Jenkins Essentials

Continuous Integration – setting up the stage for a DevOps culture

Mitesh Soni
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 7 'Managing and Monitoring Jenkins'
- A synopsis of the book’s content
- More information on Jenkins Essentials
About the Author

**Mitesh Soni** is currently working as a technical specialist at IGATE. He is an SCJP, SCWCD, and VCP. While he has interest in DevOps and Cloud computing, his real passion is to play with kids, play with his camera, and capture photographs at Indroda Park. He loves programming in Java, and he finds design patterns fascinating. He lives in the capital of Mahatma Gandhi's home state. He loves to spend time alone and loves walking at Punit Van. He believes that without a sense of urgency, desire loses its value. He has earlier authored *Learning Chef* by Packt Publishing (https://www.packtpub.com/networking-and-servers/learning-chef).
Preface

DevOps is a buzz word in 2015 and will be for the coming years as per market trends by various research firms. In DevOps culture, business owners, development teams, operations teams, and QA teams collaborate to deliver outcome in a continuous and effective manner. It enables the organizations to more quickly grab opportunities and reduce the time taken to include customer feedback into new feature development or innovation. The end goal of DevOps is to reduce the time between the initial concept and the end result of the concept in the form of production ready applications. DevOps targets application delivery, new feature development, bug fixing, testing, and maintenance releases. It improves efficiency, security, reliability, predictability, and faster development and deployment cycles. It covers all SDLC phases from development, test, operations, and release.

Continuous integration (CI) and continuous delivery (CD) are a significant part of the DevOps culture. Jenkins is a fully featured technology platform that enables users to implement CI and CD. This helps users to deliver better applications by automating the application delivery life cycle. CI includes automation of build, test and package processes. CD includes the application delivery pipeline across different environments. Jenkins enables the user to utilize continuous integration services for software development in an agile environment. Continuous integration systems are a vital part of the agile team because they help enforce the principles of agile development. Continuous Integration is a significant part of the DevOps culture, and hence, many open source and commercial tools for continuous delivery utilize Jenkins or provide integration points. Jenkins enables agile teams to focus on work and innovations by automating the build, artifact management, and deployment processes, rather than worrying about manual processes. It can be used to build freestyle software projects based on Apache Ant and Maven 2 / Maven 3 projects. It can also execute Windows batch commands and shell scripts.
There are a number of ways to install Jenkins, and it can be used across different platforms such as Windows and Linux. Jenkins is available in the form of native packages of Windows, FreeBSD, OpenBSD, Red Hat, Fedora, CentOS, Ubuntu, Debian, Mac OS X, openSUSE, Solaris, OpenIndiana, Gentoo, or in the form of WAR file. The quickest and easiest way to use Jenkins is to use the WAR file. It can be easily customized with the use of plugins. There are different kinds of plugins available to customize Jenkins based on specific needs. Categories of plugins include source code management (that is, Git Plugin, CVS Plugin, and Bazaar Plugin), build triggers (that is, Accelerated Build Now Plugin and Build Flow Plugin), build reports (that is, CodeScanner Plugin and Disk Usage Plugin), authentication and user management (that is, Active Directory Plugin and Github OAuth Plugin), cluster management and distributed build (that is, Amazon EC2 Plugin and Azure Slave Plugin), and so on.

Jenkins is very popular among its users as it allows them to manage and control phases such as build, test, package, and static code analysis. It has won InfoWorld Bossies Award, 2011; O’Reilly Open Source Award, 2011; ALM&SCM; and so on. The main users of Jenkins are NASA, Linkedin, eBay, and Mozilla Foundation.

The following are some features that make Jenkins very popular:

- An open source tool with a web-based GUI.
- A Java-based continuous build system—easy to write plugins.
- Highly configurable tool—a plugin-based architecture that provides support to many technology, repositories, build tools, and test tools.
- The Jenkins user community is large and active. It has more than 1,000 open source plugins.
- This supports CI for .Net, iOS, Android, and Ruby development.
- This supports common SCM systems such as SVN, CVS, Git, and so on.
- This supports common test frameworks such as Junit, Selenium, and so on.

Jenkins speeds up the application development process through automation across different phases such as build, test, code analysis, and so on. It also enables users to achieve end-to-end automation for an application delivery life cycle.
What this book covers

Chapter 1, Exploring Jenkins, describes in detail the basics of continuous integration and provides an overview of Jenkins. This chapter also describes installation and configuration of Jenkins. It takes a jump-start tour through some of the key features of Jenkins and plugin installations as well. It will also cover the deployment pipeline and the rest of the chapters will cover implementing it.

Chapter 2, Installation and Configuration of Code Repository and Build Tools, describes in detail on how to prepare runtime environment for application life cycle management and configure it with Jenkins—an open source continuous integration tool. It will cover how to integrate Eclipse and code repository such as SVN and Git to create a base for continuous integration in the deployment pipeline, which is explained in Chapter 1, Exploring Jenkins.

Chapter 3, Integration of Jenkins, SVN, and Build Tools, describes in detail on how to create and configure build jobs for Java applications, and how to run build jobs and unit test cases. It covers all aspects of running a build to create a distribution file or WAR file for deployment.

Chapter 4, Implementing Automated Deployment, covers one step forward in the deployment pipeline by deploying artifacts in the local or remote application server. It will give insight into automated deployment and continuous delivery process, and also cover how to deploy applications on a public cloud platform using Jenkins.

Chapter 5, Hosted Jenkins, describes how to use Jenkins on Platform as a Service (PaaS) model, which is provided by popular PaaS providers such as Red Hat OpenShift and CloudBees. Considering CloudBees, it also covers details on how various customers are using Jenkins based on their requirements. This chapter will explore details on how to use Cloud-related plugins in Jenkins for an effective use of Jenkins.

Chapter 6, Managing Code Quality and Notifications, covers how to integrate static code analysis behavior into Jenkins. Code quality is an extremely vital feature that impacts an application's effectiveness, and by integrating it with Sonar, CheckStyle, FindBug, and other tools, you can get an insight into problematic portions of code.
Chapter 7, Managing and Monitoring Jenkins, gives an insight into management of Jenkins nodes and monitoring them with Java Melody to provide details on utilization of resources. It also covers how to monitor build jobs configured for Java applications and managing those configurations by keeping its backup. This chapter discusses the basic security configuration that is available in Jenkins for better access control and authorization.

Chapter 8, Beyond Basics of Jenkins – Leveraging "Must-have" Plugins, covers the advanced usage of Jenkins that are extremely useful in specific scenarios. Scenario-based use cases and usage of specific plugins that help development and operations teams are covered here for better utilization of Jenkins.
"Fall in the beginning + Fall often + Learn to recover quickly = Faster time to market"

– Anonymous

We learned Sonar integration with Jenkins, an overview of static code analysis plugins, and notification of build status in the last chapter. Now, it's time to focus on management and monitoring of Jenkins.

This chapter gives insight into management of Jenkins nodes and monitoring of them with Java Melody to provide details on utilization of resources. It also covers how to manage and monitor build jobs. This chapter describes basic security configuration in detail that is available in Jenkins for a better access control and authorization. The following is the list of topics that we will cover in this chapter:

• Managing Jenkins master and slave nodes
• Jenkins monitoring with JavaMelody
• Managing disk usage
• Build job-specific monitoring with the Build Monitor plugin
• Managing access control and authorization
• Maintaining role and project-based security
• Managing an admin account
• Audit Trail Plugin—an overview and usage
Managing Jenkins master and slave nodes

A master represents basic installation of Jenkins and handles all tasks for the build system. It can satisfy all user requests and has the capacity to build projects on its own. A slave is a system that is set up to reduce the burden of build projects from the master but delegation behavior depends on the configuration of each project. Delegation can be configured specifically to build job.

1. On the Jenkins dashboard, go to Manage Jenkins. Click on Manage Nodes link. It will provide information on all nodes, as shown in the following screenshot:

<table>
<thead>
<tr>
<th>Name</th>
<th>Architecture</th>
<th>Clock Difference</th>
<th>Free Disk Space</th>
<th>Free Swap Space</th>
<th>Free Temp Space</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>Linux (amd64)</td>
<td>In sync</td>
<td>5.86 GB</td>
<td>1.94 GB</td>
<td>5.86 GB</td>
<td>Dms</td>
</tr>
</tbody>
</table>

Data obtained 48 sec 48 sec 48 sec 48 sec 48 sec 48 sec

2. To create a slave node, click on New Node.

Adds a plain, dumb slave to Jenkins. This is called "dumb" because Jenkins doesn't provide higher level of integration with these slaves, such as dynamic provisioning. Select this type if no other slave types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.
3. Provide **Name**, **Description**, **Labels** and so on. Select **Launch slave agents via Java Web Start** as **Launch method**. Provide **Labels**; in our case, it is `java8`:

![Node Properties](image)

4. Click on **Save**. It will open a page that gives details on how to launch the slave node.

It will open a dialogue box for downloading the application.

6. Run Jenkins Remoting Agent.
A small window for the Jenkins slave agent will open.

The slave WindowsNode will be connected via the JNLP agent.

7. On the Jenkins dashboard, go to Manage Jenkins. Click on the Manage Nodes link. It will provide information on all nodes, as shown in the following screenshot. Verify both the nodes in the Build Executor Status section of the leftmost sidebar.
8. If we want to run a selective build job on to a specific node, then we can configure it build job-wise, as shown in the following screenshot. Check **Restrict where this project can be run** and provide **Label Expression** given to the specific node on the job configuration page.

9. Click on **Build Now** to execute build. Verify the console and find building remotely on WindowsNode we configured in the preceding section.
   It will check out the code on slave and perform operations on the specific node only.

Such configuration is useful where we want to run build job in a specific set of runtime environment, which is available on the specific node.
Jenkins monitoring with JavaMelody

The Monitoring plugin provides monitoring of Jenkins with JavaMelody. It provides charts of a CPU, memory, system load average, HTTP response time, and so on. It also provides details of HTTP sessions, errors and logs, actions for GC, heap dump, invalidate session(s), and so on. Install the Monitoring plugin from the Jenkins Dashboard.

1. On the Jenkins dashboard, click on Manage Jenkins. Click on Monitoring of Jenkins master, as shown in the following screenshot:
2. It will open the statistics of JavaMelody monitoring, as shown in the following screenshot. Observe all statistics:

![Statistics of JavaMelody monitoring](image)

3. Scroll down the page and we will find **Statistics system errors logs**.

![Statistics system errors logs - 1 day](image)
4. To get more information, click on the Details link of any section. Statistics of HTTP are as shown in the following figure:

![Statistics http - 1 day](image)

5. Explore more at https://wiki.jenkins-ci.org/display/JENKINS/Monitoring to get more details on the Monitoring plugin.

### Managing disk usage

1. Disk Usage Plugin records disk usage. Install Disk Usage Plugin from the Jenkins dashboard.
Managing and Monitoring Jenkins

2. Once the plugin is successfully installed, we will get the **Disk usage** link on the Manage Jenkins page, as shown in the following screenshot:

![Disk usage screenshot](image)

3. The Disk Usage plugin will show project-wise details for all jobs and all workspace. It will also display **Disk Usage Trend**.

![Disk Usage Trend](image)

<table>
<thead>
<tr>
<th>Project name</th>
<th>Jobs</th>
<th>Builds all</th>
<th>Builds locked</th>
<th>All workspace</th>
<th>Not current slave workspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>sente-build</td>
<td>233 MB</td>
<td>180 MB</td>
<td>-</td>
<td>53 MB</td>
<td>-</td>
</tr>
<tr>
<td>PetClinic-Test</td>
<td>115 MB</td>
<td>57 MB</td>
<td>-</td>
<td>116 MB</td>
<td>-</td>
</tr>
<tr>
<td>PetClinic-Sonar</td>
<td>115 MB</td>
<td>57 MB</td>
<td>-</td>
<td>116 MB</td>
<td>-</td>
</tr>
<tr>
<td>CounterApp</td>
<td>120 MB</td>
<td>97 MB</td>
<td>-</td>
<td>67 MB</td>
<td>-</td>
</tr>
</tbody>
</table>

To get more details on Disk usage plugin, visit [https://wiki.jenkins-ci.org/display/JENKINS/Disk+Usage+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Disk+Usage+Plugin).
Build monitoring with Build Monitor Plugin

**Build Monitor Plugin** provides a detailed view of the status of selected Jenkins jobs. It provides the status and progress of selected jobs and names of people who might be responsible for "breaking the build". This plugin supports the Claim plugin, View Job Filters, Build Failure Analyzer, and CloudBees Folders plugin.

1. The Dashboard View plugin will be used for creating a view that provides details on build job-specific monitoring. Create a new view and select **Build Monitor View**.

![Dashboard View plugin](image)
Managing and Monitoring Jenkins

2. Select Jobs and save the details.

3. Click on the newly created view, and we will get a similar type of screen as given in the following screenshot:

To get more details on plugin, visit https://wiki.jenkins-ci.org/display/JENKINS/Build+Monitor+Plugin.
Managing access control and authorization
Jenkins supports several security models, and can integrate with different user repositories.

1. Go to the Jenkins dashboard, click on Manage Jenkins, and click on Configure Global Security.
2. Click on Enable security.

All options will be visible once we enable security, as shown in the following screenshot:
3. Click on **Jenkins' own user database**. Click on **Save**.

4. Now, click on the **sign up** link on the top-right corner. Provide **Username**, **Password**, **Full name**, and **E-mail address**.
5. Click on the **log in** link on the dashboard.

We will get the Jenkins dashboard with the username in the top-right corner.
6. Click on **People** to verify all users.

![People](image)

7. On the Jenkins dashboard, click on **Manage Jenkins**. Click on **Manage Users**.

![Manage Users](image)

We can edit user details on the same page. This is a subset of users, which also contains auto-created users.
Maintaining roles and project-based security

For authorization, we can define **Matrix-based security** on the **Configure Global Security** page.

1. Add group or user and configure security based on different sections such as **Credentials**, **Slave**, **Job**, and so on.
2. Click on **Save**.

---

**Authorization**

- Anyone can do anything
- Legacy mode
- Logged in users can do anything
- **Matrix-based security**

<table>
<thead>
<tr>
<th>User/group</th>
<th>Overall</th>
<th>Credentials</th>
<th>Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use user/group to add [Add]

- Project-based Matrix Authorization Strategy

**Escaped HTML**

Treats all input as plain text. HTML angle characters like `<` and `&` are escaped to their respective character entities.

We can use multiple users for matrix-based security, as shown in the following screenshot:

---

**Authorization**

- Anyone can do anything
- Legacy mode
- Logged in users can do anything
- **Matrix-based security**

<table>
<thead>
<tr>
<th>User/group</th>
<th>Overall</th>
<th>Credentials</th>
<th>Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use user/group to add [Add]

- Project-based Matrix Authorization Strategy

**Escaped HTML**

Treats all input as plain text. HTML angle characters like `<` and `&` are escaped to their respective character entities.
Managing and Monitoring Jenkins

3. Try to access the Jenkins dashboard with a newly added user who has no rights, and we will find the authorization error.

![Access Denied](image)

4. Now provide overall read rights; build, read, and workspace rights for job for newly added users.

![Authorization](image)

5. Log in with the newly added user and verify that we can see the dashboard. We can't see the Manage Jenkins link as we have provided those rights.

![Dashboard](image)
6. Click on any build job. The build link is available as we have given rights but the configure link is not available as rights were not given for it.

7. We can also set Project-based Matrix Authorization Strategy.
8. Go to a specific build jobs' configuration and **Enable project-based security**.

![Project Name Configuration](image1)

9. Assign rights to different users and log in with the specific username to verify whether authorization strategy is working or not.

![Jenkins Authorization](image2)

10. Verify the build details also, as shown in the following screenshot:
We've covered basics of security configuration in Jenkins. Explore more on the other options as an exercise. In case, authorization is not correctly set, then it can be corrected by editing `config.xml`. Consider it as self-study.

**Audit Trail Plugin – an overview and usage**

Audit Trail Plugin keeps a log of users who performed particular Jenkins operations, such as configuring jobs. This plugin adds an Audit Trail section in the main Jenkins configuration page.

Install the Audit Trail Plugin.
In Jenkins configuration, configure **Loggers**, as shown in the following screenshot:

Stop the Jenkins server and start it again. Run any build job and open log files to verify log records.

To get more details, visit [https://wiki.jenkins-ci.org/display/JENKINS/Audit+Trail+Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Audit+Trail+Plugin).
Self-test questions

Q1. What are the different ways to make slave node online?

1. Launch an agent from the browser on slave
2. Run the `slave-agent.jnlp` command from the command line
3. Run `java -jar slave.jar`
4. All of the above

Q2. For what options does Jenkins monitoring provide charts?

1. CPU
2. Memory
3. System load average
4. HTTP response time
5. All of the above

Q3. What are the options for Security Realm in Jenkins?

1. Delegate to Servlet Container
2. Jenkins' own user database
3. LDAP
4. Unix user/group database
5. All of the above

Summary

Whatever good things we build end up building us. In this chapter, we covered concepts of master and slave nodes, how to monitor build jobs, and reporting of statistics with management features. We also understood how to secure Jenkins environment with authentication and authorization configurations by using role-based security. We saw how the audit trail plugin stores audit details in Jenkins.

In the next chapter, we will cover some important plugins that add a significant value to Jenkins. Let's enjoy the last journey before we say goodbye.
Where to buy this book
You can buy Jenkins Essentials from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.