

Continuous Integration with Jenkins-CI

Table of Contents

Chapter 1 - Introduction to Continuous Integration and Jenkins-CI.....	5
1.1 Agile Development.....	5
1.2 Agile Development (cont'd).....	5
1.3 Agile Development (cont'd).....	6
1.4 What is Continuous Integration.....	6
1.5 What is Continuous Integration (cont'd).....	6
1.6 What is Continuous Integration (cont'd).....	7
1.7 What is Continuous Integration (cont'd).....	7
1.8 Typical Setup for Continuous Integration.....	8
1.9 Typical Setup for Continuous Integration.....	8
1.10 Jenkins Continuous Integration.....	9
1.11 Jenkins Features.....	9
1.12 Running Jenkins.....	9
1.13 Summary.....	10
Chapter 2 - Installing and Running Jenkins.....	11
2.1 Downloading and Installing Jenkins.....	11
2.2 Running Jenkins as a Stand-Alone Application.....	12
2.3 Running Jenkins on an Application Server.....	13
2.4 Installing Jenkins as a Windows Service.....	13
2.5 Summary.....	14
Chapter 3 - A Jenkins Job.....	15
3.1 Different types of Jenkins job.....	15
3.2 Configuring Source Code Management(SCM).....	16
3.3 Working with Subversion.....	17
3.4 Build Triggers.....	17
3.5 Schedule Build Jobs.....	18
3.6 Polling the SCM.....	19
3.7 Maven Build Steps.....	19
.....	20
3.8 Summary.....	20
Chapter 4 - Securing Jenkins.....	21
4.1 Jenkins Security - Overview.....	21
4.2 Jenkins Security.....	22
4.3 Authentication.....	22
4.4 Authorization.....	23
4.5 Confidentiality.....	23
4.6 Activating Security.....	24
4.7 Configure Authentication.....	24
4.8 Using Jenkins's Internal User Database.....	25
4.9 Creating Users.....	26
4.10 Authorization.....	26
4.11 Authorization.....	27
4.12 Matrix-Based Security.....	28
4.13 Note – Create the Administrative User.....	28

4.14 Project-based Matrix Authorization.....	29
4.15 Project-Based Authentication.....	30
4.16 Conclusion.....	30
Chapter 5 - Jenkins Plugin.....	31
5.1 Introduction.....	31
5.2 Jenkins Plugins - SCM.....	31
5.3 Jenkins Plugins – Build and Test.....	31
5.4 Jenkins Plugins – Analyzers.....	32
5.5 Jenkins for Teams.....	32
5.6 Installing Jenkins Plugins.....	32
5.7 Summary.....	35
Chapter 6 - Distributed Builds with Jenkins.....	37
6.1 Distributed Builds - Overview.....	37
6.2 Distributed Builds – How?.....	37
6.3 Slave Machines.....	37
6.4 Configure Jenkins Master.....	38
6.5 Configure Projects.....	38
6.6 Conclusion.....	38
Chapter 7 - Best Practices for Jenkins.....	39
7.1 Best Practices.....	39

Chapter 1 - Introduction to Continuous Integration and Jenkins-CI

Objectives

Key objectives of this chapter

- Agile Development
- Continuous Integration
- History of Jenkins

1.1 Agile Development

- Agile Development begins with Extreme Programming (XP)
 - ◇ Invented/Promoted by Kent Beck and associates
- Beck describes XP as:
 - ◇ A philosophy of software development based on the values of communication, feedback, simplicity, courage, and respect
 - ◇ A body of practices proven useful in improving software development.
 - ◇ A set of complementary principles, intellectual techniques for translating the values into practice, useful when there isn't a practice handy for your particular problem.
 - ◇ A community that shares these values and many of the same practices.

1.2 Agile Development (cont'd)

- As time went on, people took inspiration from XP and developed other approaches with much the same goals
 - ◇ Speed
 - ◇ Include the Customer Early
 - ◇ Don't wait til you understand the entire problem - you'll understand it while you fix it
 - ◇ Deliver usable value at every iteration

- ◇ Make reasonable use of modelling and development tools

1.3 Agile Development (cont'd)

- Two themes emerge from XP and Agile Development
 - ◇ Trust
 - Vendor, customer, managers, developers all need to trust each other
 - ◇ Automation
 - If we're going to go fast, we need tools that make it easy to produce software
- Trust is a social problem – solution is all about different approaches to process, requirements gathering, project management, etc.
- Automation is about tools – and Continuous Integration is a primary tool of Agile Development

1.4 What is Continuous Integration

- “Integrate and test changes after no more than a couple of hours.”
 - Beck & Andres “Extreme Programming Explained”
 - ◇ Integrate and build the complete product
 - ◇ If a website, build the website
 - ◇ If a GUI install, build the installer
 - ◇ Etc.

1.5 What is Continuous Integration (cont'd)

- Purposes
 - ◇ You find out quickly about integration problems
 - ◇ Immediately evident if a code change “breaks the build”
 - ◇ Prevents a long drawn-out integration step at the end of code changes

- ◇ Should be complete enough that eventual first deployment of the system is “no big deal”.

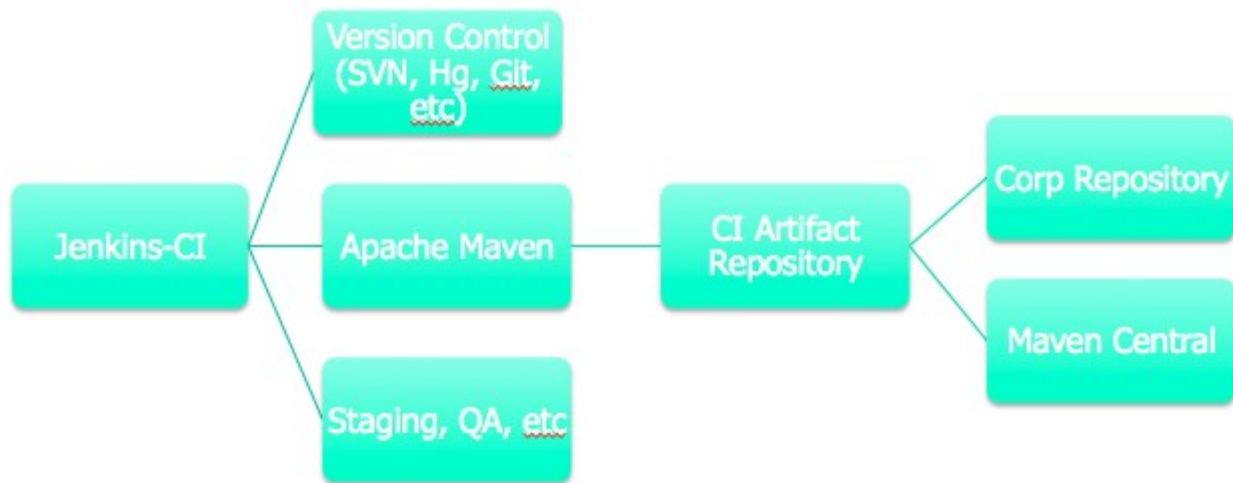
1.6 What is Continuous Integration (cont'd)

- Usually, CI goes along with Test-First design and automated QA
 - ◇ Run the unit-test suite and smoke testing to ensure the build isn't broken
- Clearly, automatic build is a pre-requisite
 - ◇ So we need a build system – Maven for instance
- Can be synchronous or asynchronous
 - ◇ Asynchronous – Integration happens automatically on code committal
 - ◇ Synchronous – Trigger the build manually after a development session

1.7 What is Continuous Integration (cont'd)

- Side Effects
 - ◇ Generate development reports
 - ◇ Install to QA, User Test, etc
 - ◇ Always have an installable artifact
- It's a great time to generate development metrics
 - ◇ E.g. Code Coverage, Standards Compliance, Static Analysis

1.8 Typical Setup for Continuous Integration



1.9 Typical Setup for Continuous Integration

- Notes:
 - ◇ CI system gets the code directly from version control
 - ◇ Build is independent of any local artifacts that are on the developer's machine
 - Controlled links to corporate repository and Maven Central
 - Goal is to ensure that the package can be built from the corporate repository
 - ◇ Jenkins can have connections to a deployment environment
 - Production Staging
 - User Acceptance Testing
 - QA
 - Load Test
 - Etc...
 - ◇ It turns out that if we use Maven, we ABSOLUTELY NEED a local repository manager

- More info to follow...

1.10 Jenkins Continuous Integration

- Originally developed at Sun by Kohsuke Kawaguchi?
 - ◇ Originally “Hudson” on java.net circa 2005
 - ◇ Jenkins forked in November 2010
 - ◇ Hudson is still live, part of Eclipse Foundation
 - ◇ But Jenkins seems to be far more active

1.11 Jenkins Features

- Executes jobs based on a number of triggers
 - ◇ Change in a version control system
 - ◇ Time
 - ◇ Manual Trigger
- A Job consists of some instructions
 - ◇ Run a script
 - ◇ Execute a Maven project or Ant File
 - ◇ Run an operating system command
- User Interface can gather reports
 - ◇ Each job has a dashboard showing recent executions

1.12 Running Jenkins

- You can run Jenkins Standalone or inside a web container
- You can setup distributed instances that cooperate on building software
- Can setup jobs in place of what might have been script commands.

1.13 Summary

- Continuous Integration is a powerful tool for agile software development
- Jenkins is currently the dominant Continuous Integration tool

Chapter 2 - Installing and Running Jenkins

Objectives

Key objectives of this chapter

- Downloading and Installing Jenkins.
- Running Jenkins as a Stand-Alone Application.
- Running Jenkins on an Application Server.
- Installing Jenkins as a Windows Service.
- Build Steps
- Reporting on Test Results
- Working with Maven Build Jobs

2.1 Downloading and Installing Jenkins

- Download Jenkins from the Jenkins website (<http://jenkins-ci.org>)
 - Jenkins is a dynamic project, and new releases come out at a regular rate.
- Jenkins distribution is bundled in Java web application (a WAR file).
- Windows users, there is a graphical Windows installation MSI package for Jenkins.



Jenkins

An extendable open source continuous integration server

[BLOG](#)[CONNECT](#)[BUG TRACKER](#)[WIKI](#)[CI](#)[TUTORIALS](#)[ARCHIVES](#)[DONATION](#)[ABOUT](#)

Meet Jenkins

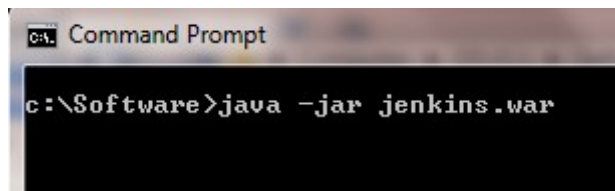
Find out what Jenkins is and get started.

Download Jenkins

[Release](#)[Long-Term Support Release](#)**Java Web Archive (.war)****Latest and greatest (1.539)**[changelog](#) | [past releases](#) | [RC](#)[upgrading from Hudson?](#)

2.2 Running Jenkins as a Stand-Alone Application

- Jenkins comes bundled as a WAR file that you can run directly using an embedded Servlet container.
- Jenkins uses the lightweight Servlet engine to allow you to run the server out of the box.
- Flexible to install plug-ins and upgrades on the fly.
- To run Jenkins using the embedded Servlet container, just go to the command line and type the following:



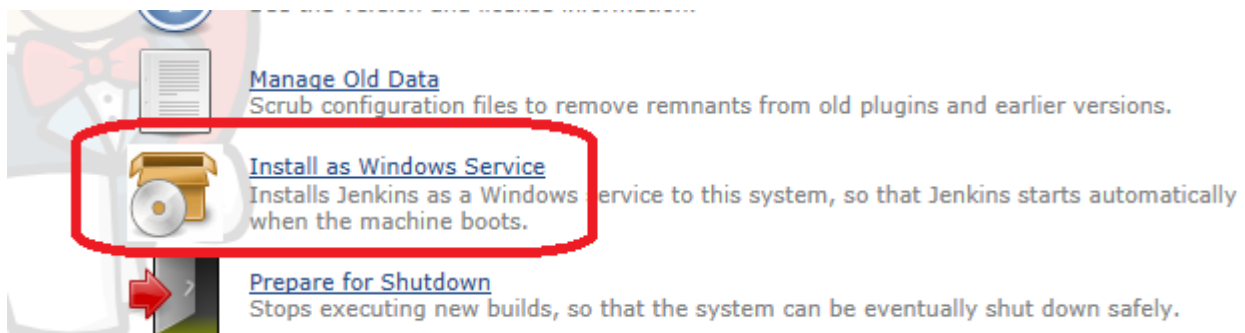
- The Jenkins web application will now be available on port 8080.
- Jenkins can be access directly using the server URL (http://localhost:8080).
- To stop Jenkins, just press Ctrl-C.
- Useful Options:
 - --httpPort
 - By default, Jenkins will run on the 8080 port.
 - Jenkins can be start on different port using the --httpPort option:
 - java -jar jenkins.war --httpPort=8081
 - --logfile
 - By default, Jenkins writes its logfile into the current directory.
 - Option to redirect your messages to other file:
 - java -jar jenkins.war --logfile=<C:/Software/log/jenkins.log>
- These options can also be set at JENKINS_HOME/jenkins.xml config file.

2.3 Running Jenkins on an Application Server

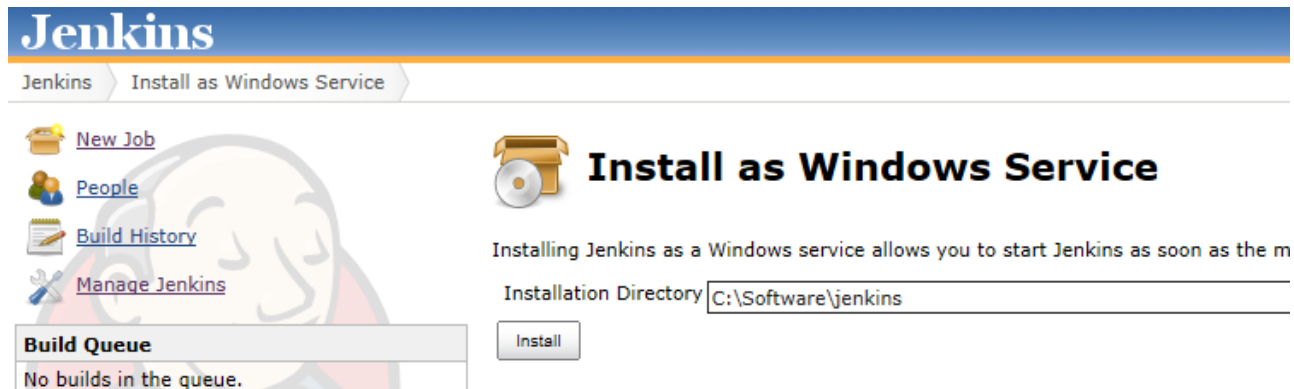
- Jenkins distribution WAR file can be easily deploy to standard Java application server such as Apache Tomcat, Jetty, or GlassFish.
- Jenkins will be executed in its own web application context (typically "jenkins").
 - URL : <http://localhost:8080/jenkins>.

2.4 Installing Jenkins as a Windows Service

- In production, installation of Jenkins on a Windows box is essential to have it running as a Windows service.
 - Jenkins will automatically start whenever the server reboots
 - Can be managed using the standard Windows administration tools.
- Jenkins using the windows installer automatically runs Jenkins as a windows service.
 - No need to do anything.
- First, Start the Jenkins server on your machine:
 - `java -jar jenkins.war`
- Connect to Jenkins by going to the following URL `http://<hostname>:8080`
- Look for "Install as Windows Service" link in the "Manage Jenkins" page.



- Clicking this link shows you the installation screen:



- Choose a directory where Jenkins will be installed, JENKINS_HOME and used to store data files and programs
- Upon successful completion of the installation, you should see a page asking you to restart Jenkins.
- At this point you can use the service manager to confirm that Jenkins is running as a service.

Name	Description	Status	Startup Type	Log On As
Indexing Service	Indexes co...		Manual	Local System
IPSEC Services	Manages I...	Started	Automatic	Local System
Java Quick Starter	Prefetches...	Started	Automatic	Local System
jenkins	This servic...	Started	Automatic	Local System
Windows Firewall	Protects co...	Started	Automatic	Local System

2.5 Summary

- How to install and run Jenkins.
- Learned a few basic tips on how to maintain your Jenkins installation once running.
- Learned how easily install, stand-alone application using WAR file.

Chapter 3 - A Jenkins Job

Objectives

Key objectives of this chapter

- Different types of Jenkins job.
- Configuring Source Code Management.
- Working with Subversion
- Build Triggers.
- Schedule Build Jobs.
- Maven Build Steps

3.1 Different types of Jenkins job

- Jenkins supports several different types of build jobs
 - Freestyle software project:
 - Freestyle projects are general purpose and allow to configure any sort of build job.
 - Highly flexible and very configurable.
 - Maven project:
 - Jenkins understands Maven pom files and project structures.
 - Reduce the work needed to do to set up the project.
 - Monitor an external job:
 - Monitoring the non-interactive execution of processes, such as cron jobs.
 - Multi-configuration job:
 - Run same build job in many different configurations.
 - Powerful feature, useful for testing an application in many different environments.



3.2 Configuring Source Code Management(SCM)


- Monitors version control system, and checks out the latest changes as they occur.
- Compiles and tests the most recent version of the code.
- Simply check out and build the latest version of the source code on a regular basis.
- SCM configuration options in Jenkins are identical across all sorts of build jobs.
- Jenkins supports CVS and Subversion out of the box, with built-in support for Git
- Integrates with a large number of other version control systems via plugins.

3.3 Working with Subversion

- Simply provide the corresponding Subversion URL
 - Supported protocols http, svn, or file.
- Jenkins will check that the URL is valid as soon as you enter it.
- If authentication needed, Jenkins will prompt you for the corresponding credentials automatically, and store them for any other build jobs that access this repository.
- Fine-tune Jenkins to obtain the latest source code from your Subversion repository by selecting an appropriate value in the Check-out Strategy drop-down list.
- Choose check-out Strategy as “Use ‘svn update’ as much as possible, with ‘svn revert’ before update”
 - No local files are modified, though it will not remove any new files that have been created during the build process.

Source Code Management

☐ CVS
☐ None
☒ Subversion

Modules Repository URL
 **Repository URL is required.**

Local module directory (optional)


Check-out Strategy
Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser

3.4 Build Triggers

- In Freestyle build, there are three basic ways a build job can be triggered
 - Start a build job once another build job has completed
 - Kick off builds at periodical intervals
 - Poll the SCM for changes

Build Triggers

☒ Build after other projects are built
Project names
 **No project specified**
Multiple projects can be specified like 'abc, def'

☒ Build periodically
Schedule

☒ Poll SCM
Schedule

Ignore post-commit hooks ☐

3.5 Schedule Build Jobs

- Build job at regular intervals.
- For all scheduling tasks, Jenkins uses a cron-style syntax, consisting of five fields separated by white space in the following format:
 - ◇ 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.
- There are also a few short-cuts:
 - ◇ “*” represents all possible values for a field. For example, “* * * * *” means “once a minute.”
 - ◇ You can define ranges using the “M–N” notation. For example “1-5” in the DOW field would mean “Monday to Friday.”

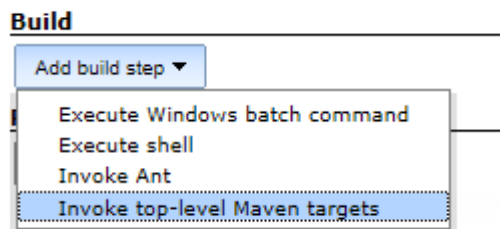
- ◇ You can use the slash notation to defined skips through a range. For example, “*/5” in the MINUTE field would mean “every five minutes.”
- ◇ A comma-separated list indicates a list of valid values. For example, “15,45” in the MINUTE field would mean “at 15 and 45 minutes past every hour.”

3.6 Polling the SCM

- Poll SVN server at regular intervals if any changes have been committed.
- Jenkins kicks off a build, source code in the project, .
- Polling frequently to ensure that a build kicks off rapidly after changes have been committed.
- The more frequent the polling is, the faster the build jobs will start, and the more accurate.
- In Jenkins, SCM polling is easy to configure, and uses the same cron syntax we discussed previously.

3.7 Maven Build Steps

- Jenkins has excellent Maven support, and Maven build steps are easy to configure and very flexible.
- Select “Invoke top-level Maven targets” from the build step lists.



- Select a version of Maven to run (if you have multiple versions installed)
- Enter the Maven goals you want to run. Jenkins freestyle build jobs work fine with both Maven 2 and Maven 3.
- The optional POM field lets you override the default location of the Maven

pom.xml file.

■

Build

Invoke top-level Maven targets

Maven Version	<input type="text" value="apache-maven-3.1.1"/>
Goals	<input type="text" value="clean install"/>

3.8 Summary

- Learned how Jenkins checking out the source code of a project.
- How to set-up and schedule a job.
- Setup build for Maven project.

Chapter 4 - Securing Jenkins

Objectives

Key objectives of this chapter

- Overall view of Jenkins Security
- Authentication Options
- Authorization

4.1 Jenkins Security - Overview

- Jenkins is used to build software
 - ◇ Source code is available in the workspace
- Code built with Jenkins is frequently “unreleased” development or test builds
 - ◇ Need to make sure only authorized users can access this software
- Jenkins can be used to execute admin jobs
 - ◇ Prevent access to prevent breaches and denial-of-service
- Jenkins is highly configurable
 - ◇ So we need to prevent accidental configuration changes

Jenkins Security Overview

Why should you secure Jenkins?

In most environments, it goes without saying that security and access control should be enforced for any widely-available system. Nonetheless, it's worthwhile to consider a few things about Jenkins and the resources it makes available:

- Jenkins is frequently used to build software. This means that the source code for your software is available through the Jenkins web interface, since each project's workspace will typically include the source code that is checked-out through a version control system.
- The software that is built with Jenkins is frequently “unreleased”, or test builds. You probably don't want these versions to be available for download or installation, except to selected people and controlled systems.

- Jenkins can also be used to schedule jobs or automate administrative tasks. Some tasks, like backups, or maintenance operations, have the potential to slow down or stop operational systems, effectively allowing a denial-of-service attack if the ability to launch these tasks is not controlled.
- In all cases, Jenkins has a great deal of capability and is highly configurable. Although this configurability is a good thing, it also implies that uncontrolled access could lead to breakage of important jobs or tasks. A lot of security stems from a need to protect operational systems from the merely curious and/or untrained.

4.2 Jenkins Security

- ◇ Security breaks down into separate issues:
 - Authentication
 - Authorization
 - Confidentiality

Jenkins Security

As with any application security challenge, Jenkins security devolves into sub-tasks:

- Authentication
- Authorization
- Confidentiality

4.3 Authentication

- ◇ The task of identifying a user
- ◇ Authentication happens against a user registry
- ◇ Jenkins can use LDAP, operating system, servlet container, or internal registry

Authentication

Authentication is the task of identifying a user. In Jenkins, we will need to decide what kind of user registry we will authenticate against.

4.4 Authorization

- ◇ Authorization is the task of deciding what operations a user can perform, once they are authenticated
- ◇ Jenkins allows a few different options for authentication
 - Anyone can do anything
 - Authorized Users can do anything
 - Permissions Matrix
 - Project-Based Permissions Matrix

Authorization

Authorization is the task of identifying which operations an authenticated user can perform, on which projects.

4.5 Confidentiality

- ◇ Making sure that unauthorized parties don't get information they shouldn't have
- ◇ Two aspects in Jenkins
 - Control access to projects (really authentication)
 - Web confidentiality
 - Use secure http
 - Handled by the servlet container or httpd

Confidentiality is the task of making sure that unauthorized parties don't acquire information that they shouldn't have. In Jenkins, there are two aspects to this confidentiality: One aspect is controlling access to projects, so that users can only access projects and artifacts that they are approved for. Another aspect is plain web confidentiality – using secure http or integrating with a secure http server.

4.6 Activating Security

- ◇ Manage Jenkins --> Configure Global Security
- ◇ Click the “Enable Security” checkbox
 - Jenkins will then display other options



Activating Security

Like most things, security is configured from the “Manage Jenkins” page. From here, we can select “Configure Global Security”. In fact, if you haven't configured security yet, Jenkins will remind you at the top of the “Manage Jenkins” page. Clicking the “Setup Security” button will take you to the “Configure Global Security” page. Once there, check the “Enable Security” checkbox to display a number of other options.

4.7 Configure Authentication

- ◇ Authentication is performed within a “realm”
- ◇ Possibilities
 - Use the servlet container's authentication
 - Different containers can be setup for different authentication techniques
 - Jenkins' own User Database
 - LDAP

- Unix User/Group Database
 - Using Pluggable Authentication Module (PAM)

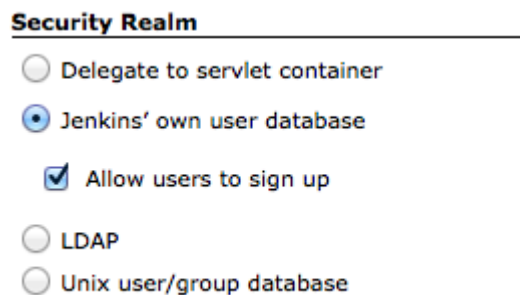
Configure Authentication

Authentication in Jenkins is performed inside a “Security Realm”, a concept that is often used in web servers to allow different applications to exist with different sets of users and authentication data. There are a few options:

- Delegate to servlet container. Under this option, Jenkins does not perform any authentication on its own, but allows the container to perform authentication. You might want to use this option if your web server or servlet container is already setup for security, or if it has some custom mechanism (e.g a proxy-based single sign-on system) to handle authentication.
- Jenkins' own user database. Under this scenario, Jenkins maintains its own list of users and passwords. You have the option to let users sign up on their own (I.e. add their names into the user database) or you can have an administrator add users.
- Lightweight Directory Access Protocol (LDAP). Jenkins can authenticate against an LDAP server, like Microsoft Active Directory, OpenLDAP or Lotus Domino.
- Unix user/group database. Jenkins can authenticate against the underlying Unix/Linux system using the Pluggable Authentication Module (PAM) library. You might want to use this option if you already have users setup as Unix or Linux users. This system will also use extensions like Network Information System (NIS) if the underlying *nix system is set up for them.

4.8 Using Jenkins's Internal User Database

- ◇ In the “Configure Global Security” screen, select “Jenkins own user database”



The screenshot shows the 'Security Realm' configuration section of the Jenkins 'Configure Global Security' screen. It features a title bar 'Security Realm' with a horizontal line underneath. Below the title bar are five radio button options: 'Delegate to servlet container', 'Jenkins' own user database' (which is selected with a blue dot), 'LDAP', and 'Unix user/group database'. Additionally, there is a checked checkbox labeled 'Allow users to sign up'.

- ◇ Choose whether users can “sign up” for a user id
 - Otherwise, administrator needs to create users

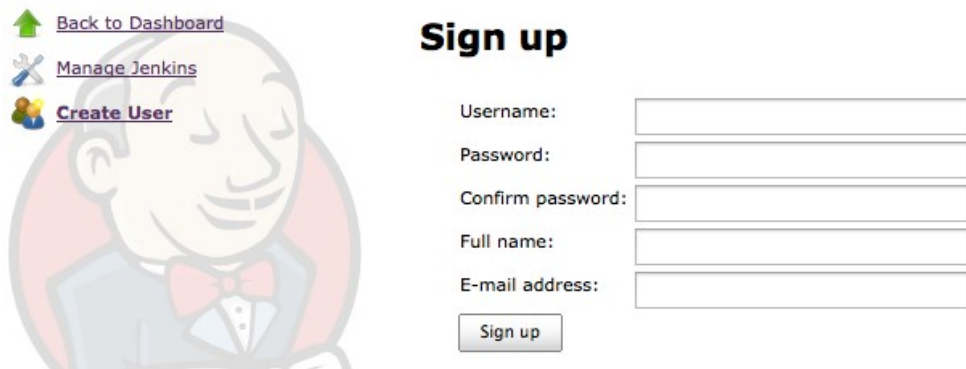
Using Jenkins' Internal User Database

To enable this option, go to the Configure Global Security screen and select “Jenkins' own user database” as the security realm.

You can also choose whether to allow people to “sign up” or create their own entry in the user database. In most cases, you would allow people to sign up on their own, but if you want, you can disable this option. If this option is disabled, then an administrator needs to create users manually through the “Manage Jenkins” console.

4.9 Creating Users

- ◇ Manage Jenkins -> Manage Users -> Create User



Creating Users

Manage Jenkins -> Manage Users -> Create User

4.10 Authorization

- ◇ Five options

Authorization

- ☒ Anyone can do anything
- ☐ Legacy mode
- ☐ Logged-in users can do anything
- ☐ Matrix-based security
- ☐ Project-based Matrix Authorization Strategy

Authorization

Once users are authenticated, Jenkins can grant access to its functionality to users.

There are five options:

The “xxx can do anything” modes are self-explanatory. “Legacy” mode recreates an old security policy where users with the servlet “admin” role can do anything, but other users are limited to read-only access.

4.11 Authorization

- ◇ Typically we want “Matrix-based security” or “Project-based Matrix Authorization Strategy”
- ◇ Authorization is separate from the authentication system
 - Need to enter userid or group entries separately from the users

Typically, you want to use either “Matrix-based security” or “Project-based Matrix Authorization Strategy”. With these settings, you can set permissions on a per-user, or per-user-per-project basis.

The authorization entries exist separately from the authentication system, so you need to enter the userid or group entries separately from the users (this separation is required so as to support the external authentication. Jenkins has no way of finding out which users exist; it just authenticates using the credentials that the user provides. As such, Jenkins has no way to list all the possible users; you need to list the users individually. Alternately, you can list the groups to which the users belong (Note: Jenkins' internal user database does not support groups – this function only works for LDAP or Unix password authentication).

4.12 Matrix-Based Security

- ◇ Each user is granted a set of permissions that apply to various operations

☒ Matrix-based security

User/group	Overall					Credentials				
	Administer	Read	RunScripts	UploadPlugins	ConfigureUpdateCenter	Create	Update	View	Delete	Manag
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"/>	

User/group to add:

- ◇ Add an entry for the user id, then select the appropriate check boxes

Matrix-Based Security

First, you need to add an entry for the user id or group that we want to set the permissions for. Then you can click the checkboxes for the permissions you want to set.

4.13 Note – Create the Administrative User

- ◇ Create an administrative user at the same time you turn on security
 - Grant full permissions to this user
- ◇ If there's no admin user, but you've turned on matrix permissions, then no-one has permissions
 - You're locked out!
- ◇ See notes for recovery from lock-out

A Special Note on Creating an Administrative User

Do it at the same time you're turning on security! There's a window of opportunity to lock yourself out of Jenkins. If you enable the matrix based security and save the screen without having created a user and assigned full permissions to that user, you'll be locked out. So make sure that you have a user

before you enable the matrix security.

If you do get locked out (this can also happen if you enable one of the other authentication mechanisms, and there's a problem, for instance the LDAP server is unavailable), it's fairly simple to restore Jenkins. The procedure is:


- Shut down Jenkins (either stop the service or stop the web server instance hosting Jenkins)
- Locate Jenkins' home directory (typically \$HOME/.jenkins).
- Edit the file 'config.xml'. Remove the 'authorizationStrategy' and 'securityRealm' elements. Save the file
- Restart Jenkins.

By removing the authentication and authorization entries, you will return Jenkins to the 'unsecured' state, and you can go back in and setup security correctly, without needing to log in.

4.14 Project-based Matrix Authorization

- ◇ Same global user/permission grid
- ◇ You still need an administrative user
 - Grant full permissions to the admin user
- ◇ Additionally, you can set permissions by project

Project-based Matrix Authorization Strategy

User/group	Overall					Credentials				
	Administer	Read	RunScripts	UploadPlugins	ConfigureUpdateCenter	Create	Update	View	Delete	ManageDomain
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"/>
 admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

User/group to add:

Add

Project-Based Authorization

With this option selected, you also have the same global user/permission grid, but in addition to permissions granted to users globally, you can also set permissions on a per-project basis. You should

still setup an administrative user (or multiple users) with appropriate global permissions:

4.15 Project-Based Authentication

- ◇ You will now have an option in each job's configuration to enable project-based security

☒ Enable project-based security

☐ Block inheritance of global authorization matrix

User/group	Credentials						
	Create	Update	View	Delete	Manage	Domains	Delete
Anonymous	<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:

- ◇ Add users to this permission list, for the project
- ◇ Permissions are additive
 - Users with global permissions still have those permissions for the project

Project-Based Authentication

You can add users to this permissions list, similar to how you added them in the global page. Note that the permissions are additive – Users with global permissions to create or edit jobs will still have them for a project, even if the project itself doesn't grant them permissions.

4.16 Conclusion

- ◇ Configure security through the “Configure Global Security” screen
- ◇ There are a variety of authentication and authorization options available.

Chapter 5 - Jenkins Plugin

Objectives

Key objectives of this chapter

- Introduction
- Jenkins Plugins – SCM
- Jenkins Plugins – Build and Test
- Jenkins Plugins – Analyzers
- Jenkins for Teams
- Installing Jenkins Plugins

5.1 Introduction

- Jenkins has over 300 plugins.
 - ◇ Software Configuration Management(SCM).
 - ◇ Test Frameworks
 - ◇ Notifiers
 - ◇ Static Aalyzer
 - ◇ Builders

5.2 Jenkins Plugins - SCM

- Version Control Systems:
 - ◇ Git
 - ◇ Subversion
 - ◇ ClearCase

5.3 Jenkins Plugins – Build and Test

- Build Tools:

- ◇ Ant
- ◇ Maven
- Test Frameworks:
 - ◇ Junit
 - ◇ Selenium

5.4 Jenkins Plugins – Analyzers

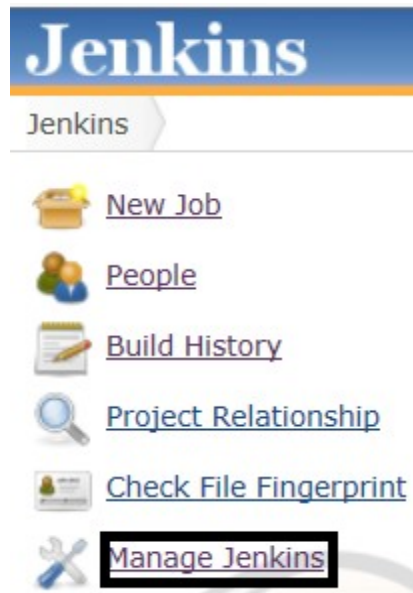
- Static Analyzers:
 - ◇ FindBugs
 - ◇ PMD
 - ◇ Sonar
 - ◇ Fortified
 - ◇ CodeScanner
- Code Coverage:
 - ◇ Emma
 - ◇ Cobertura
 - ◇ Clover

5.5 Jenkins for Teams

- Multi Configuration jobs.
- Multi Stage Jobs.

5.6 Installing Jenkins Plugins

- Installing the Jenkins Emma plugin use for code coverage.
- To access this page, click Manage Jenkins from the left-hand navigation pane.



- Click **Manage Jenkins**
- Then on click the **Manage Plugins** link:

Manage Jenkins



- Select the Available tab:

- For quick access, enter plugin name on the filter input box, located at right top corner.

Filter: ✕

- Scroll down the list of plugins to find the Emma Plugin.
- Select the check box next to Emma Plugin.
- Click **Download now and install after restart**

Updates	Available	Installed	Advanced
Install ↓			
<input checked="" type="checkbox"/>	Emma Plugin This plugin allows you to capture code coverage rep		
<input type="checkbox"/>	JaCoCo Plugin This plugin allows you to capture code coverage rep plugin is fork of the [Emma Plugin]. Big part of the c also includes functionality similar to the [Emma Cov displays the latest overall coverage numbers and lin		
<input type="checkbox"/>	Emma Coverage Column Allows you to add a column that displays line covera		

This will download the Emma plugin for Jenkins. So the download may take some time to complete. Once the installation is complete, you will get the following:

Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Emma Plugin



Downloaded Successfully. Will be activated during the n



[Go back to the top page](#)

(you can start using the installed plugins right away)



Restart Jenkins when installation is complete and no jobs are running

- Select the Restart Jenkins when installation is complete and no jobs are running.
- You have done the installation.

5.7 Summary

- Jenkins Plugins are back bone for Jenkins.
- Different Category of Plugins are available.
- How to install plugin.

Chapter 6 - Distributed Builds with Jenkins

Objectives

Key objectives of this chapter

- Setting up Distributed Builds

6.1 Distributed Builds - Overview

- Jenkins can act as master/slave setup to distribute builds over multiple machines
- Why?
 - ◇ Additional Capacity
 - ◇ Different build/runtime environments
 - ◇ Integration Test
 - ◇ Managing Machines

6.2 Distributed Builds – How?

- ◇ Have another machine
- ◇ Make sure all the required software is installed
- ◇ Version control system
- ◇ Java Development Kit
- ◇ SSH
- ◇ Configure the Jenkins Master
- ◇ If necessary, configure projects

6.3 Slave Machines

- ◇ For *nix, Master contacts slaves by SSH

- ◇ Need user created on slave machine
- ◇ Keys/credentials if necessary
- ◇ For Windows, usually just install slave as a Windows Service
- ◇ Needs to have local copies of anything that Jenkins needs
 - Git, SVN, Mercurial, etc
 - Maven
 - Java
 - Jenkins can install some of these automatically

6.4 Configure Jenkins Master

- ◇ Create a Node
- ◇ Configure Node for # executors, tool locations, etc

6.5 Configure Projects

- ◇ If desired, you can restrict where a project runs
- ◇ Each slave has one or more tags
- ◇ Project can call out a tag
- ◇ Project will only run on slaves where the tags match
- ◇ No other config required

6.6 Conclusion

- ◇ Distributed builds are readily supported in Jenkins

Chapter 7 - Best Practices for Jenkins

Objectives

Key objectives of this chapter

- Best Practices.

7.1 Best Practices

- Always secure Jenkins.
- Backup Jenkins Home regularly.
- Ensure a build is reproducible, the build must be a clean build, which is built fully from Source Code Control.
- All code including third-party jars, build scripts, release notes, etc. must be checked into Source Code Control.
- Always configure your job to generate trend reports and automated testing when running a Java build
- Set up Jenkins on the partition that has the most free disk-space
- Archive unused jobs before removing them.
- Allocate a different port for parallel project builds and avoid scheduling all jobs to start at the 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.
- Set up email notifications mapping to ALL developers in the project, so that everyone on the team has his pulse on the 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.

- Take steps to ensure failures are reported as soon as possible.
 - ◇ For example, it may be appropriate to run a limited set of "sniff tests" before the full suite.
- Write jobs for your maintenance tasks, such as cleanup operations to avoid full disk problems.
- Tag, label, or baseline the code-base after the successful build.
- Configure Jenkins bootstrapped to update your working copy prior to running the build goal/target
- In larger systems, make sure all jobs run on slaves. This ensures that the Jenkins master can scale to support many more jobs than if it had to process build jobs directly as well.
- Make your build self-testing
 - ◇ Run tests as part of the build process
 - ◇ Provide rapid feedback
 - define a test pipeline with distinct test phases
- Automate deployment
 - ◇ Always publish the latest tested build