

|  |
| --- |
|  |

**Jenkins Guidelines and Best Practices**

**Document History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version No. | Authored/Modified by | Modification Date | Reviewed by, Date | Remark / Change history |
| 1.0 | Abhijeet Kamble | 15th July 2015 |  | Initial doc |
| 1.1 | Aparna Nigdikar | 21st July 2015 |  | Added document history section |

# Table of Contents

[Table of Contents 2](#_Toc423524654)

[1. Introduction 3](#_Toc423524655)

[2. COnfiguration Guidelines FOr JENKINS 4](#_Toc423524656)

[3. USer Guidelines FOr user for jenkins 5](#_Toc423524657)

# 

# Introduction

Jenkins is a tool that provides continuous integration (CI) services.

**What is continuous integration (CI)?**

One of the most important aspects of the software development is the software quality, not just in the final product but also in the process of the development. In order to detect failures early on we need a system that can build test and deploy our software as frequently as is possible. Jenkins is a tool that allows us to do that.

Here, we use this tool for several propose, such as building the applications that are being developed within the company. Every time a developer updates the code, Jenkins builds the project with this change and if the task fails, warn the developer and the rest of team that something was wrong.

Another feature is the automatic execution of Unit Test created for each application and the Automatic Tests user level for Web applications. If any of these tests fail, as building applications, warn the team to take steps to correct the error.

In short, we have a powerful tool which we can ensure quality of the code to be checking through the project life cycle.

# COnfiguration Guidelines FOr Jenkins

Following are the guidelines for developers:

1. **Backup Jenkins home regularly.**

The Backup plugin adds a new Backup manager item in the Manage Jenkins page.

* Backing Jenkins's configuration (using the Backup Jenkins configuration link)
* Restoring Jenkins's configuration from a previous backup (using the Restore Jenkins configuration link)

1. **Use fingerprinting to manage dependencies.**

When you have interdependent projects on Jenkins, it often becomes hard to keep track of which version of this is used by which version of that. Jenkins supports "file fingerprinting" to simplify this, so make best use of it.

1. **The most reliable builds will be clean builds, which are built fully from source code control.**

To ensure a build can be reproducible, the build must be a clean build, which is built fully from Source Code Control. This practice also implies that all code including third-party jars, build scripts, release notes, etc. must be checked into Source Code Control.

1. **Always configure job to generate trend reports and automated testing while running builds.**

Trends help project managers and developers quickly visualize current project progress status. Moreover, unit testing is often not enough to provide confidence that the delivered software complies with the desired quality. The more you test the software, the better the delivered software complies with the desired quality.

1. **Setup different job for each maintenance or development branch we create.**

One of advantages of using CI tools is to detect problems early in the development lifecycle. Setting up a different job/project for each branch you create will help to maximize the benefit of detecting problems early as part of supporting parallel development efforts and reducing risk.

1. **Allocate different ports for parallel project builds and avoid scheduling all jobs to start at same time.**

Multiple jobs running at the same time often cause collisions. Try to avoid scheduling all jobs to start at the same time. Allocate a different port for parallel project builds to avoid build collisions.

# USer Guidelines FOr user for jenkins

Following are the guidelines for Lead:

1. **Always secure Jenkins.**

This best practice is around authenticating users and enforcing access control on a Jenkins instance  
in the default configuration, Jenkins does not perform any security checks. This means any person accessing the website can configure Jenkins and jobs, and perform builds. While this configuration is normally acceptable for intranet use and quick setup, it introduces high security risks, like someone accidentally deleting your build jobs, reconfiguring your job to run every minute, kicking off too many builds at the same time, reconfiguring your build instance, etc.

Role strategy plugin can be used to secure Jenkins which adds a new role-based strategy to ease and fasten user’s management. This strategy allows:

* Creating global roles, such as admin, job creator, anonymous, etc., allowing to set Overall, Slave, Job, Run, View and SCM permissions on a global basis.
* Creating project roles, allowing to set only Job and Run permissions on a project basis.
* Creating slave roles, allowing to set node-related permissions.
* Assigning these roles to users.

1. **Integrate Jenkins with issue tracking system like JIRA to reduce the need for maintaining change log.**

The integration helps to track changes as they are made, including build status, what build has been performed for this requirement or defects, and the link to the actual build results and artifacts.

* JIRA plugin is used to integrate Atlassian JIRA to Jenkins.

1. **Archive unused jobs before removing them.**

All unused jobs should be archived so they can be resurrected if the need arises.

1. **Setup email notifications mapping to all developer in the project, so that everyone on the team has his pulse on project’s current status.**

Configure each person on the people list with his or her correct email address and what role he or she is currently playing.

Email-Ext plugin is used to configure every aspect of email notifications. You can customize when an email is sent, who should receive it, and what the email says.