use HTML),Project Base Security, disable old builds

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

Project name:

Description:

[Safe HTML] [Preview](#)

☒ Enable project-based security

☐ Block inheritance of global authorization matrix

User/group	Credentials				Job								Run	SCM	Env. Inject			
	Create	Delete	Manage	Domains	Update	View	Build	Cancel	Configure	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

User/group to add:

☐ Configure Dry Run

☒ Discard old builds

Strategy:

Days to keep builds:
if not empty, build records are only kept up to this number of days

Max # of builds to keep:
if not empty, only up to this number of build records are kept

Source Code Management:

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

Repositories

Repository URL:

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

Repository browser

URL:

Additional Behaviours

☐ Subversion

Build Triggers

Build Triggers:

Once it's done, Configure how often to trigger a Build. For instance, Let's set it with a cron to trigger my build friday of each week at 6:30pm. An interesting alternative is to trigger the build when changes are detected in your code repository, for that check the Poll SCM and have a look into the documentation.

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to GitLab. GitLab CI Service URL: <http://127.0.0.1:8080>
- ☐ GitHub Branches
- ☐ GitHub Pull Request Builder
- ☐ GitHub Pull Requests
- ☐ GitHub hook trigger for GITScm polling
- ☐ Gitlab Merge Requests Builder
- ☐ Poll SCM

Build Environment:

Build Environment

- ☐ Start Xvfb before the build, and shut it down after.
- ☒ Delete workspace before build starts

Advanced...

- ☐ Provide Configuration files
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Automatically record and report code coverage using OpenClover. Currently for Ant builds only.
- ☐ Copy files into the job's workspace before building
- ☐ Create Delivery Pipeline version
- ☐ Inject environment variables to the build process
- ☐ Inject passwords to the build as environment variables
- ☐ Logentries Forwarder
- ☐ Run a PHP built-in web server
- ☐ SSH Agent
- ☐ Set GitHub commit status with custom context and message (Must configure upstream job using GHPRB trigger)
- ☐ Setup Kubernetes CLI (kubectl)
- ☐ Use secret text(s) or file(s)

Build:

We are going to use Ant, a tool created by Apache. If you are curious you should have a look to Phing which is a PHP build tool based on Ant.

Build

Invoke Ant

Targets

Advanced...

Post Build Actions:

This is the area where we mentioned about reports that needs to be generated when build is failed or pass. Few reports then can be generated are listed below.

PHP Mess Detector (PMD)

Post-build Actions

Publish PMD analysis results

PMD results

Fileset includes setting that specifies the generated raw PMD XML report files, such as `**/pmd.xml`. Basedir of the fileset is the workspace root. If no value is set, then the default `**/pmd.xml` is used. Be sure not to include any non-report files into this pattern.

Advanced...

Delete

PHP Copy Paste Detector (PCD):

Publish duplicate code analysis results

Duplicate code results

Fileset includes setting that specifies the generated raw XML report files, such as `**/cpd.xml` or `**/simian.xml`. Basedir of the fileset is the workspace root. If no value is set, then the default `**/cpd.xml` is used. Be sure not to include any non-report files into this pattern.

High priority threshold

Minimum number of duplicated lines for high priority warnings.

Normal priority threshold

Minimum number of duplicated lines for normal priority warnings.

Advanced...

Delete

PHPUnit test results:

Publish JUnit test result report

Test report XMLs

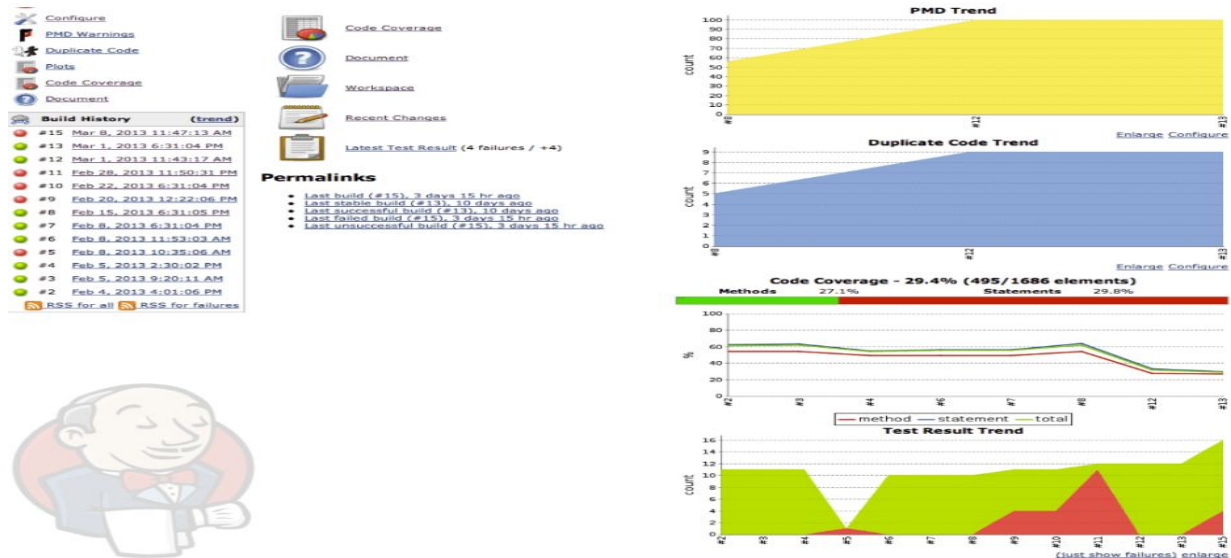
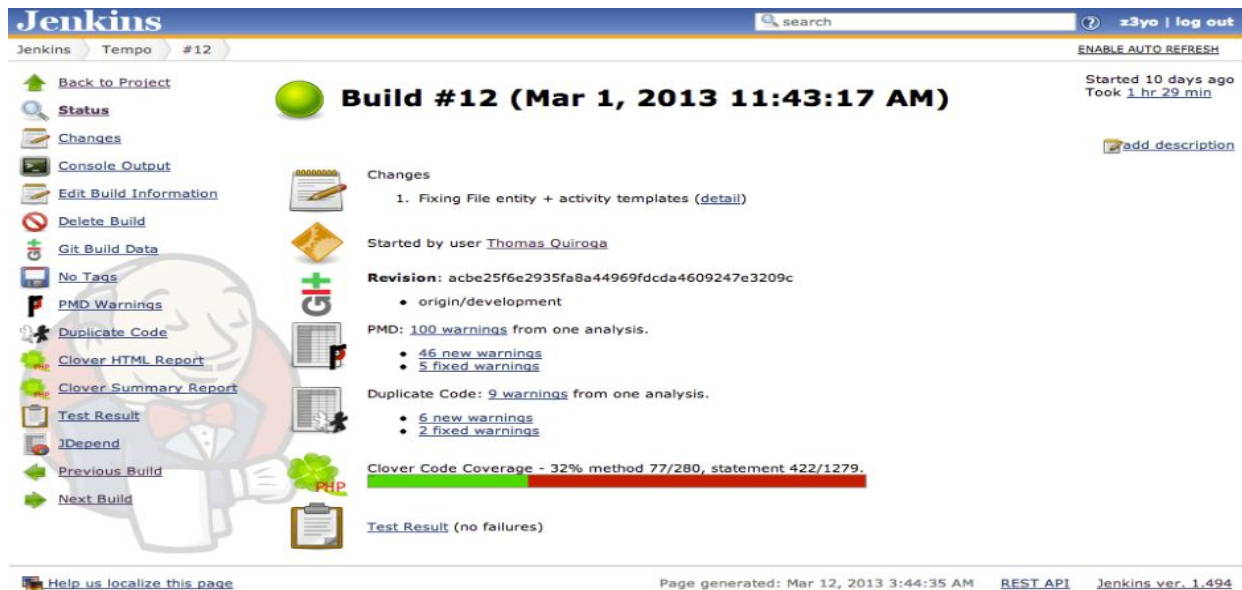
Fileset 'includes' setting that specifies the generated raw XML report files, such as `'myproject/target/test-reports/*.xml'`. Basedir of the fileset is the workspace root.

☐ Retain long standard output/error

Delete

Build results:

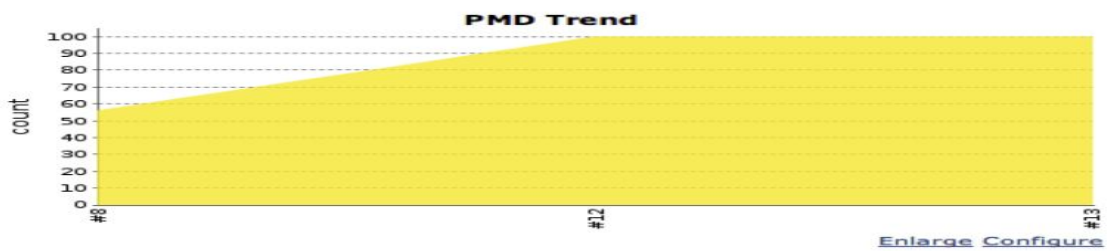
Build result after successful build :



Brief information that we can draw from the above two screenshot.

PMD Warning :

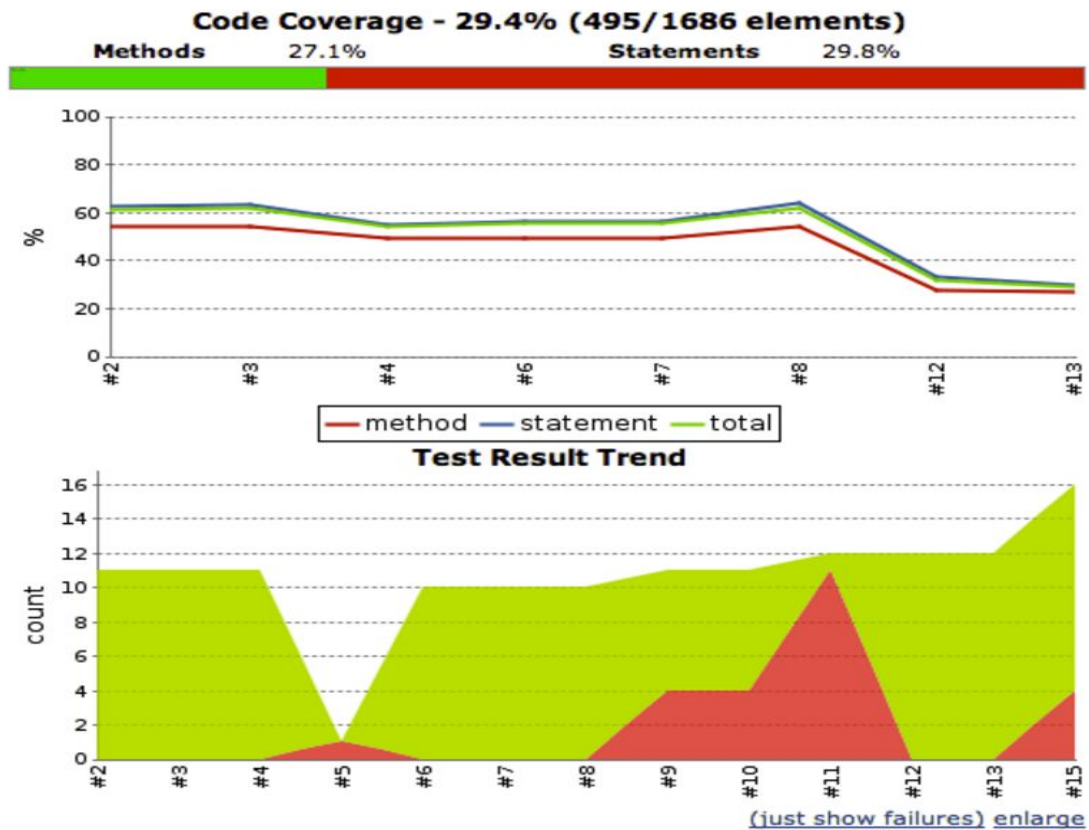
PMD displays a trend which represent the number of warning noticed. If it decreases, that means you have to improve the quality of your code.



- Avoid variables with short names like \$em. Configured minimum length is 3.
- Avoid unused private methods such as 'analyseChange'.
- The class User has 16 fields. Consider to redesign User to keep the number of fields under 15.
- Avoid unused parameters such as '\$options'.

PHPUnit

PHPUnit is generating a report about your unit tests and the code coverage. You will get : number of test passed/failed and the percent of coverage.



A more detailed report of coverage is available in “Clover Code Coverage”, your objective is to get percentage higher as possible.

	Code Coverage					
	Lines		Functions and Methods		Classes and Traits	
Total	<div><div></div></div>	55.07% 407 / 739	<div><div></div></div>	49.65% 70 / 141	<div><div></div></div>	34.29% 12 / 35
CoreBundle	<div><div></div></div>	0.00% 0 / 19	<div><div></div></div>	0.00% 0 / 4	<div><div></div></div>	20.00% 1 / 5
ProjectBundle	<div><div></div></div>	21.61% 59 / 273	<div><div></div></div>	18.67% 10 / 53	<div><div></div></div>	38.46% 5 / 13
UserBundle	<div><div></div></div>	77.85% 348 / 447	<div><div></div></div>	71.43% 80 / 84	<div><div></div></div>	35.29% 6 / 17

Legend

Low: 0% to 35% Medium: 35% to 70% High: 70% to 100%

Generated by PHP_CodeCoverage 1.2.7 using PHP 5.3.3-7+squeeze14 and PHPUnit 3.7.10 at Tue Feb 5 14:39:21 CET 2013.

Manage Jenkins Plugins :

Plugins are the primary means of enhancing the functionality of a Jenkins environment according to the user requirement.

Installing a plugin :

Jenkins provides a couple of different methods for installing plugins on the Jenkins master:

- Using the "Plugin Manager" in the web UI.

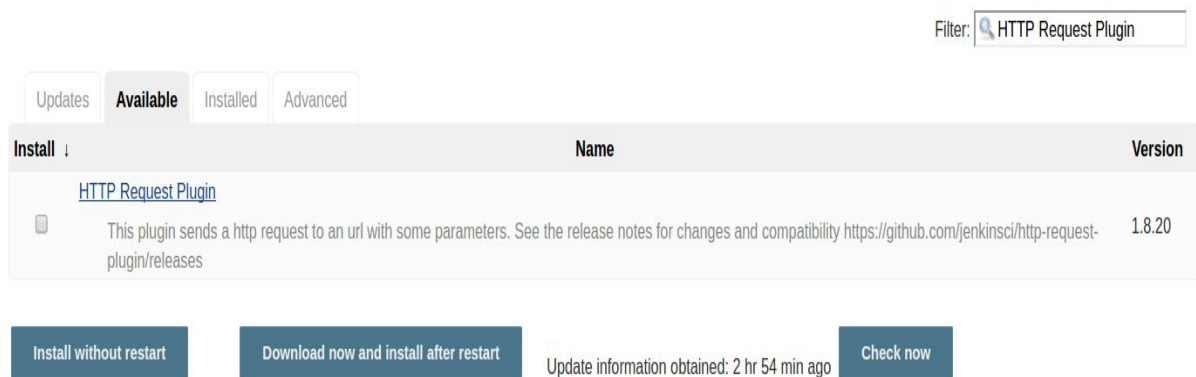
- Using the Jenkins CLI install-plugin command.

The plugins are packaged as self-contained .hpi files, which have all the necessary code, images, and other resources which the plugin needs to operate successfully.

From the web UI :

The simplest and most common way of installing plugins is through the Manage Jenkins > Manage Plugins view, available to administrators of a Jenkins environment.

Under the “Available tab”, plugins available for download from the configured Update Center . We can Filter plugin by plugin name . Select the required plugins from plugins list and click on the “Install without restart” button to install the selected plugins.



The screenshot shows the Jenkins Manage Plugins interface. At the top right, there is a search filter set to 'HTTP Request Plugin'. Below this, there are four tabs: 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A table lists the available plugins. The first entry is the 'HTTP Request Plugin', version 1.8.20. Below the table, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'. A status message indicates 'Update information obtained: 2 hr 54 min ago'.

Install	Name	Version
HTTP Request Plugin	This plugin sends a http request to an url with some parameters. See the release notes for changes and compatibility https://github.com/jenkinsci/http-request-plugin/releases	1.8.20

Buttons: Install without restart, Download now and install after restart, Check now

Update information obtained: 2 hr 54 min ago

Using the Jenkins CLI:

Administrators may also use the “Jenkins CLI” which provides a command to install plugins.

The Jenkins CLI allows a command line user or automation tool to download a plugin and its dependencies.

Syntax : `java -jar jenkins-cli.jar -s http://localhost:8080/ install-plugin SOURCE ... [-deploy] [-name VAL] [-restart]`

Note : For command line download the jenkins-cli.jar file.

Advanced installation :

The Update Center only allows the installation of the most recently released version of a plugin. In cases where an older release of the plugin is desired, a Jenkins administrator can download an older .hpi archive and manually install that on the Jenkins master from the web UI .

Assuming a .hpi file has been downloaded, a logged-in Jenkins administrator may upload the file from within the web UI:

- Navigate to the Manage Jenkins > Manage Plugins page in the web UI.
- Click on the Advanced tab.
- Choose the .hpi file under the Upload Plugin section.
- Upload the plugin file.

We can Download the Jenkins plugins from [here](#) .

List of jenkins plugins is as follows :

Ansible plugin

Authentication Tokens API Plugin

Build-metrics

Clover plugin

Git Parameter Plug-In

Groovy

Mailer Plugin

Pipeline

Robot Framework plugin

Slack Upload Plugin

Slave-status

SSH Credentials Plugin

xUnit plugin

Global Build Stats Plugin

NodeLabel Parameter

Post+build+task

Ant Plugin

Build History Metrics plugin

Clover PHP plugin

Copy To Slave Plugin

GitHub plugin

JUnit Plugin

Metrics Plugin

Project statistics Plugin

Simple Theme Plugin

Slave Setup Plugin

SSH Agent Plugin

SSH Slaves plugin

Xvfb plugin

BUild Monitor

Job Generator Plugin

HTML Publisher

Jenkins Master Slave :

Jenkins uses a Master-Slave architecture to manage distributed builds. We can divide master workload by executing build on slave servers .For more details [click here](#) .

Jenkins Theme Customization :

A Simple Theme plugin for Jenkins supports custom CSS & JavaScript . Using “Simple Theme Plugin” custom theme can be apply to jenkins. Jenkins's appearance can be customized in two ways :

- Using Custom theme.

- Existing Themes with Simple Theme plugin.

Jenkins Theme Customization Steps :

1. Add Simple Theme plugin :

Follow the below steps to install “Simple Theme”.

Installation Steps :

- Go to the Jenkins Site and Login with admin credential.
- Go to the Manage Jenkins -> Manage Plugins -> Available plugins and Install Simple theme Plugin.

1. Customize Theme :

Jenkins has a mechanism known as "UserContent", where administrators can place files inside \$JENKINS_HOME/userContent, and these files are served from <http://yourhost/jenkins/userContent> This can be thought of as a mini HTTP server to serve images, stylesheets, and other static resources that can use from various description fields inside Jenkins

Note : Add custom theme in “userContent” folder. (/var/lib/jenkins/userContent/)

Custom theme :

- To create custom theme, create folder inside “userContent” directory of jenkins setup (i.e. /var/lib/jenkins/userContent/) .
- After creating folder add images , css and JavaScript code in theme folder .
- Set css and JavaScript file path At Manage Jenkins > Configure System > Theme location

Theme	
URL of theme CSS	<input type="text" value="http://localhost:8080/userContent/layout/style.css"/>
	<small>Specify URL of a theme CSS.</small>
URL of theme JS	<input type="text" value="http://localhost:8080/userContent/layout/main.js"/>
	<small>Specify URL of a theme JS.</small>

Existing themes :

There are some existing themes available with simple theme plugins . Download these themes and add in theme folder in “userContent” directory (`var/lib/jenkins/userContent/`) . Set `css` and `js` path in Manage Jenkins > Configure System > Theme,

Existing themes with Jenkins :

- Jenkins Atlassian Theme
- Jenkins-Material-Theme
- Jenkins-Neo-Theme
- Doony
- Rackspace Canon
- Improved" Canon Themes
- Isotope Eleven
- Clean Theme

Jenkins Backup & Restore :

Periodic backup of jenkins is necessary. All jenkins configurations, jobs, plugins, build logs are stored under jenkins home directory. Simply archive this directory to make a backup. Similarly, restoring the data is just replacing the contents of the JENKINS_HOME directory from a back up.

Ways to backup and restore jenkins jobs and settings

1. Manual backup and restore

Steps for manual backup:

- a. Stop jenkins server :

Execute:

```
$ sudo service jenkins stop
```

- b. Archive jenkins home directory to make a full backup.

Jenkins home directory is like (`/var/lib/jenkins`)

Execute command :

```
$ sudo tar -zcvf locaion_to_save_backup/jenkins_backup.tar.gz jenkins_home_location
```

- c. Unarchive backup which made earlier.

Execute:

```
$ tar -xvzf jenkins_backup.tar.gz
```

- d. Copy unarchived directory to jenkins home.

Execute:

```
$ cp -R unarchived_tar/* jenkins_home/
```

- e. Change owner and group permission to jenkins:jenkins.

Execute:

```
$ sudo chown -R jenkins:jenkins jenkins_home/
```

- f. Start jenkins server.

Execute:

```
$ sudo service jenkins start
```

- g. Verify all backups restored or not.

Limitations:

1. Manual backup is not convenient, every time taking backup manually is not good.

Backup process should be automated.

2. Backing up whole directory is not required like workspace, logs need not to be backed up.

1. Backup and restore using thinBackup plugin

Steps for backup and restore using thinBackup plugin

1. Install thinBackup plugin.
2. Go to Manage Jenkins -> ThinBackup

The screenshot shows the Jenkins ThinBackup plugin interface. At the top, there is a breadcrumb navigation bar with 'Jenkins' and 'ThinBackup'. On the left side, there is a sidebar menu with the following items: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. Below the menu, there are two expandable sections: 'Build Queue' and 'Build Executor Status'. The 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows two executors, both in an 'Idle' state. On the right side of the interface, there is a large heading 'ThinBackup' with a folder icon. Below the heading, there are three main action buttons: 'Backup Now' (with a blue arrow icon), 'Restore' (with a blue arrow icon), and 'Settings' (with a wrench icon).

Click on Settings

Make settings as shown in below screen.

Backup settings

Backup directory	<input type="text" value="/var/lib/jenkins/jenkinsbackup"/> <small>⚠ The directory does not exist, but will be created before the first run.</small>
Backup schedule for full backups	<input type="text" value="0 2 * * *"/> <small>⚠ Cron schedule warning: Spread load evenly by using 'H 2 * * *' rather than '0 2 * * *'</small>
Backup schedule for differential backups	<input type="text"/>
Max number of backup sets	<input type="text" value="3"/>
Files excluded from backup (regular expression)	<input type="text"/>
<input checked="" type="checkbox"/> Wait until Jenkins/Hudson is idle to perform a backup	
Force Jenkins to quiet mode after specified minutes	<input type="text" value="120"/>
<input checked="" type="checkbox"/> Backup build results	
<input type="checkbox"/> Backup build archive	
<input type="checkbox"/> Backup only builds marked to keep	
<input checked="" type="checkbox"/> Backup 'userContent' folder	
<input checked="" type="checkbox"/> Backup next build number file	
<input checked="" type="checkbox"/> Backup plugins archives	
<input checked="" type="checkbox"/> Backup additional files	
Files included in backup (regular expression)	<input type="text" value="(*.*)"/>
<input type="checkbox"/> Clean up differential backups	
<input checked="" type="checkbox"/> Move old backups to ZIP files	

- Backup directory:**
Specify path where you want to save backup, it will automatically create directory if it not exists.
- Backup schedule for full backups:**
Specify schedule to execute backup in cron syntax format.
- Backup schedule for differential backup:**
Specify schedule to execute differential backup in cron syntax format. This backup stores only those files which are modified after full backup. For differential backup at least one full backup is needed. If full backup is not available it will get create at first execution.

- a. Max number of backup sets:

Specifies the maximum number of backup sets which you want to store. Older backup exceeding this count will automatically gets deleted.

- a. Files excluded from backup (regular expression):

If you want to exclude some files from backup you can specify those with regular expression. All files with a name matching this regular expression will not be backed up. Leave empty if not needed.

- a. Wait until Jenkins/Hudson is idle to perform a backup:

It is always better to check this option. If we disable this option, it may cause problems which results in corrupted backups.

- a. Force Jenkins to quiet mode after specified minutes:

Backup is created at scheduled period, to make sure that nothing will change when backup is executing switch Jenkins to “Quiet Mode “ where no new jobs in the queue will be handled. Specify value in minutes to switch jenkins to quiet mode.

- a. Backup build results:

If this option is enabled, build results will also be backed up.

- b. Backup build archive:

If this options is enabled, the build archive within the build results will also be backup up. Try to put this option disabled it will take disk space.

- a. Backup only builds marked to keep:

If this option is enabled it will backup only those build which are marked as “Keep this build forever”

- a. Backup 'userContent' folder:

It will backup data from userContent folder.

- b. Backup next build number file:

It will backup nextBuildNumber file.

- c. Backup plugins archives:

It will backup plugins.

- d. Backup additional files:

If you want to backup additional files form jenkins home you can specifies this files with regular expression.This only allows filename patterns.

- e. Clean up differential backups:

It will remove all differential backup whenever full backup is done.

- f. Move old backups to ZIP files

It will move all old backups to ZIP whenever new full backup is created.

Limitations

1. Including additional folder for backup is difficult, because "Backup additional files" allows only filename patterns.
2. If regular expression written in "Backup additional files" and "Files excluded from backup" is invalid it will be disregarded.

1. Backup and restore using shell script

Steps to backup and restore using shell script:

- a. Download jenkins_backup.sh script from <https://github.com/sue445/jenkins-backup-script>
- b. Give it executable permission.

```
$ sudo chmod +x jenkins_backup.sh
```
- c. Execute below command to take backup

```
$ sudo ./jenkins-backup.sh /path_to_jenkins_home /path_to_backup_location
```

```
$ sudo ./jenkins-backup.sh /var/lib/jenkins /home/neelam/Desktop/backtemp/
```
- d. Stop jenkins server
Execute:

```
$ sudo service jenkins stop
```
- e. Unarchive backup which made earlier.
Execute:

```
$ tar -xvzf jenkins_backup.tar.gz
```
- f. Copy unarchived directory to jenkins home.
Execute:

```
$ cp -R unarchived_tar/* jenkins_home/
```
- g. Change owner and group permission to jenkins:jenkins.
Execute:

```
$ sudo chown -R jenkins:jenkins jenkins_home/
```
- h. Start jenkins server.
Execute:

```
$ sudo service jenkins start
```

shell script can be modified to exclude or include files and folder from backup. Shell script can

be executed through cron for periodic backup.

Steps to setup shell script in cron

1. Login with super user in terminal.
2. Open cron file by entering command

```
$ crontab -e
```

3. Add shell script execution command with schedule and save file.

Syntax for cron is:

A crontab file has five fields for specifying day , date and time followed by the command to be run at that interval.

```
* * * * * command to execute.
```

Each start specifies:

* (Min 0-59)

* (Hour 0-23)

* (day of month 1-31)

* (month 1-12)

* (weekday 0-6 Sunday = 0)

Example:

```
15 15 * * * /home/jenkins-backup.sh /var/lib/jenkins /home/backtemp/
```

Above command will execute shell script everyday at 3:15 PM.

Upgrade Jenkins :

- Login to jenkins.
- Go to manage jenkins. This page will display available updates.
- Download update, jenkins.war download.
- Stop jenkins server

```
$ sudo service jenkins stop
```

- Locate existing jenkins.war file.

This file is usually located at location /usr/share/jenkins

If file is not present at this location go to Manage Jenkins -> System Information.

it will display path to .war file under executable-war.

- Go to jenkins.war file location and execute below commands

Backup old jenkins war

```
$ mv jenkins.war /home/Backup/jenkins_backup_war
```

Move new jenkins war to executable-jar location

```
$ mv /home/Downloads/jenkins.war /usr/share/jenkins/
```

- Start jenkins server

```
$ sudo service jenkins start
```

- Go to Manage Jenkins -> Manage Plugins update plugins if update is available.
- Verify all jobs setup.

Jenkins CLI :

You can access various features in Jenkins through a command-line tool using the JAR File "jenkins-cli.jar"

Jenkins has a built-in command line interface that allows you to access Jenkins from a script or from your shell.

Download the JAR file for the client from the URL "/jnlpJars/jenkins-cli.jar" on your Jenkins server, e.g.
<https://jenkins.example.com/jnlpJars/jenkins-cli.jar>.

Basic syntax : `java -jar jenkins-cli.jar [-s JENKINS_URL] command [options...] [arguments...]`

Example : `java -jar jenkins-cli.jar -s http://127.0.0.1:8080/ help`

Note : To check Available command with jenkins cli from "/cli" url for example "http://127.0.0.1:8080/cli/"

Jenkins Security :

As of Jenkins 2.0, many of the security options were enabled by default to ensure that Jenkins environments remained secure unless an administrator explicitly disabled certain protections.

Project-based Matrix Authorization Strategy :

Project-based Matrix Authorization Strategy is one of the authorization strategies available for securing Jenkins.

[Click here](#) to see steps to configure Project-based Matrix Authorization Strategy.

Build Metrics Report :

The build metrics Report display the success and failure ratio of jenkins jobs.

To generate the Build Metrics Report need to install Build Metrics Plugin and Global Build Stats Plugin.

Build Metrics Plugin uses the results from the Global Build Stats Plugin to generate some basic build metrics.

To generate Build Metrics Report :

Step 1 : Go to Manage Jenkins -> Build Metrics after clicking the "build metrics" the "Built Metrics Search screen" will be appear .



Build Metrics Plugin

Search criteria

Label for Search Results :

Show builds for the last :

Filters :

- Job filtering : ☒ ALL Jobs ☐ Job name regex :
- Node filtering : ☒ ALL Nodes ☐ Master only : ☐ Node name regex :
- Launcher filtering : ☒ ALL Users ☐ System only : ☐ Username regex :
- Cause filtering : ☒ ALL Causes ☐ Cause regex :

Configure the following parameters:

- Search Results Label: set this to whatever text you wish to be displayed at the top of the search results screen.
- Select the time window: default is 2 weeks (must be an integer)
- Configure the filters

Step2 : After clicking on the search button, Build Metrics Report Screen will appear.

Search Results

Build Metrics:

Job name	# Successful	# Failed	# Unstable	# Aborted	# Not Built	# Total Builds	Failure Rate
Library Ideas-crm	1	0	0	0	0	1	0.00%
Simple test job	0	1	0	0	0	1	100.00%
Total:	1	1	0	0	0	2	50.00%

Failed Builds:

Status	Job name	Build #	Date	Duration	Node	User	Cause
Failures	Simple test job	#17	2017-06-26 02:40:25 PM	72791	sample test	SYSTEM	

The build metrics report screen is divided into two section. The top section contains the selected jobs and their associated build metrics. The bottom section contains a list of failed builds with the job name.

Troubleshooting:

1. Java version: Jenkins 2.54 and higher requires Java 8.
2. Jenkins out of memory: Update settings in below files
 - a. On ubuntu /etc/default/jenkins -> update value of JAVA_ARGS to -Djava.awt.headless=true
-Dhudson.ClassicPluginStrategy.noBytecodeTransformer=true -Xmx1048m
-XX:MaxPermSize=512m
 - b. On RedHat /etc/sysconfig/jenkins -> update value of JENKINS_JAVA_OPTIONS to
Djava.awt.headless=true -Xmx2048m
3. MaxPermSize=512m is not required in Java 8

Configuration management:

Configuration management is a process which is used to systematically handle changes into a system in such a way that it maintains integrity over time. It plays important role in server configuration management. Below are the widely used open source configuration management tools:

1. Puppet: It is based on client server module. In this architecture, managed nodes run the Puppet agent application, usually as a background service. One or more servers run the Puppet master application, usually in the form of Puppet Server.
2. Chef: It is also based on client server architecture. Machine setup is described in Chef recipes .There is a Chef server which stores each of these recipes and are sent to tell the Chef node how to configure itself.
3. Ansible: Ansible doesn't have a client server module where a server is a master. Ansible is more of a standalone solution where a computer manages other devices without those other devices having to be "Ansible aware".

Introduction To Ansible :

Ansible is general purpose automation tool which is used for configuration management or workflow automation. With Ansible multiple machines can be configured at a time with less time. In comparison to other tools such as Chef or Puppet, Ansible is agent-less it communicates to nodes using SSH.

For more details [click here](#)

Virtualbox :

Introduction :

VirtualBox is a cross-platform virtualization application. It installs on your existing computer (and operating system) and extends the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. VirtualBox allows you to run more than one operating system at a time. By using a VirtualBox feature called “snapshots”, you can save a particular state of a virtual machine and revert back to that state, if necessary. This way, you can freely experiment with a computing environment. If something goes wrong, you can easily switch back to a previous snapshot and avoid the need of frequent backups and restores.

Why is virtualization useful?

- Running multiple operating systems simultaneously.
- Easier software installations.
- Infrastructure consolidation.

Install Virtualbox :

Step 1 : Setup Apt Repository :

Firstly edit `/etc/apt/sources.list` file and add following line :

```
deb http://download.virtualbox.org/virtualbox/debian trusty contrib
```

Step 2 : Setup Oracle public key

After adding required apt repository in your system, download and import the Oracle public key for apt-secure using following commands.

```
$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -
```

```
$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add -
```

Step 3 : Install Oracle VirtualBox

After completing above steps, let's install VirtualBox using following commands. If you have already installed any older version of VirtualBox, Below command will update it automatically.

```
$ sudo apt-get update
```

```
$ sudo apt-get install virtualbox-5.1
```

Step 4 : Start VirtualBox :

We can use dashboard shortcuts to start VirtualBox or simply run following command from a terminal.

```
$ virtualbox
```

Vagrant :

What is Vagrant?

- It Create and configure lightweight, reproducible, and portable development environments
- Vagrant is one of the tool for building and managing the virtual machine environment.
- Vagrant allows you to easily manage and control multiple virtual machines.
- Vagrant is freely available.
- It automates virtual machine creation using Oracle's Virtualbox
- The package/product is written in Ruby
- Avoiding all confusions, it is not a Virtual Machine solution, like Virtualbox, VMware
- It is just a helper for automated solution to interact easily with VirtualBox.
- It sets-up a new VM based on the preconfigured Box that is specified on the Vagrantfile.
- We can create environment using Ansible or Puppet to perform some jobs
- The best part is, once the job is done, the environment can be destroyed or removed

Why Vagrant and What are the Advantages of using it?

- Reduces the Setup time
- Very simple and straightforward
- Repeatability (A new VM can be created and crushed in few minutes)
- Helps create a production-like VM in minutes

Installation Steps :

1. Install VirtualBox :
`sudo apt-get install virtualbox`
2. Install Vagrant :
`sudo apt-get install vagrant`
3. Add Vagrant Box :
`vagrant box add ubuntu/trusty64 (box to install ubuntu)`
4. Start Vagrant :
`vagrant init ubuntu/trusty64`

After executing the above command. A 'Vagrantfile' has been placed in the current directory.

The generated 'Vagrantfile' is a Ruby file that controls your [one or more] virtual machines.

5. Download box and Dependencies :
`vagrant up`

once we execute the above command, Vagrant will be installed and fully up and running on your Ubuntu 14.04 machine.

6. Login to the vagrant box :
`vagrant ssh`
Above command is used to login vagrant box using ssh.

Docker :

Introduction :

Docker is an open source project for developing, shipping, and running applications. With Docker, you can manage your infrastructure in the same ways you manage your applications. Docker provides the facility to run an application in the isolated environment which is called container. You can run many containers simultaneously on a given host. It is lightweight, so starts instantly and uses less RAM.

Docker Features :

Docker provides lots of features, we are listing some major features which are given below.

- Easy and Faster Configuration :

This is a key feature of docker that helps us to configure the system easily and faster.

We can deploy our code in less time and effort. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

- Increase productivity :

By easing technical configuration and rapid deployment of application. No doubt it has increase productivity. Docker not only helps to execute the application in isolated environment but also it has reduced the resources.

- Application Isolation :

It provides containers that are used to run applications in isolation environment. Each container is independent to another and allows us to execute any kind of application.

Docker Architecture :

Docker follows client-server architecture. Its architecture consists mainly three parts.

1. Client: Docker provides Command Line Interface (CLI) tools to client to interact with Docker daemon. Client can build, run and stop application. Client can also interact to Docker_Host remotely.
2. Docker_Host: It contains Containers, Images, and Docker daemon. It provides complete environment to execute and run your application.
3. Registry: It is global repository of images. You can access and use these images to run your application in Docker environment.

The Docker daemon :

It is a process which is used to listen for Docker API requests. It also manages Docker objects like: images, container, network etc. A daemon can also communicate with other daemons to manage Docker services.

Docker Registries :

Docker registry is used to store Docker images. Docker provides the Docker Hub and the Docker Cloud which are public registries that anyone can use. Docker is configured to look for images on Docker Hub by default.

When we use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.

Docker Images :

Docker image is a read-only template with instructions for creating a Docker container.

A docker image is described in text file called a Dockerfile, which has a simple, well-defined syntax. An image does not have states and never changes.

Docker Container :

Docker container is a running instance of an image. We can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. We can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. We can also provide configuration for the network and environment variables.

Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

Docker Installation :

We can install docker on any operating system whether it is Mac, Windows, Linux. Docker Engine runs natively on Linux distributions. we are providing step by step process to install docker engine for Ubuntu 14

Prerequisites:

Docker need two important installation requirements:

- It only works on a 64-bit Linux installation.
- It requires Linux kernel version 3.10 or higher.

To check your current kernel version, open a terminal and type uname -r command to display your kernel version:

Installation steps:

1) Remove older version

```
sudo apt-get remove docker docker-engine
```

2) Update package information

```
sudo apt-get update
```

3) Install Recommended Extra packages

```
sudo apt-get install \ linux-image-extra-$(uname -r) \ linux-image-extra-virtual
```

4) Install packages to allow apt to use a repository over HTTPS:

```
sudo apt-get install \ apt-transport-https \ ca-certificates \ curl \ software-properties-common
```

5) Add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

6) Use the following command to set up the stable repository.

```
sudo add-apt-repository \ "deb [arch=amd64] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) \ stable"
```

7) Update the apt package index.

```
sudo apt-get update
```

8) Install the latest version of Docker

```
sudo apt-get install docker-ce
```

GitLab :

GitLab do almost everything that GitHub does.

GitLab Community Edition is free and open sourced. That in itself gives it a huge boost when compared to GitHub Enterprise Edition.

With GitLab community edition you might be comfortable until the end of time or you might start with the enterprise edition right away.

Basic Difference of GitHub and GitLab :

Features	GitLab	GitHub
Pricing	Unlimited public and private repositories / unlimited public and private collaborators	Free for public repositories / Paid plans for private repositories
Code review features	yes	yes
Bug & issue tracking	yes	yes
Private branch	yes	Yes (with paid plans)
Build system	yes	Yes (with 3rd party service)

Self-hosting	Yes	Yes (with enterprise plan)
detailed pricing	Free: gitlab.com / Free: GitLab Community Edition / \$39 / Year: GitLab Enterprise	Free: public projects / \$7/month: Personal plan / \$25/month: organization plan / \$2.500/year: Enterprise

We can maintain the jenkins metadata using gitlab . Create repository for Jenkins.

This repo contains meta files(like build.xml, deploy.php) required by jenkins to execute Trex jobs. First step in every trex job is to clone/pull this repo.

How to pull Jenkins Jobs metadata from GitLab:

Create one shell script to clone or pull the the GitLab Repository and save this shell script on jenkins root directory

.

1. On Configure job first add credentials required to pull or clone repo. Follow below steps:
 - a. Configure Job -> Build Environment -> Check option (Use secret text(s) or file(s)) -> click on Add and select (Username and password separated)
 - b. Add Username variable as GIT_USER
 - c. Add Password variable as GIT_PASS
 - d. And select credentials
2. Select Execute shell option from “Build” and execute script written in first step.
 - a. Build -> Execute Shell
 - b. Put Command `$ {JENKINS_HOME}/jenkins_metada_pull.sh DIR_NAME_TO_CLONE
http://${GIT_USER}:${GIT_PASS}@infobeans.gitlab.com/infobeans/jenkins_gitlab_metadata.git`
3. Select Inject environment variables from Build and do below configuration.
 - a. Build -> Inject environment variables -> Properties Content (Add variables to use GitLab configurations in the jenkins job)

Repository Structure of Jenkins Job Metadata on Gitlab :

└── DevJob

- └── Build (This folder contains build related files like build.xml)
- └── Deployer (It will contain deployment related files like deploy.php)
- └── Functional (This folders contains functional test cases and all configuration related to it)
- └── QaJob
- └── Build (This folder contains build related files like build.xml)
- └── Deployer (It will contain deployment related files like deploy.php)
- └── Functional(This folders contains functional test cases and all configuration related to it)

Deployer :

What is deployer ?

Deployer is a simple deployment tool for PHP. By installing Deployer and setting up some simple configuration, you can quickly and easily deploy your applications to either staging or production environments with typically only one short command.

Standard installation steps :

```
curl -LO https://deployer.org/deployer.phar
```

```
mv deployer.phar /usr/local/bin/dep
```

```
chmod +x /usr/local/bin/dep
```

Benefits of deployer :

- Deployer supports multiple server deployment.
- Deployer supports rollbacks
- Deployer supports atomic deploys
- Deployer supports parallel tasks
- Deployer supports deployment consistency

Using Deployer:

Deployer uses a regular PHP file for its configuration and deployment tasks. Named it as 'deploy.php' and save this file on your application root directory. You can create a configuration file (deploy.php) by running Following command:

```
$ dep init
```

Your deployment procedure will typically consist of one or more tasks. Deployer makes it really easy to define your own tasks within deploy.php, for example:

```
task('helloworld', function () {  
    writeln('Hello world!');  
})->desc('Says hello to the world!');
```

The task 'helloworld' will now be available via the dep command when run in your project root:

```
dep helloworld
```

Setting up Servers :

You can define your application servers in two ways: either by using the 'server' function within your deploy.php file, or by using a YAML file. Here's an example of the server function:

```
server('dev', 'Server IP')
  ->user('server user')
  ->password('server password')
  ->stage('development')
  ->env('branch', 'dev')
  ->env('deploy_path', '/var/www/html/testDevDeployment');
```

For multiple server configuration:

```
server('dev', 'Server IP')
  ->user('server user')
  ->password('server password')
  ->stage('development')
  ->env('branch', 'dev')
  ->env('deploy_path', '/var/www/html/testDevDeployment');

server('qa', 'Server IP')
  ->user('server user')
  ->password('server password')
  ->stage('development')
  ->env('branch', 'qa')
  ->env('deploy_path', '/var/www/html/testDevDeployment');
```

Alternatively, you can use a YAML file to define your servers:

```
production:
  host: myapp.example.com
  user: server user
  identity_file: ~
  stage: 'development'
  deploy_path: /var/www/html/testDevDeployment
```

YAML for multiple server :

```
dev:
  host: myapp.example.com
  user: server user
  identity_file: ~
  stage: 'development'
  deploy_path: /var/www/html/testDevDeployment

production:
  host: myapp.example.com
  user: server user
  identity_file: ~
  stage: 'development'
  deploy_path: /var/www/html/testDevDeployment
```

You can load the YAML server configuration in your deploy.php file by using `serverList('servers.yml');`

Defining your Repository :

Specify the repository from which to download your project's code. Deployer needs to know where it can clone your project's code from. You can do this by using the `set` function to define a configuration parameter called 'repository':

```
set('repository', 'git@domain.com:username/repository.git');
```

The Deploy Task in deployer :

deploy.php contains one of the main task name as 'deploy' for example:


```
task('deploy', [
  'deploy:prepare',
  'deploy:release',
  'deploy:update_code',
  'deploy:create_cache_dir',
  'deploy:assets',
  'deploy:vendors',
  'deploy:assetic:dump',|
  'deploy:symlink',
  'database:migrate',
  'cleanup',
])->desc('Deploy your project');
```

In the above deploy task :

```
deploy:prepare = Preparing server for deploy
deploy:release = Prepare release
deploy:update_code = Updating code
deploy:create_cache_dir = Create cache dir
deploy:assets = Normalize asset timestamps
deploy:vendors = Installing vendors
deploy:assetic:dump = Dump assets
deploy:symlink = Creating symlink to release
database:migrate = Migrate database
cleanup = Cleaning up old releases|
```

Note : type dep or dep list command you will see list of all available tasks.

After configuration execute deployer script using following commands :

```
$ dep deploy dev
```

If something goes wrong, you can always roll back to your previous release by running following command :

\$ dep rollback.

Sample Deployer File :

```
<?php

server('dev', 'Server IP')
    ->user('server user name')
    ->password('server user password')
    ->stage('development')
    ->env('branch', 'dev')
    ->env('deploy_path', '/var/www/html/testDevDeployment');

set('repository', 'git@domain.com:username/repository.git');
env('env_vars', 'SYMFONY_ENV=dev');
env('env', 'dev');

task('deploy', [
    'deploy:prepare',
    'deploy:release',
    'deploy:update_code',
    'deploy:create_cache_dir',
    'deploy:assets',
    'deploy:vendors',
    'deploy:assetic:dump',
    'deploy:symlink',
    'database:migrate',
    'cleanup',
])->desc('Deploy your project');
after('deploy', 'success');
task('upload:parameters', function () {
    upload('/var/lib/jenkins/deployer/testDevDeployment/parameters.yml', '{{release_path}}/app/config/parameters.yml');
    write('Upload parameters.yml done!');
});
after('deploy:create_cache_dir', 'upload:parameters');

task('deploy:assetic:dump', function () {
    if (!get('dump_assets')) {
        return;
    }
    run('{{bin/php}} {{release_path}}/' . trim(get('bin_dir'), '/') . '/console assetic:dump');
})->desc('Dump assets');
/**
 * Copy app_dev.php for dev environment
 */
task('data:copy_app_dev', function () {
    upload('/var/lib/jenkins/deployer/testDevDeployment/app_dev.php', '{{release_path}}/web/app_dev.php');
    write('Upload app_dev.php done!');
});
after('cleanup', 'data:copy_app_dev');
```

SonarLint for Command Line :

We can configure the sonar lint with IDE such as : eclipse ,intellij ,visual studio .

SonarLint for Command Line provides pre-commit feedback to developers on new bugs .

SonarLint offers the ability to scan code for issues before checking it in, with minimal configuration.

SonarLint for Command Line is free and open source. The connected mode is compatible with SonarQube 5.6+.

Using sonarlint we can generate HTML report.

Installation Steps:

- Download the updated sonar lint from [here](#) .
- Add <extraction_path>/bin to your PATH.

```
export PATH="<extraction_path>/bin:$PATH"
```

Or

Edit .bashrc in your home directory and add the following line:

```
export PATH="/path/to/dir:$PATH"
```

after that execute the following command :

```
source ~/.bashrc
```

Note : To use sonar lint on developer machine, Instal java 8 .

Configuration For SonarLint :

Global configuration :

Create "global.json" for sonarqube server details on {home}/.sonarlint/conf/global.json with below content :

```
{
  servers: [
    {
      "id": "local",
      "url": "http://localhost:9000",
      "token": "mytoken"
    }
  ]
}
```

Several server configurations should be defined. It's also possible to define a login and password instead of a token.

Per-project configuration :

Each project can be bound to a specific remote project in a SonarQube server defined in the global configuration file. Create sonarlint.json file on application root directory. The project key and the server id should be defined in the sonarlint.json .

sample of sonarlint.json :

```
{  
  "serverId": "local",  
  "projectKey": "my.organization:project1"  
}
```

use following command to Update binding with SonarQube server before analysis :

```
sonarlint -u
```

Note : To check all possible options with sonarlint , use “sonarlint -h”

Sonarqube:

What is Sonarqube?

Sonarqube is continuous inspection application that can help us automate code inspection.

It's a free and open source application developed and maintained by Sonarsource, previously SonarQube name is only Sonar. SonarQube can analyze source code files and related binaries, calculate a set of metrics and show the result on web based dashboard.

SonarQube platform is divided into 4 parts :

1. SonarQube Server :

2. SonarQube Database :
Store configuration of the SonarQube instance and snapshots of projects, views
3. SonarQube Plugins :
Plugins are installed on server. It includes integration, authentication, and governance plugins
4. SonarQube Scanners :
Running on Continuous Integration Servers to analyze projects

Hardware Requirement :

Memory	2 GB
disk space	depend on how much code you analyze with SonarQube
Java	8
Database	Microsoft SQL Server : 2012 ,2014 MySQL : 5.6,5.7 Oracle : 11G, 12C,XE Editions are supported PostgreSQL : 8.x , 9.x
Web Browser	Microsoft Internet Explorer : IE 11 Mozilla Firefox : latest Google Chrome : latest Safari : latest

SonarQube Rules :

Introduction:

In the SonarQube platform, plugins contribute rules which are executed on source code to generate issues. Those issues are used to compute remediation cost . Rule Templates are provided by plugins to allow users to define their own rules in SonarQube .

There is two options to install a plugin into SonarQube :

- Automatically, from the SonarQube UI using the Update Center
- Manually

Quality Profiles :

Quality Profiles are collections of rules to apply during an analysis.

For each language there is a default profile. All projects not explicitly assigned to some other profile will be analyzed with the default.

Benefits of Using SonarQube :

- SonarQube increases productivity by enabling development teams to detect duplicate code.
- SonarQube facilitates the team members to reduce the size of application, code complexity, maintenance time and cost and make code easy to read and understand.
- Code quality becomes a part of development process and development teams. By enabling continuous code quality management, the software quality is raised and decreases the cost and risk of software management.
- SonarQube addresses not just bugs but also coding rules, test coverage, duplications, API documentation, complexity, and architecture, providing all these details in a dashboard.