**Working with Jenkins: Fundamentals**

Top of Form



Bottom of Form

Top of Form



Bottom of Form

|  |
| --- |
| [Play This Course](javascript:var%20warnMsg%20=%20'You%20are%20about%20to%20access%20content%20that%20may%20not%20support%20web%20accessibility.%20Click%20OK%20to%20continue,%20or%20Cancel%20to%20abort.';flagPlayerCourse%20=%20true;launchApplet(lmsSessionTokenId,courseName,%20courseType,winParams,%20use508);disablePlayButton(1,%200);) |

[Overview/Description](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-022169684&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=sd_jenk_a01_it_enus#C1)  
[Target Audience](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-022169684&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=sd_jenk_a01_it_enus#C2)  
[Expected Duration](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-022169684&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=sd_jenk_a01_it_enus#C3)  
[Prerequisites](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-022169684&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=sd_jenk_a01_it_enus#C5)  
[Objectives](https://cognizant.skillport.com/skillportfe/bcs/SinglePassUserCmd.cfm?sessionid=anil.mandava@cognizant.com-022169684&courseaction=launch&x508=0&x508defined=0&restype=0&callfromolsa=1&coursename=sd_jenk_a01_it_enus#C4)

**Overview/Description**

Jenkins is a continuous integration tool that is used to streamline the task of performing application builds and deploys. Jenkins is open source and is Java based. It integrates with traditional Java build tools such as Apache Ant and Apache Maven. Jenkins is widely considered a DevOps tool. In this course, you will learn how to use Jenkins to build and continuously deploy applications. You will also learn about how Jenkins integrates with other DevOps tools.

**Target Audience**

This path is designed for software professionals or anyone interested in learning Jenkins or implementing Jenkins in their organization.

**Expected Duration**

2 Hours, 25 minutes

**Prerequisites**

None

**objectives**

**Course Introduction**

**Overview of Continuous Delivery**

 describe the concept of continuous delivery

**Jenkins as a Continuous Delivery Tool**

 prepare, configure, and set up a deployment of Jenkins

**Building Jobs with Jenkins**

 work with Maven and freestyle job types

**Monitoring Jenkins Jobs**

 work with the Jenkins user interface and recognize how to monitor jobs

**Organizing Jenkins Jobs**

 identify how to organize jobs as the number of jobs grow

**The Jenkins Ecosystem**

 describe how the Jenkins ecosystem works and where to find job examples and plugins

**Working with Git**

 use version control to get code into Jenkins

**Measuring Quality**

 compare the different ways metrics can be used in Jenkins to measure code quality

**Jenkins Automation**

 recognize the different ways to automate Jenkins through the command line interface and other methods

**Access Control**

 work with Jenkins jobs and the background identity service

**Code and Build Promotion**

 work with Java code and the techniques in promoting it

**Parameterized Builds**

 identify how to add parameters into a workflow

**Deployment Automation**

 work with Java and Apache Tomcat to automate Java builds

**Distributed Builds**

 describe how Jenkins can be used to distribute builds to multiple machines

**Different Types of Jenkins Jobs**

 identify the different types of Jenkins jobs

**Build Triggers**

 identify how Jenkins can build jobs based on the occurrence of an event

**Scheduling Build Jobs**

 describe how jobs can be scheduled in Jenkins

**Polling the SCM**

 identify how Jenkins can poll for software changes

**Jenkins Security Overview**

 describe how Jenkins implements security

**Authentication**

 identify how Jenkins authenticates applications

**Authorization**

 identify how Jenkins authorizes users

**Running Jenkins as a Standalone Application**

 use Jenkins as a stand-alone application

**Running Jenkins on an Application Server**

 use Jenkins as a web application

**Installing Jenkins as a Windows Service**

 describe the process for installing Jenkins as a Windows service

**Non-Java Options**

 identify how Jenkins can be used for non-Java applications

**Exercise: Using Jenkins**

**Course Number sd\_jenk\_a01\_it\_enus**

<p>This page contains a JavaScript function that prints out the current frame document. To print out the document without using JavaScript, you need to press Control P.<br /></p>

| [Print](javascript:window.print()) | [Close](javascript:parent.window.close()) |

# **Course Transcript**

# Working with Jenkins: Fundamentals

## **Jenkins Basics**

| [1. Course Introduction](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t2) |

| [2. Overview of Continuous Delivery](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t6) |

| [3. Jenkins as a Continuous Delivery Tool](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t10) |

| [4. Building Jobs with Jenkins](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t14) |

| [5. Monitoring Jenkins Jobs](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t18) |

| [6. Organizing Jenkins Jobs](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t22) |

## **Jenkins Usage**

| [1. The Jenkins Ecosystem](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t27) |

| [2. Working with Git](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t31) |

| [3. Measuring Quality](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t35) |

| [4. Jenkins Automation](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t39) |

## **The Jenkins Environment**

| [1. Access Control](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t44) |

| [2. Code and Build Promotion](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t48) |

| [3. Parameterized Builds](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t52) |

| [4. Deployment Automation](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t56) |

| [5. Distributed Builds](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t60) |

## **Jenkins Jobs and Security**

| [1. Different Types of Jenkins Jobs](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t65) |

| [2. Build Triggers](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t69) |

| [3. Scheduling Build Jobs](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t73) |

| [4. Polling the SCM](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t77) |

| [5. Jenkins Security Overview](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t81) |

| [6. Authentication](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t85) |

| [7. Authorization](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t89) |

## **Jenkins Run Options**

| [1. Running Jenkins as a Standalone Application](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t94) |

| [2. Running Jenkins on an Application Server](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t98) |

| [3. Installing Jenkins as a Windows Service](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t102) |

| [4. Non-Java Options](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t106) |

## **Practice: Working with Jenkins**

| [1. Exercise: Using Jenkins](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#t115) |

# Course Introduction

## Learning Objective

After completing this topic, you should be able to

* *start the course*

## **1.**

Welcome to Jenkins. Jenkins is a continuous integration tool that is used to streamline the task or performing application builds and deploys. Jenkins is open source and is java based. It integrates well with traditional Java build tools, such as Apache Ant and Apache Maven. Jenkins is widely considered as DevOps tool. In this course, you will learn how to use Jenkins to build and continuously deploy applications. You will also learn how Jenkins integrates with other DevOps tools.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Overview of Continuous Delivery

## Learning Objective

After completing this topic, you should be able to

* *describe the concept of continuous delivery*

## **1.**

There are many ways an application can be deployed. As of this day, most code deployments are manual and take a whole bunch of steps. Most traditional application deployment models are slow. They are also error prone as they contain many manual steps. DevOps adds Agile methodologies to the code promotion process. This results in a different way of looking at deploying software. The code promotion and deployment process are accelerating. Code pipelines are now replacing the traditional application-promotion mechanisms. Applications can now be moved into production on a continuous basis. Code deployment to production can now be a daily event. It's important to know the vocabulary describing the application development/deployment process. Sometimes terms can mean different things and can be confusing. First, a pipeline refers to each promotion stage an application takes while it's being developed. Common promotion stages include development, test, and production. Continuous integration refers to code being constantly added to a pipeline as it's being changed. Code is never withheld. It's always in the pipeline. Continuous delivery is the next logical step and code must be production ready. Code here compiles and is bug free. Continuous deployment comes next. Code is production ready and is automatically deployed. This can literally happen with any code change.   
  
Automation is the cornerstone of DevOps code promotion. Basically, it takes the same manual steps and automates them. Almost nothing is performed manually. Manual steps are error prone and slow. Unit testing and acceptance testing are automated and performed instantly. Testing is now automatic. After testing, code is staged for deployment. All this code promotion is also automatic. A variety of DevOps tools are used throughout this promotion process. There is not just one of them, but groups of tools that perform this process. Each tool is unique and does something quite different. Another Neat Tool or ANT is a scripting language used to build Java applications. ANT has been around for a while and allows the automation of Java builds through scripts. When code changes in the code repository, application builds can be triggered. Selenium is an automated testing tool. Testing is becoming more and more automated. Sonar is a tool that measures quality. Quality tools have built-in graphical tools that measure the health of applications. Jenkins is a continuous integration tool – really a glorified pipeline. Of course, Jenkins is the tool we'll be talking about here.   
  
Jenkins works with other tools to continuously integrate and deliver code. Jenkins can be utilized to propagate in serial or parallel pipelines. Core Jenkins is not really that complicated. Most of it consists of a simple scheduling API on the operating system. Jenkins is basically a scheduler on steroids. Jenkins does nothing with the code except build it. What happens next is up to you. Among other things, code builds are performed on a triggering event. Events can be a scheduled time or even a code change in the pipeline. Jenkins uses pipelines to break down builds into smaller, manageable pieces. And all of this is configurable by you.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Jenkins as a Continuous Delivery Tool

## Learning Objective

After completing this topic, you should be able to

* *prepare, configure, and set up a deployment of Jenkins*

## **1.**

What I'm going to show you over the next couple of minutes is I'm going to show you how you can install Jenkins on your local machine. And while we're doing it, we're going to talk about some considerations that you may have to do on your machine as far as configuration and a little bit more on how Jenkins runs on your local machine. Now what you see on my desktop is you see the Jenkins web site. And, if you look on the upper left-hand side, I have the URL in notepad with a larger font in it. So you can see it a little bit more clearly. This is where we're going to install Jenkins from. Now, if you go to the right-hand side and navigate down to where it says download Jenkins...let's talk about the three different ways that Jenkins can be installed on your machine.   
The home page of Jenkins is open in a browser. Running along the top of the home page, different tabs are present, such as Blog, Connect, Bug Tracker, and so on. Below the tabs, a content section is present. The content section has a horizontal section. Below the horizontal section, two vertical sections are present. The horizontal section has a banner. The vertical section to the left has few links, such as Meet Jenkins, Use Jenkins, and Customize Jenkins. The vertical section to the right has two tabs, Release and Long-Term Support Release. By default, Release tab is selected. The Release tab shows a list of native packages, such as Windows, Ubuntu/Debain, Red Hat/Fedora/CentOS, and so on.   
  
Now what you have to understand from the very very beginning is Jenkins is a Java application and it runs in a servlet container. So technically, it's not a Java application. It's a Java server-side Java application or server-side component. So it does run in a servlet container. Now what does that mean? Well, that means that it runs as a web application. If you have an enterprise web application or web server, http server – such as Apache Tomcat – what you can do is you can deploy Jenkins as a WAR file and have the server actually set Jenkins up as a full-fledged web app. You can also run Jenkins as standalone, which means that you're not running an enterprise web server, Apache Tomcat or any of the other large ones and you're running Jenkins really in its own small servlet container that Jenkins actually downloads as part of the install. Also, we can run...if you like you can run Jenkins as a Java Web Start application. So there are three different ways that we can actually run Jenkins. Now, as far as configuration goes, there is not much that you really need to have on your local machine for this to actually work. Or you do need Java. So you're going to have to have Java support on your machine. Other than that, there is really not that much – which is good news. Because the first requirement is, you have a computer. The second requirement is – is your computer on? So there is not a lot to do. In all seriousness, there is not a lot to do to get Jenkins up and going. It's really designed to work easily and have a simple install, a simple interface, and to run with minimum amount of configuration and effort.   
  
Now, on your right-hand side, you should see native packages. Jenkins runs on about everything. Windows, UNIX, Linux, Macintosh – it runs on it. I'm sure I'm going to get an e-mail, and you're going to mention, "Hey Bob, I got this machine and it doesn't run on it." Well it doesn't run on everything. But most major operating systems and platforms will run Jenkins just fine. So what I'm going to do next is I'm going to download the Windows native package. And the Windows native package will install Jenkins as a Windows service on my local machine and actually start it within its own servlet container. It doesn't do anything, and so far is installing it as part of a full-blown web app in Apache Tomcat or any other web server. But it does download and install Jenkins as a Windows service, sets it up as a Windows service, and actually does the actual install. So, what I did is when you weren't looking, I actually downloaded this file, and I exploded it and unzipped it into a local folder. So let me move my browser here to the right – a little bit – and on the left-hand side, here is, this is the setup program that actually was downloaded from the Jenkins web site. Now it should take about may be four to five minutes to download depending on your bandwidth. But at this point, it's just a simple...is giving us a click and going through the Windows setup routine.   
The presenter restores down the browser to the right side of the window to show the Jenkins setup file. The presenter has downloaded the Jenkins setup file beforehand and stored it in the JenkinsInstall folder. The JenkinsInstall folder is stored on the C drive.   
  
So I'm going to give setup.exe a click and we're going to go through the Windows installer. Now, in this case, we're installing Jenkins 1.6. So we click on Next. We will leave the default folder in which we are installing Jenkins. We will click Install. We'll wait for the install to complete. It should only take 90 seconds or so. It's moving the installation files to my hard drive and it is also setting Jenkins up as a Windows service. So, as soon as I click on Finish...I'll make my web browser maximized and you should see Jenkins is running on port 8080. And you should see "Welcome to Jenkins!" Congratulations, you've just installed Jenkins on your local machine. It's really this simple. Now configuring Jenkins is a little bit more of a chore where we're going to have to start things such as put in configurations for users and projects and things of that nature. But we went over the first step, which is installing Jenkins. And while we're installing Jenkins, we went over any configuration issues and things that we need to have working on our local machine. And as you can see, Jenkins is running successfully.   
The presenter double-clicks the JenkinsInstall setup file to install Jenkins. The Jenkins 1.620 Setup welcome wizard with the Back, Next, and Cancel buttons appears on the screen. The presenter clicks Next to set the destination folder, where he wants to install the Jenkins. The page includes change button, which is associated with a text field. The text field shows the default folder location for the setup files of Jenkins. He clicks Next to get ready for installing the Jenkins. The Next button replaces with Install button and the page shows a message, "Click install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard." The presenter clicks Install to install the Jenkins. After few seconds, the wizard displays a message, "Click Finish button to exit the Setup Wizard." The presenter closes the wizard and switches to the browser. A new tab opens in the browser. The Jenkins dashboard appears in the browser. The dashboard is split into two sections. The section to the left has a navigation pane. The section to the right has a content pane, which shows the content of the selected node in the navigation pane. The navigation pane has different nodes and subnodes such as New Item, People, Build History, Manage Jenkins, Credentials, Build Queue, and Build Executor Status. The right section shows a welcome message and a "create new jobs" link. The welcome states, "Welcome to Jenkins!"

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Building Jobs with Jenkins

## Learning Objective

After completing this topic, you should be able to

* *work with Maven and freestyle job types*

## **1.**

What I'm going to cover for you over the next few minutes is I'm going to show you how you can build and run a Jenkins job. We're going to concentrate on free-form jobs and discuss the different types of job types that are out there and how to actually get Jenkins jobs up and going. This is what you see on my desktop. I have Jenkins up and running in my web browser. I have it installed as a Windows service. As you can see, it's running on port 8080. And also towards the bottom, here is I have an example of a build script. Now I built this build script in a text editor and it is on my local machine. And I named this script JavaJobBuildScript.bat. It's simply a build script that builds Java applications. And, in the build script, I just have one javac command. So basically, what this build script does is it compiles one Java program, which I call here MyJavaJob.java.   
The JavaJobBuildScript.bat file is open in Notepad and the Jenkins dashboard is open in a browser. The JavaJobBuildScript.bat file includes "java c:\example\_java\_project\MyJavaJob.java" command line that will build the MyJavaJob java application. The right section in the browser displays a table that provides information about a project, TestJob. For example, when did the job run successfully for the last time, when the job failed, what was the last duration, and so on. Below the table, in the right corner four links are present: Legend, RSS for all, RSS for failures, and RSS for just latest builds.   
  
Now what does this Java program do? MyJavaJob.java...it doesn't matter. What matters is I have a build script and that build script compiles Java components. And, in this case, I just have one component here that I want to build. Now in the world without Jenkins, what I can do is I could run this script from the command line. If you have a Java background – and presumably, you do because you're in a Jenkins class – you've done this before where you go to the command line and you run a batch file that actually performs your build. Now you could run different types of files. You can run shell scripts, you can run ANT files, you can run Maven, you can run a variety of different scripts that actually build your Java stuff. And I'm being vague on purpose because what it builds is really up to you. Jenkins doesn't change any of this. What Jenkins does is it automates and schedules your build process. So again, what I have here is I have a script called JavaJobBuildScript.bat and what this does is it builds my Java stuff. In this case, it just has one Java program in it. A more on this in a little bit. So, anyway, in a world without Jenkins we know how to build this. Now with Jenkins, there is a couple of things you need to do. First and foremost, you need to tell Jenkins where your JDK is. So I'm going to click on Manage Jenkins and then click on Configure System. Now I've already done this but you may have to do this from scratch. But if you navigate down, you should see JDK. I'm going to click on JDK installations.   
The presenter switches to the browser to open the settings related to Manage Jenkins. He selects the Manage Jenkins node in the navigation pane and its associated options appear in the right section, such as Configure System, Configure Global Security, Reload Configuration from Disk, Manage Plugins, System Log, and so on. The presenter clicks the Configure System option to view and modify advanced settings associated with it. Now the right section shows advanced options of Configure System such as Home directory, System Message, Labels, and Usage. Below these options, different sections are shown, such as Global Properties, Maven Configuration, JDK, Ant, Maven, Maven Project Configuration, and so on. At the bottom of the right section, the Save and Apply buttons are present. The JDK section includes the JDK installations button. The presenter clicks that button to show advanced options for JDK installation, such as Name, Java\_HOME, Add JDK, and Delete JDK.   
  
Now I've already supplied this. If you have a new Jenkins installation, you have not supplied this. What you need to do is you need to tell Jenkins where to find your JDK. Here is a hint, "Don't rely on the Java home variable on your local machine and don't rely on the class path of your local machine." This stuff should be set up in Jenkins. So what I did is I have currently on my computer here a JDK found in this path. So what Jenkins is doing is it's setting it as JAVA\_HOME for the purpose of Jenkins. And I get to actually name this. And I name this Java 8 compiler. Now you can also download a JDK, if you click on Add JDK. And I'm not going to do all of this but what you can do is, if you don't have one local you want to download another one, Jenkins will allow you to do so. And I'll let you to do that on your own if you feeling client to do that. But we need to specify at least one JDK and we did so here. Okay, so now with Jenkins having a JDK, we can specify a project. I will leave the page. And I will call this Jenkins Example. And it could be a Freestyle project, which is the most common, which means it's about anything. Also, it could be Maven, External, Multi-configuration or you can Copy existing Item. Now all of these work slightly differently. But why you get one of these recipes down? While you get one of these recipes down, you'll be able to apply what you've learnt to the rest.   
The presenter clicks the New Item node in the navigation pane, and a message box with Leave Page and Stay on Page buttons pops up. The message asks to confirm that he wants to leave the page without saving the changes or not. He leaves the page, and the right section shows the Item name and Copy from text fields and five radio buttons. The radio buttons provide five options: Freestyle project, Maven project, External Job, Multi-configuration project, and Copy existing Item. The Copy from text field is located below the Copy existing Item radio button. In addition, below the Copy from text field, the OK button is present.   
  
So I'm going to click on Freestyle and then click OK. And for the Description, I will say First Example. You could put anything you like. And we have some parameters here. Discard Old Builds, build is parameterized – if we actually pass some things into the build. We're going to keep things as simple as possible for here. For Source Code Management, we're going to leave this as simple as possible. We're not going to use Source Code Management for our first example. For Build Triggers, we won't use these either for our first example. We're going to keep things as simple as possible. And we need to add one thing here, which is the Add build step. Now this is important because these are the four things or the four actions that you can choose from to do your build. Execute Windows batch command, which is what we're going to do. It could be a shell script, it could be an ANT script, it could be a top-level Maven target. If you use a top-level Maven target, you're going to have to use some of the Maven interfaces. And here is the deal with this and this is...what makes Jenkins so cool is Jenkins doesn't change anything above these four different ways that we could build Java applications. Jenkins is a scheduler – that's all it is. If you know how to use ANT, Jenkins doesn't change that. If you don't know how to use ANT, you'll have to learn ANT if you want to use it. Remember Jenkins is a completely different technology that works with these technologies. It doesn't reinvent them and it doesn't really change them. So, if you already have ANT scripts out there, you can get Jenkins actually run them.   
The presenter creates a freestyle "Jenkins Example" project. Then he clicks OK and the navigation pane shows new nodes: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The right section shows advanced configuration settings related to the Jenkins Example project. All the settings are categorized into different sections, such as Advanced Project Options, Source Code Management, Build Triggers, Build, and Post-Build Actions. Above these sections, one more section is present that includes the Project name and description text fields, and four checkboxes. The four checkboxes provides options to discard old builds, parameterize builds, disable builds, and execute concurrent builds, if necessary. The bottom of the right section has the Save and Apply buttons. The Advanced Project Options section includes Advanced button. The Source Code Management section contains four radio buttons: None, CVS, CVS Projectset, and Subversion. By default, the None radio button is selected. The Build Triggers section comprises three checkboxes: Build after other projects as built, Build periodically, and Poll SCM. By default, none of them is selected. The Build section has the Add build step drop-down box. The Post-Build Actions section has the Add post-build action drop-down box.   
  
So, here we're just going to choose Execute Windows batch command. And we're going to say call and we'll put in the path of our batch file c:\example\_java\_project\ and the name of our batch file is JavaJobBuildScript.bat – call c:\example\_java\_project\JavaJobBuildScript.bat. This is our batch file, which I showed you just a minute ago, which will compile our Java application.   
The presenter selects the Execute Windows batch command option and the Command rich text field appears.   
  
So let me take a quick look at this to make sure I don't have any typos and I look like I'm good. So I'll click on Save. Now I've just created my first Jenkins project. So now let me navigate to dashboard and you can see the projects I have here. Now, when you were not watching, I had another project that I built here. But let's look at the one I just did here, which is Jenkins Example. It shows Jenkins Example has not run yet. Last Success, Last Failure, Last Duration is not available, which means that nothing has happened yet. So we haven't scheduled it, haven't done much with it. Now, to actually run that project, I could do two things. I could click on the run icon and it will actually schedule a build and you'll see on the lower left-hand side that Jenkins Example is being scheduled. And we'll give it a second. And if you look at to the left, you see the sun, which is basically the health of the actual project. The cloudier it gets, the more poor the health.   
The presenter adds a command and clicks the Save button to create the Project Jenkins Example project. Now the right section shows the name of the project and the Workspace and Recent Changes links.   
  
But right now, it's sunny because we haven't run this yet. So the health is assumed to be good. And also to the left, you're going to see a sphere with a color in it. That describes what the last build did if it was successful or not successful. And we're still waiting for it to run. And we can, kind of, force it if you want to click on the job or click on this little drop-down here and you could look at things such as any Changes, Workspace. And this is another way we could build this. Or we can also delete the project if we like or Configure. But I'm going to give this a double-click and you'll see the last build happened 1 minute and 12 seconds ago. If I give it a click, I can actually go to Console Output and it will show me the status of the build. Now, if you look at the Console Output, the build was successful. Now, if I navigate back to my Jenkins dashboard, you should see that the last successful build was 1 minute and 42 seconds ago. And my status is Success. So I was able to build a Jenkins project. I was able to schedule the Jenkins project. And I was able to successfully run and build the Jenkins project.   
The presenter clicks the Back to Dashboard node in the navigation pane and the right section shows a table including details of two projects, Jenkins Example and TestJob. The presenter double-clicks the Jenkins Example project name and the right section shows the Workspace, Recent Changes, and permalinks links. The permalinks includes last build, last stable build, and last successful build links. The presenter clicks the last build link and all the nodes in the navigation pane replaces with the new links, such as Back to Project, Changes, Console Output, Edit Build Information, and Delete Build. The right section shows the information about the Build, for example, no changes has been made in the project and an anonymous user starts it. He expands Console Output, which has the "View as plain text" subnode. Now the right section shows Console Output. Console Output shows that an anonymous user starts the project and the project built successfully.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Monitoring Jenkins Jobs

## Learning Objective

After completing this topic, you should be able to

* *work with the Jenkins user interface and recognize how to monitor jobs*

## **1.**

What I am going to do for you in the next couple of minutes is I am going to show you how you could monitor Jenkins jobs and look at some of the statuses. And learn a little bit more about how you can manage Jenkins jobs and how you can, kind of, look at kind of what's going on here. Now what you see on my desktop is you see Jenkins. And I am at the Dashboard. So you could see that I have two Jenkins projects, one called Jenkins Example and one called TestJob. So let's look at these in a little bit more detail. If you look at Jenkins Example and you look at the column's Last Success, Last Failure, and Last Duration. Last Success shows me when the last successful build for this project was performed. Last Failure shows when the job last failed. And Last Duration shows how long the last or the previous build took to actually run. And the same with TestJob, you have the Last Success, which was 39 minutes ago, Last Failure, which is 41 minutes ago, and it takes at least the Last Duration was 1.8 seconds.   
The Jenkins dashboard is open in a browser. The right section shows a table, which includes the information about the projects such as Jenkins Example and TestJob.   
  
Now, kind of more importantly, if you now look at Last Success and Last Failure is you see a number here. This is the number of the build. Now this is a fairly recent Jenkins installation that I am working with. So these numbers are kind of small because I have only done a couple of builds in each of these. But these number will get larger, of course, as we're building more of these projects and as the number of these projects actually grow. But, if you want to see more about a specific job and the last time it was run, you can simply do a couple of things. You can click on the number and, if you click on the drop-down, you can look for the Console Output for that build, you can Edit Build Information, you can Delete Build matrix, or you can change some stuff such as the descriptions. I am going to navigate to Console Output. So we'll see the Console Output for the second build for TestJob. And here it shows who started it. I don't have a current login right now, so I am anonymous. And it shows exactly what the Console Output would look like in this case it was successful. Now, in our example here, I am using a fairly simple Windows batch script to do this. Now in reality, it might be a little bit more complicated, but I am just trying to show you the basics of getting this done without throwing in a bunch of loops that might be a little confusing.   
The presenter clicks the drop-down arrow next to the TestJob project name, a drop-down appears. The drop-down includes different options, such as Changes, Console Output, Edit Build Information, and Delete Build. The presenter selects the Console Output option and Console Output opens in the right section. Console Output shows that an anonymous user starts the project and the project built successfully.   
  
Now I am little more about this. I am going Back to Project. And, when you go back to project, you get the project name here – Project TestJob. And, if you look down under Permalinks, you get the full history of this project. You get the last unsuccessful build, the last failed build, the last successful, the last stable, and all of this. And, if you go even further to the left, you have your Build History. And your Build History again goes without saying, "That is the history of building this specific project." You can also delete the project, you can rebuild it if you like. Right now, we don't have any of these jobs scheduled or these projects actually scheduled, so you can build now if you like. You can Configure the project or what you can do is you could go back to the Dashboard. Now again I want to emphasize that Jenkins is really just a build tool. Notice we didn't look at any code – you never will. All you're going to do is look at these projects and look at when they were build last and when they were successful and when they actually failed.   
The presenter clicks the Back to Project node in the navigation pane and the right section shows the name of the project, that is Project TestJob, and Permalinks, such as Last build and Last stable build. Now he clicks the Back to Dashboard node to return to the Jenkins dashboard.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Organizing Jenkins Jobs

## Learning Objective

After completing this topic, you should be able to

* *identify how to organize jobs as the number of jobs grow*

## **1.**

Hello everybody. Welcome to my demonstration! What I am going to cover over the next few minutes is how to manage Jenkins jobs. What you see in front of you, on my desktop, is you see Jenkins running. And I am at my Dashboard. So what we are going to do is we are going to go over some techniques in managing Jenkins jobs. Now first and foremost, you see all of the jobs on your Dashboard. And here I have two of them. I have a job called Jenkins Example and I have one name TestJob. Now here are some things that we could do with these jobs to manage them. First and foremost, you can take a look if there are any jobs that are currently weighing to be build. And you can see them in the Build Queue. Right now, we don't have any but you would see all the jobs here that were currently in line to be actually built – you would see here. And at that point, what you can do is you can actually sort them, you can move them around, or you could actually delete them so they wouldn't actually get build. Also, the same with the Build Executor Status – for what we can do is while they are waiting is we could specify which ones get priority. Let's talk about the jobs themselves. Let me do a couple of things here. Let me click on Jenkins Example.   
The Jenkins dashboard is open in a browser. The navigation pane in the left section includes Build Queue and Build Executor Status nodes. The right section shows a project table, which includes the information about the projects such as Jenkins Example and TestJob. By default, there is no build in the queue and status of the Build Executor for both of the projects is idle. The presenter double-clicks the Jenkins Example project name in the table to manage Workspace and Recent Changes. The right section shows the Workspace, Recent Changes, and Permalinks.   
  
Now couple of things here, we have somethings that we can actually manage like, for example, the Workspace that the job lives in. I can click on Workspace and I can change the Workspace, which is the current folder and the local machine, which this job actually runs. I can change it if I like or I could just leave it by default, which is underneath the Jenkins folder or a Jenkins folder. If you look to the left, we can do things such as Wipe Out Current Workspace, which means we could just completely wipe out the workspace where this job executes in and create another one. We can do things such as Delete Project, which we will do in a second. And we can also Configure. Like, if I click on Configure the project, it's going to bring me to the configuration screen. Now this is the same screen that you get when you first actually create the job. But what we can do is we can change things such as the Project name, the Description. We can change some of these options such as if we want to pass parameters into it, if we want to Discard Old Builds. Also, if we choose to add Source Code Management – right now, we don't have any – we can change it here in the Configure page. And also very important one – Build Triggers – where we can actually specify that this specific job gets filed when an event happens – when the projects are built or periodically. And also, what we can do is we can change the command. Here we have a Windows batch command that actually runs this job – we can change it here. Also, what we can do is we can do the following. Let me go to Build History and click on one of these guys.   
The presenter clicks the Workspace to show files and folder included in it. The navigation pane shows a subnode under the Workspace node, that is "Wipe out Current Workspace." The right section shows that currently no file is present in the workspace of Jenkins Example on master. Next, to configure the project, he clicks the Configure node in the navigation pane. Now the right section shows the advanced options for configuring the project. All the configuration options are categorized into different groups such as Advanced Project Options, Source Code Management, Build Triggers, Build, and Post-build action. The Build category includes the Execute Windows batch command rich text field, which includes a batch command, "call c:\example\_java\_project\JavaJobBuildScript.bat". At the bottom of the right section, the Save and Apply buttons are present. The presenter refers to the Build node in the navigation pane to show build history of the project. The Build node has two subnodes, "#2" and "#1". Each of the build is followed by their timestamp, on which they were created. The presenter clicks the "#2" to show details of the build in the right section.   
  
So here, you got specific build history for a job. Now what you can do is you can Edit Build Information, you can Delete Build. For example, if this job is been out there for weeks, months, or years, you can go out there and delete some of those old builds. You can navigate to Previous Build. Let me go back to the Dashboard. But before I go to the Dashboard, let me show you right to the right here versus Jenkins Example. If you click on the drop-down here, you can have these icons where you can change the job definition, you can modify change to Workspace, you can Build Now, you can Configure, you can Delete Project. But let me go back to the Dashboard and show you something new here. Let me navigate to Build History. If you want to manage and to view the build history for all of your jobs – as you can give this a quick click – and you would see here in this graph...and again I don't have a lot of builds in here but you would see by date and by job. And you would see the job number, the build number. You would have the job name, the build number. And, if you scroll down, you get some more stuff that you could manage. It shows again which ones were successful, which ones were not. And you get like a nice Status here. If you want to manage these specific jobs and builds and also, if you want to manage wide field – you can always click on here – which you would see the console messages.   
The right section shows that an anonymous user started the Build #2 and no changes have been made to it. Above the left section, the path of current node is displayed, that is, Jenkins - Jenkins Example - #2. The presenter hovers the mouse over the Jenkins Example in the path and a drop-down arrow appears. He clicks the drop-down arrow and a drop-down menu appears. The drop-down menu includes different options such as Changes, Workspace, Build Now, Delete Project, and Configure. Then he clicks the Jenkins in the displayed path to return to the Jenkins dashboard. The presenter clicks the Build History node in the navigation pane and the right section shows the Build History of Jenkins in the graphical and tabular format. The graphical representation shows the name of the job and its build number on the date and hour. The same graphical data is displayed in the tabular format as well. The table provide the build number, time of the build, and status of the build. For example, the Jenkins Example #2 is created 57 minutes earlier, and after its build, the project is in a broken status.   
  
Now let me go back to the Dashboard and look at some of these jobs here – these different projects. Now I am not going to Delete Project but I can. But I am going to leave it one here because I am going to need it for another example. You can always delete these projects and readd them. What I want to point out is when you delete a project, you are really not deleting anything as far as your Java code is concerned, you are not deleting any Java code, you are not deleting any Java applications, you are not deleting any batch file, scripts, or anything. What you are doing is you're simply just deleting the Jenkins job that does the builds. So feel free to delete a project and readd them if you like. You are not really deleting anything permanently as far as Java goes on your hard drive.   
The presenter goes back to the Jenkins dashboard.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# The Jenkins Ecosystem

## Learning Objective

After completing this topic, you should be able to

* *describe how the Jenkins ecosystem works and where to find job examples and plugins*

## **1.**

Jenkins has a growing and vibrant ecosystem. In this ecosystem, you can find friends, training, and code. There are small local users groups and larger national organizations. Jenkins users groups hold conferences and training events all over the world. These conferences offer networking, vendor events, and other cool activities. Jenkins is also in public source control. The source code lives on GitHub and allows user forks. The Jenkins ecosystem also has hundreds of useful plugins that are available. Open source projects can also be found. Dozens of open source projects use Jenkins. Many are very interesting and would welcome your involvement. The Jenkins User Conference or the JUC travels all around the globe. This is the national users group and has its annual conference in United States, England, and in Israel. Internet events are held as well. Online meetings called Office Hours are held regularly. If you like to meet or collaborate online, various online groups and mailing lists are available. Jenkins has partnered with a training partner named CloudBees. CloudBees has helped grow the Jenkins user community. Online resources are also available via Jenkins. Jenkins users create and post free tutorials on the Jenkins website.   
  
Jenkins encourages your technical contributions. You can actually be one of Jenkins authors. The requirements are low to join the Jenkins project. The Jenkins project takes pride in the fact that it has an open door policy to developers and ideas. The Jenkins project consists of a core Jenkins group and a plugin group. You can belong to one or both groups. The Jenkins project folks will also allow you to push Jenkins code into Git. They give away push access very easily. The contribution process is generally self-policed – first by the Jenkins group, then by the executive committee. The real power of Jenkins is its plugins. Core Jenkins does surprisingly little. Plugins are available for all types of functionality. Pipeline Plugins create and automate code promotion. These plugins replace manual code promotion. Source code management plugins are used for code repository management. Popular plugins are available for GitHub and Bitbucket. Monitoring plugin generate eye pleasing performance reports. These are available by many users and vendors. Build trigger plugins are used to define build events. Some build trigger plugins come with Jenkins, others have to be downloaded. Most are free and can be downloaded and installed directly from Jenkins.   
  
There are all kinds of open source projects that use Jenkins in some capacity. Icinga uses Jenkins within its enterprise-monitoring tool. Jenkins is used to automate builds. Mozilla is another open source project that uses Jenkins. Here Jenkins is used as a pipeline tool. JBoss Continuous Integration Monitor also includes Jenkins. JBoss uses Jenkins to trigger builds based on code changes. The Pentaho continuous integration server is powered by Jenkins. Jenkins offers Pentaho stability through automating its build process. JRuby also runs Jenkins. Jenkins has dozens of large known open source projects and perhaps dozens of small ones. Jenkins has proven itself as a robust and reliable tool in the open source community.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Working with Git

## Learning Objective

After completing this topic, you should be able to

* *use version control to get code into Jenkins*

## **1.**

What I am going to demo for you over the next couple of minutes is I am going to show you, how you can extend Jenkins to work with other source code management systems, such as Git. And while we are doing that, I am going to discuss a kind of in general how plugins work within Jenkins. So let's get started, what I am going to do to start out is I am going to pick one of our existing Jenkins projects. So let me choose dropsofjupiter and click on Configure. Now, if I scroll down, in this project I see an entry for Source Code Management. Now it's not a coincidence that you see these options here on my desktop. For Source Code Management, I have None, I have CVS Projectset, and Subversion. Now the reason I have these here is because, out of the box, Jenkins came with these plugins already installed.   
The Jenkins dashboard is open in a browser. The navigation pane in the left section includes different nodes, such as New Item, People, Build History, Manage Jenkins, Credentials, Build Queue, and Build Executor Status. The right section includes a table that lists the information about the projects such as Deploy our web application, dropsofjupiter, Jenkins Example, Promote from Dev to Test, and TestJob. The information includes the name of the project, last success and failure time, last duration, and the status of the project. The presenter clicks the dropsofjupiter project in the table and its information appears in the right section. Then he clicks the Configure node in the navigation pane to modify the settings for the dropsofjupiter project.   
  
Let me talk about plugins, just in general, code Jenkins is very minimum as far as what it includes. It really includes just a bare basics in interfacing with the operating system and doing some scheduling. The real power of Jenkins is the use of plugin. Now, out of the box, Jenkins just so happened to have CVS, CVS Projectset, and Subversion. Those plugins are already installed, so you see options to configure source code management here. If we had other plugins, presumably in this example with other source code management system, you would see options here for them to be configured. So let's talk more about that, and we scroll up and navigate to Back to Dashboard. And from here, Manage Jenkins and you will see all of the links that we can use here to manage Jenkins. Now, if you look the fourth link down, we have a link to Manage Plugins. Plugins are everything in Jenkins. So let me give this link a click, and you will notice a couple of things. First one I want you to do is I want you to click on Installed. Now here you are going to have a list of all of your plugins that are already installed.   
For the dropsofjupiter projects, no description is mentioned. The Source Code Management is set to none and the Build Triggers is set to Poll SCM. The presenter goes back to the Jenkins dashboard and clicks the Manage Jenkins node in the navigation pane to show all the links to manage Jenkins. The right section shows all those links, such as Configure System, Configure Global Security, and Manage Plugins. He clicks the Manage Plugins link and four tabbed pages are displayed in the right section. In the top-right corner of the right section, the Filter field is present. In the bottom, three buttons are present: Download now and install after restart and Check now. The four tabbed pages are Updates, Available, Installed, and Advanced. By default, the Updates tabbed page is selected, which shows the list of plugins that a user can install. The Install checkbox is associated with each plugin box and none of them is selected. The presenter selects the Installed tabbed page, which shows the list of plugins that has been installed in the system. In front of all the plugin names, a checkbox is present that allows a user to enable or disable a plugin by selecting or deselecting the checkbox. Some of the installed plugins are Ant Plugins and Credentials Plugin Updates.   
  
Now how did these plugins get installed? Well, either you install them or when you install Jenkins, Jenkins already installed these. Jenkins does a lot when you do the initial installation. A lot of what it does is it takes some of the most existing plugins and does the install for you. Sometimes you look at the stuff and you think its core Jenkins functionality, but it really isn't. It just these plugins were actually installed when you installed Jenkins for the first time. Now, if you go to Updates, it will show you all of the plugin that you have installed. That currently have new versions for you to download. And you can see here what the installed versions is and what the current version is. I am going to click on Installed again and notice that as far as source code is concerned CVS is selected, JUnit is selected. And, if we scroll down even further, we will notice that Subversion is selected, and some other stuff such as Maven is selected. So that's why you see those options for source control when you're configuring a project. It's because these plugins are already installed.   
  
So I am moving to go over to Available and in Filter, I am going to type git. Now what it shown to me is all of the plugins that have Git as string match. Now obviously, the context might be different where, you know, we are looking for a Git – the source control management systems such as GitHub as a plugin that we can actually install. Now here is the thing with these plugins, there's so many of them. I am just getting you started and the basics of plugins and here with source control plugins more specifically with Git plugin. So you are going to have to do some research to determine exactly the plugin that you want to use. And once you figured that out, you can download it, install it, and you'll see those options available to you when you actually configure a Jenkins project. If you are not using Git, for example, I am going to type in bitbucket. This shows me all of the plugins that are available if I use bitbucket for my source control management systems. So it really depends on your situation with source code management, what are you using, what do you want your plugin to do. Remember we live in a world where we have plugins that are so customized. You get to choose the one that fits your needs the best. And you certainly see that when you're using Git for source control management.   
The presenter switches to the Available tabbed page, which also includes a list of plugins available for installation. In front of all the plugin names, a checkbox is present that allows a user to install the plugin. The presenter types the word "git" in the Filter field to list all the available plugins related to the word "git," such as Github Authentication plugin and Gitcolony Build Notification plugin. Now he types the "bitbucket" in the Filter field to list all the available plugins related to the word "bitbucket," such as Bitbucket Approve plugin and ShinningPanda plugin.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Measuring Quality

## Learning Objective

After completing this topic, you should be able to

* *compare the different ways metrics can be used in Jenkins to measure code quality*

## **1.**

What I am going to show you over the next couple of minutes is I am going to show you some simple ways that you could monitor the health of your Jenkins jobs. What you see here on my desktop is I have Jenkins running and I am at my Jenkins dashboard. Now let's get right to it. Let me show you two metrics that at a glance will show us the health of our Jenkins job. The first is, if you navigate over to the S, you should see a bubble help pop-up status of the last build. This is one metric this will show you at a glance what happened in the last build. Now, if you're blue, you're good that means that the last build was successful. Now red is bad that means that the last build has failed. Now to the right of your status is you have your weather report. Now your weather report shows the aggregate or the aggregated status of all the builds for this specific project.   
The Jenkins dashboard is open in a browser. The table in the right section shows information about the Jenkins Example and TestJob projects. The table provides the status and weather of the listed project names. For example, the build of TestJob project is successful and the build of Jenkins Example project failed.   
  
Now, to look at the Legend for this, let me just kind of show you what these icons mean. But before I do that, if you navigate your mouse over the weather report icon, it will tell you – on your right-hand side there – the percentage of builds that have failed. So here, it shows that my build stability – one out of the last two builds failed. And, if I go to my second one – which is TestJob – I get the same. And let me show you a little bit more about these icons. You can always go to Legend. And I am not going to read all these to you because you can pause the video and look at them. But, if I click on Legend, this shows basically the status and also it shows the weather report. So I showed you this one, "The last build was successful." And you could see this one here, "The last build fatally failed." Now you can also see, you know, the different colors and what blinking means. And again, I'm not going to read them to you because you can do that yourself. But you do know that your status will show you what currently is happening to their project and what has currently has just happened to that project. Now moving on navigating down here are the icons for your project health. At a glance we start off sunny, which means that it is 80% healthy that means that 8 out of 10 or 80% of your last builds have been successful. And then your weather increasingly gets cloudy. And, so we got some horrible thunderstorms here, which shows that your project health is 20% or less.   
In the table, for the Jenkins Example project, weather report shows that one out of two builds failed. For the TestJob project, weather report shows that one out of two builds failed. The presenter clicks the Legend link below the table to show the status and health of the projects.   
  
So let me scroll back up and go back to the dashboard and show you a couple more things about the health here. Now we have the status. And we have the weather report. We can also look at trans. For example, I am going to navigate to Jenkins Example, which has failed 4 min 34 sec -#2 ago. And I could look to see possibly the trend has it bend feeling, has it just recently failed? What is going on with this specific job? Also, "Last Duration" can tell me a lot about this as well – where you might look at the duration of these builds and some of them might be hanging and maybe very long duration builds. But, I'm going to look at trans, so I'll click on Jenkins Example. And I only have two builds here but this will show me the trend and I can also sort on that as well. Now I only have two builds here, but I could have hundreds like I could look at this and I could see exactly when the build started to fail. So, I could go, look for developer, I can look for some operations personnel, or I can investigate further – and why this build is failing? But it will show me the trend for the specific project and give me some insight. And, when the build started to fail. Also, if I look in Permalinks, it'll show me all of the details and the builds. So perhaps I could look at the last build that failed and I could click on this one here, which is "Last failed build (#2). 5 min 25 sec ago". And I could look at perhaps the Console Output. And it will tell me why this build failed or at least console messages.   
The presenter goes back to the Jenkins dashboard and shows the information about the project such as workspace, recent changes, and permalinks. Then he clicks the last failed build (#2), 5 min 25 sec ago link under the permalink to view the details of the last failed build (#2). To view the details in the Console Output, he clicks the Console Output node in the navigation pane and the output returned in the Console Output is shown in the right section. The output says that an anonymous user starts the Jenkins Example project and its build was failed.   
  
Now what I did is I'm purpose is I put a syntax error in here. I put in rather than MyJavaJob.javan. I just made the syntax incorrect, just so it would fail. So you could see here that this job failed because the syntax was incorrect. See I got invalid flag, Usage. So I could look at this as a system's engineer and say, "The job failed because someone updated the Windows batch file." And it was incorrect so the job actually failed. But, the core part about this is if you look at the big picture at a glance, we can look at all of our Java projects. Again we have two here and really at a glance a kind of have control over all of them. And looked to see what the health is and which ones are working, which ones are not. And it's very, very helpful to look at some of these simple icons to get an at a glance view of how are applications and how are projects are running and the overall health of our system and each individual project.   
The presenter returns to the Jenkins dashboard to give overview of the displayed table.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Jenkins Automation

## Learning Objective

After completing this topic, you should be able to

* *recognize the different ways to automate Jenkins through the command line interface and other methods*

## **1.**

What I am going to show you over the next couple of minutes is I am going to show you how you can extend Jenkins and Jenkins functionalities to work at the command line. Now, as you know, by now, Jenkins is a web application – Jenkins can work as an enterprise, web application, in commercial servers such as Apache Tomcat. It can run local in its own small servlet container and as a Windows service. You can also run Jenkins from the command line via some command line commands via a JAR file. Now what I am going to do is I am going to go into my URL and I am running Jenkins from the localhost. And I am going to put a /cli, which stands for command line interface. Now here what you get to see is you get to see some examples of how you can run Jenkins from the command line.   
The Jenkins dashboard is open in a browser. To open the command line interface of the Jenkins, the presenter modifies the URL. Now the right section shows a list of available commands for the Jenkins CLI, such as build, clear-queue, console, and so on.   
  
Now the JAR file can be downloaded from this link, right here. You are going to need a JAR file called jenkins-cli.jar. Now the good news is if you can't get it off the web and there's no reason you shouldn't be able to get it off the web is you probably already have this JAR file downloaded locally. So here, we have an example of running this JAR file. Now the JAR file being a Java file response to any jar command. So in this example, we have java -jar we specify the JAR file and then the arguments. So here, we would just specifying the URL where you can find the Jenkins server, the port where you can find the Jenkins server, and then you have here help. And we have listed here all of the available commands that you can use from the command line. And it's quite extensive. For example, if you wanted to build a job, I am going to click on the build link. And this is an example of the syntax to start a build and you would specify your specific Jenkins server here with your URL and any command line arguments to the right. And you would be able to start a build via the command line.   
To start a build, the following command should be used:java - jar jenkins-cli.jar -s https://localhost:8080/ build JOB [-c] [-f] [p][-r -N] [-s] [-v] [-w]   
  
Now there is some caveats here you are going to have to do some security administration and security configuration within Jenkins. You probably should have some user IDs actually configured with some matrix space authentication. So this actually works and works as you expected. So depending on if you are running Linux, Windows, or any flavors thereof, you may have to go into Jenkins and then navigate to Manage Jenkins and go into Configure Global Security. And again, a lot of this is going to depend on your situation, your configuration, and how you have firewalls put up. And you may have to put in some user ID authentication in here to be able to run or have other users from around your organization or around the world. Run your Jenkins implementation from your URL at the command line.   
The presenter returns to the Jenkins dashboard and clicks the Manage Jenkins node in the navigation pane. The right section shows the configuration settings related to the Manage Jenkins node. He selects the Configure Global Security link and its respective configuration settings appear in the right section, such as security realm and authorization for access control.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Access Control

## Learning Objective

After completing this topic, you should be able to

* *work with Jenkins jobs and the background identity service*

## **1.**

What I'm going to demonstrate for you over the next couple of minutes is we are going to go over some authentication within Jenkins and some access control within Jenkins. Now let's get started. What you see here on my desktop is I am running Jenkins in my web browser. I'm running it locally as you can see from localhost:8080. Now the first thing I want to do is I want to go and click on the People link. This will show me all of the specific people in the login's that are currently within our security realm. In English that means these are all the user IDs that are configured within our Jenkins environment. Now notice you don't see any. Now this is very important because by default you don't need to have users in Jenkins. Now I am not saying this is a good idea. As a matter of fact, it is probably not a good idea. But, when you first install Jenkins, you're able to login and do something. Because Jenkins does allow nobody being a user of nobody as a user or anonymous as a specific user. So here, you see no one is actually configured to actually work with Jenkins. Now let's go over a little bit more about access control. Now I'm going to navigate down the Credentials. Credentials is where you can actually supply user IDs and passwords. So let me click on Global credentials. And it tells me that I don't have any Global credentials. The credential domain is empty, so let me add some credentials. Now here what I can do is I can specify the kind of security realm I want to actually configure Username with password, SSH Username with private key, and Certificate.   
The Jenkins dashboard is open in a browser. The presenter clicks the People node in the navigation pane and the right section shows an empty table. The table includes the User ID, Name, Last Commit Activity, and On columns. Then he clicks the Credentials node in the navigation pane and the nodes in the left section reduced to the UP, Credentials, and Add domain. The right section shows a table that lists the domain name and its description. The table includes a domain name, Global credentials. Its description states that the Global credentials are all those credentials that are not bound to a specific name. The presenter clicks the Global credentials to show some example of global credentials. Another empty table is displayed. Once the global credential is created, the table will include the name, kind, and description of the global credential. The table includes a link, "adding some credentials." He clicks that link to create a global credential. Now the left section shows Back to credential domains and Add credential nodes. The right section shows the Kind, Scope, Password, and Description fields. The OK button is present between Scope and Password fields. The Kind field has a default value that is Username with password. The Scope field also has a default value that is "Global."   
  
So, if I want to add a username, I specifically can choose Username with password. Scope can change from Global to System, but we probably want to keep Global – we put the user id and we put their password. Now, once we get user IDs and passwords in the system, what we can do is we can specify what they're allowed to do. So the first step is to put Credentials within the system. And let me show you what comes next. Let me go Back to credential domains and let me go to my dashboard and then navigate to Manage Jenkins. So, after you get user IDs and passwords and you specify domains, you can go to Manage Jenkins and then from here you could click on the link for Configure Global Security. Now here what you could do is you could define who is allowed to access and use this system and what they are allowed to do. Now remember anyone can do anything right now in Jenkins. Right now, anyone can login and do anything they want. And that's probably not a good idea. It's probably a good point and important for me to point out here. Well, I am going through this that Jenkins doesn't really have a real secure or comprehensive security model. It's really not meant to...Jenkins runs on a machine and that machine implements its own security model. So Jenkins kind of piggy backs on that. Soon we talk about security. We're really talking about Jenkins security and what the users are allowed to do in Jenkins, which is important. But Jenkins is it known for really having anything else as far as security goes other than the protect itself.   
The presenter clicks the Back to credential domains to return to the previous page. Then he returns to the Jenkins dashboard. Now in the right section, the presenter shows the advanced settings of the Configure Global security. The section includes three checkboxes: Enable security, Prevent Cross Site Request Forgery exploits, and Use browser for metadata download. Below the Enable security checkbox, the Markup Formatter drop-down list box is present. The drop-down list box is set to Plain text. The Enable Security checkbox allows the user to modify security realm and authorization for access control. At the bottom of the right section, the Save and Apply buttons are present.   
  
So I am going to click on Configure Global Security. And by default, actually Enable security is not checked, which means that anyone can login and do anything. I am going to click on Enable security and show you some things – they have to do with the Security Realm and Authorization. At least two different ways that we can configure security. The first is we use Jenkins' own user database. This is for smaller configurations or possibly smaller configurations where you want Jenkins to maintain its own security. We can also look into your enterprise security. For example, if you use LDAP, we can use LDAP for your enterprise security. So we can use the same authentication as far as user IDs and passwords go. Now look for Authorization here. In Authorization, by default again Anyone can do anything. We probably don't want that. So there are other choices for us, such as Legacy mode, Logged-in users can do anything, and we also have Matrix-based security and we also have Project-based Matrix Authorization Strategy. So I am going to click on Matrix-based security and show you how we can set this up. Now we only have one user right now, but we could have different users configured in the system. And we could just go through these checkboxes and choose what they can do.   
The presenter sets the authorization for access control to Matrix-base security.   
  
Now this is based on Update, Delete, Add, and Configure. Based on what the users are allowed to do, what you can do is for each specific user, what they're allowed to do. And, when you're done, click on Save. Right now Anonymous unless I choose things to do will not be allowed to do any of these things. So, for my case, I'm still going to check Enable security off because at this specific time I am not going to configure anyone. I just want my own user ID to be able to do anything in anytime. So to do that again, you can take Enable security and uncheck it or like I mention before you can look at specific users and choose what they are allowed to do. Now again, these user IDs are going to be Jenkins' users, not necessarily your own organizations enterprise network users. On last, you choose LDAP. And then you can use single sign-in with your own user ID, password as far as authentication goes. So let me scroll all the way back up. So we went over how to add a user to the system, how to add their user ID and their password, how to go into Configure Global Security, and went over some Global security options. As far as the Security Realm and Authorization, and showed you how you could go into Matrix-based security and choose what a user is allowed to do – either a Jenkins user if you using the Jenkins' own user database or any of your enterprise users if you're using LDAP security to authenticate network users from within your own organization.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Code and Build Promotion

## Learning Objective

After completing this topic, you should be able to

* *work with Java code and the techniques in promoting it*

## **1.**

What I am going to cover over the next couple of minutes is we're going to talk a little bit about Java application development and a little bit about code promotion. Let's say, for example, that we work in an organization where we have a group of ten Java developers. And, while we're doing our day-to-day Java programming, eventually, while we're working on an application, we get to the point where all of our code has been dev tested, which means all developers have went through their Java programs and went through their Java code and everything basically tests out okay. At least as far as unit testing goes. And then we get to the point where we like, you know, what we need to take this Java code and promote it to a different environment where we can put it into a test environment. So therefore, we can maybe build an application based on that code in the test environment. And, then give that Web application or that Java application to users, so they could do some QA on it, maybe some user acceptance, or some quality assurance testing on our specific application. Now, when we take code and applications and we move them from group to group like that like from a development environment to a test environment and then maybe a QA environment and then eventually a production environment. What we're doing is we're talking about promotion where we're actually promoting the code or promoting our stuff from one environment to the next. Now, when you're talking about code promotion usually in a traditional software development life cycle, you are talking about development, test, user acceptance, and then finally it would go in your production. Now, with dev apps that might change a little bit where we might have our code promotion groups that might kind of maybe navigate around the same, sort of, area rather than going from one group to the next. But anyway, the concept is still valid.   
The Jenkins dashboard is open in a browser.   
  
So what I'm going to show you is I'm going to show you some techniques in Jenkins that we can actually do code promotion and build a code promotion project in Jenkins, which takes Java code and promotes it from one environment to the next. Again, very common where we all develop Java programs until they work. And then they get promoted to test until the testing people say, "They work." Then it gets promoted to somewhere else. So we have kind of a pipeline here where we're constantly, continuously integrating code from one, sort of, environment to the next. So while something is going from dev to test we might have some new stuff going into dev. And, when we have code and applications going from test to UAT, we might have code going from dev to test. So we're constantly, continuously integrating code throughout our pipeline and either testing it either in series or in parallel. So, in Jenkins, what we can do is we can build jobs to actually enforce that pipeline or to actually use that pipeline process. How we use that pipeline is completely up to us – where the designers, where the people that put, this all together.   
  
What you see here on my desktop is you see a web browser running Jenkins in it. So Jenkins is running and on the bottom, you see a very small Windows script. Again, this could be an Ant build script. It could be Maven. It could be shell. It's Windows batch here because it's supported on the system that I am running. And what this script does is it takes all of my code or all of my start files from a folder called dev, which will represent our dev environment and copies it to a folder called test, which will represent our test environment. So this is a very simple code promotion script or a pipeline script that takes code from one environment and puts it into the next. Now, in reality, your script is going to probably be a little bit more complex than this, but it doesn't have to be – the concepts still works either way. So anyway, I got a script called promote.bat. So let's build a Jenkins job that will fire off this script that will promote our application and our code from one environment – development – to the next, which will be test.   
  
So let me go to New Item and I will build a new item called Promote from Dev to Test. It will be a Freestyle project. And I will not give it a description just to keep things moving. I will not do anything with Build Triggers or Source Code Management, although certainly I could. So let me Add build step. And let me Execute Windows batch command where I will do the following call: c:\example\_java \_project and the file is called promote.bat and I can Apply saved. And let me Save it for real and close. Now again I can schedule this if I want or I can associate it with a Build Trigger, but I'm not going to. Let me navigate to my dashboard and we see our new job – Promote from Dev to Test. And notice it's sunny because it has not failed yet, hopefully it will. And we don't have a current status because we have it even done on initial build yet. So let's do a build or run this script. And again an error message. So let's see what that error message is all about. Let me choose my Last Failure, go to Console Output and it could not find my batch file.   
The presenter creates the "Promote from Dev to Test" name freestyle project. To execute the Windows batch command, he goes to the Build section and selects the Execute Windows batch command option to open the Command rich text field. He adds the "call: c:\example\_java\_project\promote.bat" command in the Command rich text field and clicks the Save button. Then he returns to the Jenkins dashboard to check that the name of his new project, that is Promote from Dev to Test, is added in the displayed table or not. The presenter clicks the build icon of the "Promote from Dev to Test" project. The build fails. He clicks the value of last failure of the "Promote from Dev to Test project" and the build details of the project appear. To view detailed information of the failure, the presenter opens the Console Output.   
  
So let's see what the error we have here – could not find the batch label file. So we have C:. Let us look at our spelling c:\example\_java\_ project and we have it called promote.bat and let's see how we have it configured. So we will go Back to Project, we will go to Configure, and we will look to see if we have something perhaps misspelled. And actually, we don't need this colon. So let's do a Save. Again, I just took the colon out, I had one too many. Still a good example because it shows exactly what happens when a batch file actually fails. Let's go back to the dashboard and run this job again and we're successful Promote from Dev to Test. We can look at our last successful build. Look at the Console Output and here we only have only one Java program because I just did not put a whole bunch in there, but it could have been hundreds. And it was successful. So we have a script here that does promotion. This is a pipeline script that takes all of our objects and all of our stuff in our dev environment and promotes it to our test environment.   
The presenter opens the configuration settings of the Promote from Dev to Test project and removes the colon present after the work "call" in the command displayed in the Command rich text field. He again saves the settings. He goes back to the Jenkins dashboard and rebuilds the project. This time the build run successfully. To confirm this, he shows the Console Output.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Parameterized Builds

## Learning Objective

After completing this topic, you should be able to

* *identify how to add parameters into a workflow*

## **1.**

What I am going to show you over the next couple of minutes is I am going to show you how we can get parameters into our job. Basically, we can prompts or just apply them either within Jenkins or as part as the URL. And get values into a...either maybe an Ant scripts or a batch job or a shell script or anything. So what we could do is we could use a variable for really anything we want. We can, maybe, pass in some sort of environment. We could pass in a string. We could pass in an environmental variable. What we pass into our Jenkins job is really up to us. And I am just going to show you how you can do that. Let's modify an existing project that we have. We have a project that is called Promote from Dev to Test. And what we will do is we will make a modification to this project to actually pass a parameter into it. So let me go to Configure. And let me click on this checkbox – This build is parameterized. And looks like I would have my values in here, but we give it a name. And this is going to correspond to the name either in your shell script or your Ant or your batch file. I'll put in – how about this – "FirstName". The Default Value – I'll put "NONE". And for the Description – we'll leave it "Who ran this build". Basically, just a demo with us is I am going to just pass in a string to this specific build. And, within the script, it's just going to echo out – "This build was run by" – and then it's going to take the parameter. And, to show you that it indeed got there.   
The Jenkins dashboard is open in a browser. The right section shows a table that includes information about the projects, such as the Jenkins Example and Promote from Dev to Test projects. The information includes status, weather, name, last success, last failure, last duration, and build option for the project. The presenter clicks the Promote from Dev to Test project to show its detailed information page, which includes Workspace, Recent Changes links, and Permalinks. Permalinks include detailed information about the builds. Then he clicks the Configure node in the navigation pane and its respective advanced settings open in the right section. He checks the "This build is parameterized" checkbox to set name, default value, and description for the Promote from Dev to Test project. He sets the name of the project to FirstName, keeps the default value to none, and description to "Who ran this build."   
  
Let me show you a couple of more things with this – now I am going to click on Add Parameter. Although I am not going to add one, I want you just to see a couple of things here. There are a variety of different parameters that you can pass in almost all the time, maybe not almost all. You are going to use string quite a bit, but these are the ones that are available as well. So let me click on Save. And there we go. All right. So let me go to Build with Parameters. It is going to ask for the FirstName. And I will put in my first name, which is Bob. Click on Build and you can see it's building here. In Build History, it built successfully. So click on build 5. And then, if I navigate to Console Output, you can see that our parameter was passed from Jenkins to our job. So we successfully passed the parameter into our Jenkins job.   
To show different parameters available for the project, the presenter clicks the Add Parameter drop-down list box and different list options are shown, such as Run Parameter, String Parameter, and Text Parameter. Then he saves the modification and returns to the same information page of the project, Promote from Dev to Test project. Now to add parameter with build, he clicks the Build with Parameter node in the navigation pane, and the right section shows a FirstName text field with a Build button. He sets the first name to Bob and the Build history node in the navigation pane shows a subnode, which shows the processing state and time of the build. After few seconds, the build is complete. Then he clicks the build and opens the Console Output to confirm that the parameter, Bob, which he passed with the build, reflects in the Console Output or not.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Deployment Automation

## Learning Objective

After completing this topic, you should be able to

* *work with Java and Apache Tomcat to automate Java builds*

## **1.**

What I am going to cover over the next couple of minutes or so is we are going to talk about how we can automate web applications with Apache Tomcat. We can basically take WAR files and we can propagate them and promote them into production and the fly. Let's take the following scenario. Let say that we have a job – a Jenkins job or any other job – that builds a WAR file and takes the WAR file and then puts it on some file or share somewhere. We may want a situation or have a situation or we may want to actually deploy that web application and a schedule. Like, maybe every hour we deploy the application or at the end of every day or maybe every ten minutes or whatever. Truly up to us because we can use Jenkins to actually do the scheduling for us. So what we are going to do is we are going to go over an example of using Jenkins and Apache together to deploy web applications. Now this is what we have on my desktop. In the background, I have Jenkins running. And, on the lower left-hand side, I have the command line, which we're going to use to look up a file. And on the lower right-hand side is I have a batch file that's going to be used to actually deploy the application.   
Three windows are open on the screen: browser, Command Prompt, and Notepad. In the browser, Jenkins dashboard is open. In the Command Prompt, few lines of commands are displayed. In the Notepad, deploy\_webapp.bat file is open.   
  
Let me go to the command prompt and do a dir and look for all my WAR files. In this folder – in example\_java\_project – I have a WAR file called Calendar.war. And, in this scenario, we have another Jenkins job that produces this WAR file. We, an Ant script, or what have you run a job and it creates a WAR file every hour or so. What we are going to do is we are going to create a job that's going to get this WAR file and deploy it to Apache Tomcat. And, if you look on the right, here I have a batch file that actually does that – where we copy the WAR file from c:\example\_java\_project\Calendar.war and puts it in C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps. And, if you know about Tomcat, any WAR file that gets put in this folder automatically gets deployed as a web application. So there is a reason why we are putting it in that specific folder. Now what is important here is we have a mechanism – Jenkins – that can actually run jobs that would schedule. This deploy and actually takes this WAR file on a regular basis and deploy it to Tomcat, which presumably would be deployed as a live application.   
To look all the files with "war" extension, the presenter executes the "dir \*.war" command in the Command Prompt window. The output returns the 542.057 Calendar.war file and its details such as time stamp and file size. The bat file open in the Notepad window includes the following line: C:\example\_java\_project\Calender.war "C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps".   
  
So let's build a job that's going to do that. Or let's build the project that's going to do that. So let me...just going to maybe copy and paste some stuff here. So I don't have to remember them. Let me go into Jenkins and create New Item. And we will call this guy "Deploy our web application". We will make him a Freestyle project. I'll skip the Description and I'll keep this simple. No Source Code Management. No Build Triggers. And, for the build step, I will Execute Windows batch command. And again, this could be an Ant script, it could be shell. I did so happen to keep it simple and make it a batch command. And let's put our...right here let's put the name of our batch file and I will paste that in. So we have call c:\example\_java\_project\deploy\_webapp.bat. Good, so let me Save this, and go Back to Dashboard. So now, we have a new project called Deploy our web application. Now, we didn't put a schedule on this, but we can. But let see if it works first. Let me immediately schedule a build. And the build ran successfully. We could go into the build number 1, go to Console Output, and we could see there a WAR file has been deployed to Apache Tomcat successfully.   
The presenter switches to the browser and creates a freestyle project with the name, Deploy our web application. In the advanced configuration settings of the created project, he goes to the Build section and opens the Execute Windows batch command rich text box. To execute the bat file, he adds the name of the bat file in the previously added command. Now the modified command is "call c:\esample\_java\_project\deploy\_webapp.bat". He saves the configuration settings and returns to the same information page of the project, Deploy our web application. Then he returns to the Jenkins dashboard to show that his new project is added in the displayed table. Then he opens the Console Output of the project to show that the deployment of the file is successful.   
  
So what we have done here is we have a job, a project that goes out, gets the WAR file, and deploys it to Apache. So we have our web app that's deployed. Now let me go back to the project and let's schedule this job. So let me go to Configure and then scroll down. And, for Build Triggers, we will choose Build periodically, and put in the following – I'm going to put in \*/1 \* \* \* \*. What this will do is this will deploy our web application every minute. A little silly for sure. But it's my application, I could do whatever I like. And then, give this a Save and we are done. And I'll go Back to Dashboard. And now we have our project deployed. Our web application will deploy this WAR file every minute. Well, that might be a little silly. Obviously, we probably would not deploy it every minute, but you get the point. All right. Let's review what we did. What we did is we built a project. And what that project does is it takes a WAR file that presumably gets updated often via another Jenkins job. Takes that WAR file and deploys it to Apache Tomcat. So basically, we have an automated deploy process for our web application.   
The presenter returns to the Jenkins dashboard to modify the configuration settings of the Deploy our web application project. The presenter goes to the Build Trigger section and selects the "Build periodically" checkbox. It opens the Schedule rich text box below the checkbox. The presenter adds a line in the rich text field, \*/1\*\*\*\*, and saves the configuration settings. Then he returns to the Jenkins dashboard.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Distributed Builds

## Learning Objective

After completing this topic, you should be able to

* *describe how Jenkins can be used to distribute builds to multiple machines*

## **1.**

Jenkins can be configured to share the build process workload. Most Jenkins installs are standalone, but they don't have to be. Jenkins can manage over 100 build agents – computers that perform the build. In this way, we can say that Jenkins is a distributed build tool. Build agents can be on different operating systems and hardware. Since Jenkins runs on about anything, masters and agents do not have to be on the same operating system. Build agents can be added and removed easily, actually very easily like on the fly. Jenkins can maintain its own "build farm" of build agent machines. All the machines are controlled at one central point. The "master" is the basic installation and configuration of Jenkins. This is the most basic and common install. It's how Jenkins configures itself by default. The master handles all tasks for the build farm – all scheduling and load balancing. The master offloads the build process to the build agents. The build agents do all the work in doing the builds. The master concentrates its resources on the pipeline process. So the master just controls the process. The build agents concentrate their resources on actually building the code. This separation of resources allowed Jenkins to optimize how it runs and how it manages your Java build environment.   
  
Build agents don't need to be full-fledged Jenkins machines. They can be configured just to do the builds and nothing else. Each computer runs a program that makes it a build agent. All the build agents work without knowing that each other exists. Communication between Jenkins and its build agents is bidirectional – they communicate both ways – but never from agent to agent. The distributed build process is mostly transparent to you. The way Jenkins moves jobs around the build farm for the most part is automatic and requires almost no input or configuration by you. The master Jenkins machine becomes a portal into the application build farm. The master takes care of all the details. The master Jenkins box can start a build agent with SSH. Communication is secure. Windows build agents can also be started remotely. Build agent functionality varies from platform to platform. Scripts can also be used to start build agents. Again, script support and syntax will vary by platform. Build agents can also be started with a Java Web Start application or by a variety of other ways. Build agents can automatically be configured with tools like Chef and Puppet. This is a cool way to integrate automated configuration tools with Jenkins.   
  
It's not an exact science how the build agents handle the workload, but it doesn't have to be. Most of this functionality cannot be controlled by you anyway. There are a few guidelines though. If a project has been configured to be built on a specific computer, it always will be. Also, Jenkins will attempt to build the same projects on the same machine each time you build it. Furthermore, Jenkins will attempt to distribute long builds among build agents. Jenkins takes into account latency due to communication. Jenkins is aware of the agent's location and workload.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Different Types of Jenkins Jobs

## Learning Objective

After completing this topic, you should be able to

* *identify the different types of Jenkins jobs*

## **1.**

What I am going to do for you over the next couple of minutes is I am going to show you the different project types that you can build in Jenkins. And while I am added, I'll go through some matter of detail about the different project types and when you would use them. Now what you see on my desktop is Jenkins and I am currently running it as a Windows service. And let me click on New Item and we'll talk about the different types of projects. Now there is basically five – at least five, at least in so far as these radio buttons – types of projects that you can create. You can create a Freestyle project, a Maven project, an External Job project, and a Multi-configuration project. And we can also build a project based on an existing project. Now the most common type of project that you can build is a Freestyle project. It's also, depending on who you talk to, one of the more difficult ones. Because, in a Freestyle project, you have to specify amost all configuration options which you may or may not find as a burden. You might find that being...for a liberating because you know a lot about configuration and you would like to be able to do that yourself without a lot of these different project types can do in some configuration for you.   
The Jenkins dashboard is open in a browser. The presenter clicks the New Item node in the navigation pane and discusses the different job types one by one.   
  
So Freestyle is certainly the most common type of project in Jenkins and it's really the easiest way to at least get one up and going without using things, such as maven to actually to do a lot of this configuration for you. Also, with Freestyle projects, it gives you the most flexibility with source control management. And also, you can use Freestyle projects for stuff other than software builds. You can build other sorts of things with Freestyle projects. Now moving on with Maven projects is you could build a project based on maven. Now there are two more types of Maven projects. But basically, when you create a Maven project is you're taking an advantage of a lot of preconfigured configuration or some existing POM files that you might have out there to actually build a project based on maven. And, depending on how you look at it, it can greatly reduce the configuration because you already going to have this stuff. Although, please keep in mind that everything is relative here and your mileage may vary. And, you know, certain people's opinions might not necessarily be yours. So your mileage may vary when you're looking at Freestyle projects versus Maven projects.   
  
The third type of project is an External Job. And we can create an External Job project when we want to record the execution or something that's not running locally or even within Jenkins or even on a remote machine. What we can do is we can build a project based on an External Job. I will also have Multi-configuration projects. Now this is a project that we may have different configuration files for different things. For example, we have a project that has a configuration for dev, a configuration for test, a configuration for production. If we have something of that nature – rather than building a multiple Freestyle projects – we can build a single Multi-configuration project. And we have the last option here which is Copy existing Item, which is basically take an existing project. And let's build a project based on that. Now how you use these different project types is completely and utterly up to you. You will find, throughout your experience through Jenkins, that some of them would work better for you under certain situations. Some may not work as well for you under different situations. You are the ultimate guide on which type of project fits your needs. Now obviously there is going to be someone that you can gravitate towards rather than other ones. But there is not a lot of standards or best practices in so far of what works well for you and your machine. So, depending on the project that you are going to build, you'll pick one of these five project types.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Build Triggers

## Learning Objective

After completing this topic, you should be able to

* *identify how Jenkins can build jobs based on the occurrence of an event*

## **1.**

What I am going to cover for you in the next few minutes or so is I am going to show you the different ways that we can run our Jenkins projects, the way we can run our Jenkins jobs from our projects. Now what I am going to do is let me pick a project. I will pick Deploy our web application and will talk about the different ways that we can run the sky. Now, when we create this application, we can basically determine from a couple of things how we are going to run it – are we going to run it based on a schedule? Are we going to run it manual? How we are going to do this? So let's go over those options. I am going to navigate down to Configure. And, if you look at some of the parameters for this project, I want to focus on these two different sections here – Source Code Management and Build Triggers. Specifically, I want to focus on Build Triggers right here. Now what we can do is we can specify that this project gets build upon some sort of triggering mechanisms. Now what I have here is I have checked Poll SCM. What this does is this polls our source control management and determines based on if there is better change in our source code management. What is going to happen is it's going to fire this build off. It's very, very, very common if you have a situation where you have developers checking in code or you have, maybe, a promotion group and source control – they periodically get updated. So, in our scenario, if that gets updated, we can actually trigger a build. Now, if you look at the Schedule here, Schedule is in cron format just like it is if we do a build periodically. So, if you don't know of the cron format for dates, this is probably a good time to learn them if you are going to be scheduling build triggers.   
The Jenkins dashboard is open in a browser. The right section shows a table that includes information about the projects, such as Deploy our web application, Jenkins Example, and TestJob. The information includes status, weather, name, last success, last failure, last duration, and build option for the project. The presenter clicks the Deploy our web application in the table to open its details. The navigation pane also shows the Build History of the opened project. The presenter clicks the Configure node to open the configuration settings of the Deploy our web application project in the right section.   
  
Now, in addition of Poll SCM, we also have Build periodically. And Build periodically will fire off a build upon some sort of numerical or some sort of time interval. We could build something every five minutes, we could build something every five years. But, when that trigger happens – where the time interval is hit – this build will be fired. The third build trigger we have is Build after other projects are built. And let me give this a check. And this is common because we may want to fire this job off or we may want to build this project upon successful completion of, perhaps, another project where we have project A. And project A is built successfully. And then, when project A is built successfully, we may want to fire off this one, which is project B. So we can fire projects off each other when they are built successfully. And, by default, it will only trigger if the build is stable. This is the build that we're triggering off of. So there are three different build triggers. We have Build after other projects are built. And here we specify a project to watch. And we have three options that we could trigger the build only if the build is stable. I suppose we could do so if it's unstable or the build fails. We can trigger on a specific time interval when we check Build periodically. Or what we can do is we can build based on polling source control management and based on anything changing the source control management. We can actually perform a build that way.   
The presenter scrolls down to select the "Build after other projects are built" checkbox in the Build Triggers section. This checkbox enables the user to add the name of the project he wants to keep check of and triggers built on different situations such as trigger only if build is stable, trigger even if the build is unstable, and trigger even if the build fails.   
  
I am going to uncheck Build after other projects are built. And I have Poll SCM. And I checked that on purpose when you were not watching because we could always look to see the results of this poll if we navigate to the left and go to the Polling Log. Polling Log will show us the last time source control is checked. And here it is going to check every minute. And it just says here that source control is fine, there have been no changes so no build was actually fired. Okay, so we went over our triggering mechanisms and how Jenkins job can be fired based on some triggering event.   
The presenter unchecks the "Build after other projects are built" checkbox and clicks the Polling Log node in the navigation pane to show polling log details of the project.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Scheduling Build Jobs

## Learning Objective

After completing this topic, you should be able to

* *describe how jobs can be scheduled in Jenkins*

## **1.**

What I am going to show you over the next few minutes is I am going to show you how you can schedule these Jenkins jobs or how you can go to a Jenkins project and schedule these jobs to actually run. Now currently we don't have any schedules for these jobs. So, when we set these jobs up, we basically set them up as one-offs. And, if we want to actually run them, we have to go and actually click on the icon to actually schedule the build. Now let me show you how we can automate this and put these projects on some kind of schedule. So we can have some sort of job get fired off on regular intervals. So let me navigate to, first, the job that I want to schedule. And, in this case, I will pick Jenkins Example and then I am going to navigate towards, say, Configure. So what we are going to do is we are going to take an existing project, change the configuration. So we could launch some jobs that go off at regular intervals or schedule. So let me navigate down. And you should see a couple of checkboxes under Build Triggers. One is Build after other projects are built, one is Poll SCM, and the other is Build periodically. I am going to choose Build periodically. Now what can I do here is in this schedule area is I can specify the interval in which I want to consistently and continually run this job. Now it's in cron format. If you don't know how to use cron date format you are going to have to learn it. So I am going to kind of just go over some of the basics of it.   
The Jenkins dashboard is open in a browser. The right section shows a table that includes information about the projects, such as Jenkins Example and TestJob. The presenter opens the configuration settings of the Jenkins Example. He goes to the Build Triggers section and explains each of its options.   
  
Cron format or date format has five mandatory places which I am going to denote by the star. The first is minutes, the second is hours, the third is the day of the month, the fourth is the month, and the last one is the day of the week. Now there is an optional one for the year that comes next, but we are going to leave that out. So, based on cron format, you specify the interval and when you want this job to run. For example, if I want this job to run 5 minutes after every hour, I would change the minute to 5. So, as it sits here, this job will fire off 5 minutes after every hour. Now there are wildcards that are supported as well such as the following – if I put in a /5, this means I want this job to run every 5 minutes. Now you see where I am going with this. After you learn cron, you can specify what the interval is and run them at that interval. I am going to pick one to run this job every minute because it's my job, I can do this. So let me save and it actually prompted me," Are you sure you mean every minute?" And I do. So let me go Back to Dashboard. Now, if we wait a minute, we will see that this job Jenkins Example will run. So we'll give it a minute and will refresh our browser. And we should see that it ran. And it last ran 7.2 seconds ago. So our scheduling worked and Jenkins Example will run every minute.   
The presenter adds the "\*/1\*\*\*\*" command in the Schedule rich text box of the Build periodically checkbox and saves the modification. Then he returns to the Jenkins dashboard and refreshes the page.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Polling the SCM

## Learning Objective

After completing this topic, you should be able to

* *identify how Jenkins can poll for software changes*

## **1.**

What I am going to demo for you over the next couple of minutes is I'm going to show you how you can configure Jenkins jobs and Jenkins projects to actually poll the source control management system and look for changes and trigger jobs based on those changes. And it should just take a couple of minutes or so. So let's get started. What I have here on my desktop is I have Jenkins. I am running Jenkins on local host, so it's on my local machine. Now, if you're following along with me, I'm going to navigate to New Item. And, from there, what I'm going to do is I'm going to build a new project that's going to scan the source control system and look for changes. So let me give it a click. And we need to give this Item name, which is basically a Jenkins project. So I'm going to name my project dropsofjupiter. And you can use any of these project types – you can use Freestyle project, you can use Maven project, External Job, Multi-configuration project, or you could even copy a project from an existing one if you want your new project dropsofjupiter to have all of the properties of the copied project. So I'm going to select Freestyle project and then click on OK. And I'll skip the Description. And I'll skip all of the properties except for these two. These two properties are very important. We have Source Code Management and we have Build Triggers.   
The Jenkins dashboard is open in a browser. The right section includes a table that lists the information about the projects such as Deploy our web application, Jenkins Example, Promote from Dev to Test, and TestJob. The presenter creates a freestyle, "dropsofjupiter," project. Now the right section shows the advanced configuration settings of the dropsofjupiter project.   
  
Now, under Source Code Management, it defaults to None because Jenkins assumes that you do not have any Source Code Management installed on your local node. Now out of the box or shipping with Jenkins, you have CVS and you have Subversion. So, if you're using either of those Source Code Management systems, you're not because right out of the box, you can take Jenkins and you could hook Jenkins up or you can connect it from an interface from the existing Jenkins library. Now, if you have other source control management systems, you would see options for them in here, but you would have to install them as a plugin. So Jenkins will allow other source code management systems, but you have to install them separately. Now our most important property is right here – where we have Build Triggers. Now there is three different ways that we can actually assign a trigger to a Jenkins build job. First is we can fire this job off after other projects are built. We can build this job or build this project via a job periodically by putting in a cron time. And this is the one that we're going to do. We can click on Poll SCM – poll on source code management. Now there are two things that we can do here. We can either leave the Schedule blank, which means that we won't actually go out on a regular interval. Or what we can do is we could put in a Schedule, which would actually go out to source code management and see if there has been any changes. Any changes being that someone logged in and perhaps made a code change and check their change in. So I'm going to put in the following. I'm going to put in \*/5 \* \* \* \*. Now, if you don't know this format, this is the cron format for time. And, if you need to learn it, this is probably a good time.   
The presenter goes to the Build Trigger section and selects the Poll SCM checkbox to add "\*/5\*\*\*\*" command in the Schedule rich text box.   
  
So what this is saying here is we're going to poll source control management every five minutes. And actually, I will change this for demonstration purposes to poll every minute. Now, in reality, you're probably not going to poll your source control management system this often. It's a matter of fact – you probably poll it maybe at the end of the day or maybe even weeks at a time. But it's your project, you can do whatever you like. Now I'm going to Apply changes. And I'm going to Save. And there we go. Now, if you look on the left-hand side, is you see a link for the Polling Log. And you see Project dropsofjupiter. And, if we want to look to see how often or the last time that this project actually polled source control management, always we need to do is give Polling Log a click. And, in this case, we have "No existing build. Scheduling a new one. Done. Took 1 ms Changes found." Now, in this case, this was the first time that dropsofjupiter actually polled source control. It did not find a build, which means that there has been a change because we go from nothing to something meaning we don't have a build and we want one. So the Polling Log tells us that a new build has been scheduled based on this result being that there has been no build that's out there. And it says it took 1 ms to actually do the poll. And we have the result – changes found. Now every minute, which might be a little bit ridiculous because it's a very, very small interval, what is going to happen is Jenkins is going to poll SCM on behalf of dropsofjupiter. And, if there has been any changes in source control, what is going to happen is it's going to trigger a build.   
The presenter modifies the command in the Schedule rich text box to "\*/1\*\*\*\*". Then he clicks the Apply button at the bottom of the right section and the information of the dropsofjupiter project is shown. The presenter clicks the Polling Log in the navigation pane to show polling log details of the dropsofjupiter project.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Jenkins Security Overview

## Learning Objective

After completing this topic, you should be able to

* *describe how Jenkins implements security*

## **1.**

By default, Jenkins is completely insecure. When it's downloaded, it does not even need a user to log in. Anyone who has access to the URL can access Jenkins. These users can access and run anything you can. They may as well be you. The first step is securing the Jenkins URL. The URL is the only thing that needs to be secured as this is the only way Jenkins is accessed. Security settings can be configured via the Jenkins HTML landing page. Security in Jenkins is configured directly through the Jenkins application. Settings such as the security realms and authorization strategies are set via the Jenkins HTML interface. It is possible to use Apache security for a Tomcat instance that runs Jenkins. Although technically this is Apache security and not really Jenkins. This is performed by enabling a proxy. Jenkins will run as this proxy. This will only work if Apache and Jenkins run on the same machine. This may be an issue as...if Jenkins is not configured as an Apache application, both Jenkins and Apache may compete for the same ports. The host will have to be configured to use basic authentication. In this way, we are using Apache security to run Jenkins. Using Apache security for Jenkins does not scale very well.   
  
Jenkins security only exists on the user and role level. And these users and roles are Jenkins users and roles in most cases. Jenkins does nothing to secure the code or Jenkins project configuration. The code is not directly exposed through the Jenkins interface anyway, but the files and the scripts that are built are. Most files that Jenkins uses are secured by the network and the computer that they live on. Jenkins is a bunch of files on the host computer and relies on the same security as everything else. Server configuration and system files cannot be accessed directly by Jenkins. So these files are not at risk. Don't expect Jenkins to secure your code or plugins. This is not really, what Jenkins does as far as security goes. Jenkins relies heavily on the security of the host computer. Sensitive code should be secured by the host server. The host should keep these files secure. Consider storing your code in a private code repository. The private repository will add additional layers of security to your code. GitHub is a popular source repository tool that offers private code repositories. GitHub has free public repositories. The private ones are pay only. Bitbucket offers free private code repositories for up to five users. Curiously, Bitbucket uses Git as its ultimate source code repository.   
  
Jenkins security only augments the security already in place. Jenkins doesn't really have a security model. Never solely rely on Jenkins security. Especially since Jenkins by default will give anyone access to the URL. The Jenkins security model was not meant to be robust. Future versions of Jenkins may offer a few more security features. The features it does have – user and role – are not very robust. Also security must be reapplied in Jenkins when upgrading to a more current version.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Authentication

## Learning Objective

After completing this topic, you should be able to

* *identify how Jenkins authenticates applications*

## **1.**

Jenkins provides authentication through its security realm. Authentication has to do with who a user actually is. The security realm authenticates users and passwords. Here Jenkins users and their passwords can be maintained. The security realm also maintains groups. Groups can be assigned security tasks just like users. There are two ways to establish a user authentication method. The one you choose is up to you but you must use one or the other. You can choose a method such as LDAP or use built-in Jenkins authentication. Jenkins maintains its own database for simple authentication needs. Using the Jenkins database, users can be set up with their own accounts. The Jenkins database is the most simple of the two mechanisms to set up. Simply create a user and a password. User accounts in Jenkins are different than network credentials. Jenkins users exist only for Jenkins jobs and tasks. The security realm settings determine what a user is allowed to do. This can be along the lines of viewing projects or being allowed to create and run them. Roles can also be specified via the Role Strategy Plugin. Roles and user authentication are similar. Users can be assigned to the roles that they are allowed to do.   
  
Jenkins allows global roles and project roles. These project roles can be assigned to individual users or a role of users. Global roles specify what can be performed at the master and build agent level. Tasks would include configuring the build process and setting up user role authentication. Project roles specify read, write, and execute privileges at the project level. Tasks here would include who can create and run projects. Neither global or project roles have nothing to do with access to the Java code, which Jenkins does not allow anyway. Remember that Jenkins is a scheduler, not a code editor. There are also roles that can be specified solely at the build agent level. Like how that agent communicates with the Jenkins master machine. Scripted clients need to be authenticated as well. These scripted clients can call Jenkins. Scripts have many purposes such as scheduling a build. Examples of this include calling Jenkins through a URL or accessing Jenkins through the command line. Authentication occurs through HTTP Basic. HTTP is really performing the authentication, not Jenkins. In HTTP Basic Authentication, a username and a password is supplied as part of the request. The username and password pair must be set up in Jenkins using one of the methods described. Jenkins does not perform authentication negotiation. This is performed through the HTTP server.   
  
Jenkins also uses the OpenID plugin for authentication. This is one of many plugins for third-party authentication. The OpenID plugin allows Single Sign-On. This is useful when logging into outside applications or when you want to use third-party sign in such as Facebook. The OpenID and Role Strategy Plugins can work together to authenticate. This is actually pretty simple. Other plugins offer additional authentication support. The Google login plugin allows you to sign in to Jenkins with your Google account. You will find many plugins that work this way.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Authorization

## Learning Objective

After completing this topic, you should be able to

* *identify how Jenkins authorizes users*

## **1.**

The Jenkins authorization strategy defines who has access to what. This is different than authentication that defines who users are. The authorization strategy is configured in the authorization section on the Configure Global Security page. All authorization is defined within a Jenkins web page. There are no security files per say. In Jenkins, authentication comes before authorization. Jenkins must first resolve who user is before deciding what they can do. Jenkins authorization solves certain security issues. With authorization, users can be granted privileges in a very exact way as far as Jenkins jobs go. Access to project code and execution of processes can be granular and assigned on a user-by-user and a process-by-process basis. Jenkins authorization is tiered and configured in levels. All authorizations are really subsets of the following Jenkins blanket authorization types. The first type is anyone can do anything – this is by default. When you install Jenkins, there is no user ID. You must first create them. The second type is the GitHub committer authorization strategy. Here the authorization strategy for GitHub is applied to the user. Next up is logged in users can do anything. Once a user has been created, they are basically an administrator. Last is the most precise type – the project-based authorization strategy. Here authorization can be defined project by project.   
  
The root-level user is the admin user. Like on other systems, the admin user can do just about anything on the Jenkins system level or project level. The Admin user can view all projects regardless of who wrote them. Admin can also create and authorize other users. Users can be assigned system or project-level grants. Admin can also create, read, update, and delete Jenkins projects regardless of who wrote them. The administrator can also manage all Jenkins processes. This can be performed on the master or any of the build agent machines. Authenticated users have a minimum level of what they are allowed to do based on their security type. For example, authenticated users are authorized to log in to Jenkins. They cannot create projects and run them. But they can view Jenkins projects on the dashboard. They cannot create other users, but authenticated users can view other Jenkins users. Other privileges must explicitly be given by the admin. These privileges can differ from project to project. Privileges can be added via the project matrix. This is the most detailed authentication method used in Jenkins.   
  
The project-based matrix sets the authorization strategy. It is the most detailed way authorization can be defined in Jenkins. The authorization strategy is create, view, execute, and delete based. Each of these can be configured simply by checking a box indicating what the user is allowed to do. This can be very molecular as you can give a user execute access to some projects and not to others. The authorization strategy can also be defined for a build agent. Build agents are really just users here. Privileges can be usually defined by the user role. These privileges are defined in the matrix.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Running Jenkins as a Standalone Application

## Learning Objective

After completing this topic, you should be able to

* *use Jenkins as a stand-alone application*

## **1.**

What I am going to show you how to do over the next couple of minutes is how to run Jenkins as a standalone application. Now Jenkins can run in three different modes. We can run it as a full-fledged web app, we can run it as a Java Web Start application, or we can run it standalone – meaning it's going to run in its own servlet container without any other supporting software. And that's where I am going to demo for you right now. This is what you see on my desktop. On the top, we have a web browser. And, if you are following along with me, you can use any web browser – anyone will do just fine. And on the bottom of my desktop, you see the command prompt. And this is where we are going to start Jenkins as a standalone application. Now I navigated to a folder called JenkinsInstall. Now JenkinsInstall is a folder that I created that has the Jenkins WAR file in it. Now, if I just quickly just go and do a directory, you will see in this folder I currently have jenkins.war.   
Two windows are open, browser and Command Prompt. The "jenkins-ci-org" URL is open in the browser, which shows the home page of the Jenkins website.   
  
Now anything else it really doesn't matter what is in this folder. As long as you have the WAR file, you are okay. So from that directory, issue the following command – java -jar jenkins.war. What this will do is this will search Java. It will unjar all the files that mean jenkins.war file. It will explode the directory structure and all the files to this root folder. And it will start the Winstone servlet engine, which is the small servlet container that ships with Jenkins. So let me quickly just give this an Enter. And you should see some affirmative messages coming back to you. And it's initializing Winstone. It's giving some messages about the status of exploding this WAR file. And you should get the message after about 30 seconds – "Jenkins is fully up and running". So basically, what you just did is you started Jenkins as a standalone application. Now the term application is used liberally here because it's really not a Java application as much as it's a servlet container. But it's running in its own independent servlet container. It's not using Apache Tomcat or any other piece of software to actually run. Now to confirm that this is running, you can go to your browser and type in localhost:8080. And you should get your Jenkins dashboard – "Welcome to Jenkins!" So it looks like everything is up and running. And Jenkins is running in its own servlet container, and we have been successful. So congratulations, you just installed Jenkins and ran Jenkins as a standalone application.   
The presenter executes the "java –jar jenkins.war" command in the Command Prompt window. The output returned shows a message, Jenkins is fully up and running. Now the presenter opens the "localhost:8080" URL in the browser. It shows the Jenkins dashboard.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Running Jenkins on an Application Server

## Learning Objective

After completing this topic, you should be able to

* *use Jenkins as a web application*

## **1.**

What I am going to show you over the next couple of minutes is I am going to show you how you can download and install Jenkins as a web application. Now there are three different ways that you could run Jenkins. You can run Jenkins as a full-fledged web application, meaning that you can deploy the .war file and an application server where web application server, http server such as Apache Tomcat. You can run Jenkins as a Java web start application. Or you can run Jenkins standalone in its own servlet container. So I am going to go over the first scenario where we are going to download the .war file and deploy the .war file and run Jenkins as a full-fledged Web App. Now, if you go to the Jenkins site, on the right-hand side you will see a heading that says "Java Web Archive or the .war file."   
Two tabs are open in a browser. In the first tab, the Jenkins home page is open and in the second tab, Apache Tomcat is open.   
  
Now, in my current experience here, the latest and the greatest is 1.6. So, while you were not looking, what I did is I took this file and I downloaded it to my local hard drive. Now it's a fairly good size .war file. It's about 16 megabytes or so. So you may have to change some of your server settings to be able to deploy a .war file of this size. But what we are going to do is I am going to install this .war file in Apache Tomcat. Now what I am going to do is I am going to change my tab to go to Apache Tomcat. And I have Apache Tomcat installed on my local machine. So I have localhost:8080. Now obviously you would change it to whatever port number you have, Apache Tomcat listening on, and, of course, your machine name – in my case localhost:8080. So let's deploy this .war file as a Web App. I am going to navigate to the Manager App. And I'm already logged in. So I don't need to put in my admin user id or password. And, as you can see, this is a fairly fresh install of Apache Tomcat. If you look at the applications, I only have the two default applications running. So let's deploy Jenkins. So I am going to navigate down towards its WAR file to deploy. Click on Browse and then choose on your local drive where you have the .war file.   
The presenter goes to the second tab to open the home page of Apache Tomcat. The home page includes three buttons on the right side: Server, Manager App, and Host Manager. He clicks the Manager App button to open the Manger App page. The page includes different sections, such as Manger, Applications, Deploy, WAR file to Deploy, and Diagnostics. The Applications section includes a table, which lists the path, version, display name, running status, sessions, and commands associated with the application. Some of the applications listed in the table are Tomcat, Tomcat Documentation, and Tomcat Manager Application. The WAR file to Deploy section includes the Browse and Deploy buttons. The presenter clicks the Browse button to browse and select the file from the stored location.   
  
Now I downloaded the .war file and put it on my local drive. So let me select it and then click on Deploy. Now what Apache Tomcat is doing is it's deploying the .war file. It's creating a folder under Web Apps. It's exploring the .war file under the Web Apps folder and creating a web application. And it should take about maybe 90 seconds or so maybe 2 minutes to actually finish exploring the .war file and making the .war file available to us. So we completed successfully. Now, if I navigate to the left, I should see that we have Jenkins as a web application. Now I could put Jenkins as part of the context in URL or I can just click on the link, they both will do the same thing. But let me click on Jenkins and you should see "Welcome to Jenkins!" And to verify look in the URL. And you see your web server, the port you are listening on, and the context – which is Jenkins. So congratulations you just installed Jenkins as an Apache Tomcat web application.   
The presenter clicks the Deploy button to deploy the selected file. After few seconds, the file deploys successfully and the Jenkins application gets added in the table displayed in the Application section. The presenter clicks the Jenkins path displayed in the table and the Jenkins dashboard opens in the browser.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Installing Jenkins as a Windows Service

## Learning Objective

After completing this topic, you should be able to

* *describe the process for installing Jenkins as a Windows service*

## **1.**

What I am going to go over is I am going to show you how we can do an installation of Jenkins and set Jenkins up to be a Windows Service. Now, when we set Jenkins up as a Windows Service what we are doing is we are setting it up to actually be started every time the server is started. What you see here on my desktop is, on the upper right-hand side, I have a web browser, which we're going to use it to confirm the installation of Jenkins. On the left, I have my Windows Services pane that we can see that we have installed Jenkins as a Windows Service. And on the bottom, I have my command line where we're going to be issuing commands to actually do the install of Jenkins. Now the first thing that you need to do the install and to set Jenkins up as a Windows Service is to make sure that you have jenkins.war. Now where do you get this? Well you could get this from the Jenkins' web site. And you can download the most current build of the WAR file. It's usually on their home page.   
Three windows are open: Services manager, browser, and Command Prompt.   
  
So, once you download jenkins.war, the next step is to actually do the install. Install Jenkins from the WAR file and then from there what we can do is we can set Jenkins up as a Windows Service. Now the next step is to do the actual installation and we do that via this command – java -jar jenkins.war. Now what this does is it runs java, specifies that we're going to be unjarring a file and that file name is jenkins.war. So let me hit Enter and we only have to wait about 90 seconds or so because right now it's installing Winstone which is its servlet container engine, which will fire up and run on port 8080. So we have to make sure that we don't have anything else running on port 8080, for example, you might have Apache Tomcat running on port 8080. So you have to be careful to make sure you don't have any conflicts. So at the end of the install you should see Jenkins is fully up and running. So the install worked fine. So let's go to our web browser and let me make this little bit larger so we can Maximize some real estate, and let's verify our install. So we can just do localhost:8080 and we give it a second and we should get the Jenkins dashboard, and we do.   
The presenter executes the "java - jar jenkins.war" command to install the Jenkins. The presenter maximizes the browser and opens the Jenkins dashboard.   
  
So we have been successful in installing Jenkins as a standalone application. So it's running in its own container engine. Winstone is not running on an Apache Tomcat or any other third-party HTTP web server. Now the next step is to set this up as a Windows Service. So we could do that from the dashboard by going into Manage Jenkins. And, if we scroll all the way down, you will see an icon and a link that says Install as Windows Service. Now here is the deal with this. If for some reason Jenkins is already installed on this Windows machine and it's already installed as a Windows Service, you wouldn't see this icon. So, if you're running Jenkins and you don't see this icon, it's already installed as a Windows Service. And again, you can verify that by going into Windows Services and just look to see if Jenkins is indeed running. But in this case, we see the icon and that tells us that it is not running as a Windows Service. So, to set it up as a Windows Service, I'm going to click on Install as Windows Service. It's going to ask us for the Installation Directory. Now this Installation Directory must already exist. So, while you were not looking, I created a folder called jenkins.   
The presenter clicks the Manage Jenkins node in the navigation pane and the right section shows its respective advanced options, such as About Jenkins and Manage Old Data, and Install as Windows Service. He clicks the Windows Service option to show its respective settings. Now the right section shows the Install Directory field and Install button. The presenter sets the jenkins folder in the C drive as a default folder of the installation files.   
  
So let me double-check to make sure my spelling is correct, and it looks okay. And let me click Install. So right now, Jenkins is being set up as a service and we see this message "Installation is successfully completed. Do you want to stop this Jenkins and start a newly installed Windows service?" Why not? Click on Yes. And we'll wait for Jenkins to restart. Now it's restarting as a Windows Service. And we'll give it a moment, and in a couple of seconds we should get a successful message. And we can verify when it's done by going to Windows Services and looking to see if Jenkins is indeed running. Now it's important to point out that we can't have Jenkins already running as a service. So, if you're going to perform the step, it's probably a good idea to go under Windows Services and make sure that Jenkins isn't already installed as a Windows Service. So what I'm going to do is I'm going to Minimize this slightly, and let me make some room. And, if I go to Windows Services, which is right here on my left, if I scroll down, I should see Jenkins. And, if I don't see it, I can right-click, and now you can see that jenkins is running as a service. So we've successfully installed Jenkins as a standalone application and then configured Jenkins to run as a Windows Service.   
The presenter clicks the Install button and a message with the Yes button appears in the right section. The messages prompts that the installation is successfully completed. Do you want to restart the application? The presenter clicks the Yes button and the Jenkins dashboard appears in the browser. Then he goes to the Service manager window to confirm that the Jenkins is running as a service in the Windows.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Non-Java Options

## Learning Objective

After completing this topic, you should be able to

* *identify how Jenkins can be used for non-Java applications*

## **1.**

Jenkins is built on Java for Java. Jenkins evolved from Hudson – another Java tool. Neither was built to support non-Java applications. Support for non-Java projects is limited and not close to the level of support given to Java. There are plugins that bridge Jenkins out of the Java world. Plugins is how Jenkins supports non-Java platforms. It's more of an issue of other languages and platforms supporting Jenkins. Jenkins has a predefined open-source API. Jenkins does not play well with proprietary non-open-source software. It's not just the Java, non-Java thing. Java-based platforms and systems are just more open than non-Java ones. Jenkins can work with .NET – support is spotty and very limited. C# and Visual Basic applications can be built in Jenkins. Or rather Visual Studio projects containing C# and Visual Basic can be built using Jenkins. Functionality is performed via a .NET plugin. Actually, there are a few plugins to choose from. Functionality is not seamless and must be implemented using several techniques. Integration still requires manual steps when dealing with .NET.   
  
.NET applications require many dependencies. The Microsoft world contains its own framework and dependencies. These are configured easily in .NET as it expects them. Each of these dependencies must be defined and configured manually in Jenkins. This can be unpredictable and buggy. In many cases, functionality is changed when configuring dependencies. Sometimes it can't be done at all. Jenkins works better with more current versions of Visual Studio. Don't expect any backward compatibility. Older .NET applications may not even work at all. Jenkins will be better supported as more platforms become "open". Don't though expect Jenkins to be heterogeneous with non-open-source tools. Microsoft Visual Studio 2013 is showing signs of becoming more open – even allowing git for source control. Visual Studio is still a closed system. These closed systems have secret APIs. Open-source tools like Jenkins cannot work well with tools that have undocumented APIs. Many closed tools work only with tools from the same vendor. Closed tools have nondocumented dependencies. Remember that Jenkins is first and foremost a Java tool. Don't expect wholesale support for other languages.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

# Exercise: Using Jenkins

## Learning Objective

After completing this topic, you should be able to

* *describe in detail the Jenkins tool and how Jenkins implements continuous delivery*

## **1.**

Now that you have had a chance to learn about Jenkins, let's see if you could answer the following questions. In this exercise, you will identify the two ways that Jenkins can be installed; you will describe what is meant by a build farm; you will name the languages that Jenkins can work with; you will identify the platforms Jenkins can be run from; and finally, you will describe how Jenkins is considered a continuous integration tool. Now, if you like, what you can do is you can pause the video and perform the exercise. After you're done coming up with your answers, you can resume the video and compare them to what I came up with.   
  
Hello and welcome back. Did you come up with your answers? Let's compare your answers to what I came up with. Now, if our answers aren't exactly alike, that's okay. There is more than one solution to each of these questions. Identify the two ways Jenkins can be installed? Jenkins can be downloaded and run standalone or can be deployed as a WAR file and run as an Apache Tomcat web application. Additionally, Jenkins can run as a service on Windows machines. Describe what is meant by a build farm? Jenkins machines can be configured as masters and build agents. Build agents can number in the hundreds. Build agents help balance the workload of building applications. The masters and the build agents collectively are considered a build farm.   
  
Name the languages that Jenkins can work with? Jenkins primarily works with Java. There is limited support for other languages such as C# and VB.NET, and plugin support for other languages such as C. Identify the platforms Jenkins can run from? Jenkins can run on Windows, UNIX, Linux, and Macintosh. Describe how Jenkins is considered a continuous integration tool? Jenkins continuously integrates code into different promotion groups in your pipeline. Code can be promoted in series or in parallel. So how did you do? Were your answers close to what I came up with? As long as you're close, you'll probably be OK. Congratulations, you're well on your way to learning Jenkins.

[Back to top](https://library.skillport.com/courseware/Content/cca/sd_jenk_a01_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited