Glossary

General Terms

**Agent**

An agent is typically a machine, or container, which connects to a Jenkins master and executes tasks when directed by the master.

**Artifact**

An immutable file generated during a [Build](https://jenkins.io/doc/book/glossary/#build) or [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) run which is **archived** onto the Jenkins [Master](https://jenkins.io/doc/book/glossary/#master) for later retrieval by users.

**Build**

Result of a single execution of a [Project](https://jenkins.io/doc/book/glossary/#project)

**Cloud**

A System Configuration which provides dynamic [Agent](https://jenkins.io/doc/book/glossary/#agent) provisioning and allocation, such as that provided by the [Azure VM Agents](https://plugins.jenkins.io/azure-vm-agents) or [Amazon EC2](https://plugins.jenkins.io/ec2) plugins.

**Core**

The primary Jenkins application (jenkins.war) which provides the basic web UI, configuration, and foundation upon which [Plugins](https://jenkins.io/doc/book/glossary/" \l "plugin) can be built.

**Downstream**

A configured [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) or [Project](https://jenkins.io/doc/book/glossary/#project) which is triggered as part of the execution of a separate Pipeline or Project.

**Executor**

A slot for execution of work defined by a [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) or [Project](https://jenkins.io/doc/book/glossary/#project) on a [Node](https://jenkins.io/doc/book/glossary/#node). A Node may have zero or more Executors configured which corresponds to how many concurrent Projects or Pipelines are able to execute on that Node.

**Fingerprint**

A hash considered globally unique to track the usage of an [Artifact](https://jenkins.io/doc/book/glossary/#artifact) or other entity across multiple [Pipelines](https://jenkins.io/doc/book/glossary/#pipeline) or [Projects](https://jenkins.io/doc/book/glossary/#project).

**Folder**

An organizational container for [Pipelines](https://jenkins.io/doc/book/glossary/#pipeline) and/or [Projects](https://jenkins.io/doc/book/glossary/#project), similar to folders on a file system.

**Item**

An entity in the web UI corresponding to either a: [Folder](https://jenkins.io/doc/book/glossary/#folder), [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline), or [Project](https://jenkins.io/doc/book/glossary/#project).

**Job**

A deprecated term, synonymous with [Project](https://jenkins.io/doc/book/glossary/#project).

**Label**

User-defined text for grouping [Agents](https://jenkins.io/doc/book/glossary/#agent), typically by similar functionality or capability. For example linux for Linux-based agents or docker for Docker-capable agents.

**Master**

The central, coordinating process which stores configuration, loads plugins, and renders the various user interfaces for Jenkins.

**Node**

A machine which is part of the Jenkins environment and capable of executing [Pipelines](https://jenkins.io/doc/book/glossary/#pipeline) or [Projects](https://jenkins.io/doc/book/glossary/#project). Both the[Master](https://jenkins.io/doc/book/glossary/#master) and [Agents](https://jenkins.io/doc/book/glossary/#agent) are considered to be Nodes.

**Project**

A user-configured description of work which Jenkins should perform, such as building a piece of software, etc.

**Pipeline**

A user-defined model of a continuous delivery pipeline, for more read the [Pipeline chapter](https://jenkins.io/doc/book/pipeline/) in this handbook.

**Plugin**

An extension to Jenkins functionality provided separately from Jenkins [Core](https://jenkins.io/doc/book/glossary/#core).

**Publisher**

Part of a [Build](https://jenkins.io/doc/book/glossary/#build) after the completion of all configured [Steps](https://jenkins.io/doc/book/glossary/#step) which publishes reports, sends notifications, etc.

**Stage**

stage is part of Pipeline, and used for defining a conceptually distinct subset of the entire Pipeline, for example: "Build", "Test", and "Deploy", which is used by many plugins to visualize or present Jenkins Pipeline status/progress.

**Step**

A single task; fundamentally steps tell Jenkins *what* to do inside of a [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) or [Project](https://jenkins.io/doc/book/glossary/#project).

**Trigger**

A criteria for triggering a new [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) run or [Build](https://jenkins.io/doc/book/glossary/#build).

**Update Center**

Hosted inventory of plugins and plugin metadata to enable plugin installation from within Jenkins.

**Upstream**

A configured [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) or [Project](https://jenkins.io/doc/book/glossary/#project) which triggers a separate Pipeline or Project as part of its execution.

**Workspace**

A disposable directory on the file system of a [Node](https://jenkins.io/doc/book/glossary/#node) where work can be done by a [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) or [Project](https://jenkins.io/doc/book/glossary/#project). Workspaces are typically left in place after a [Build](https://jenkins.io/doc/book/glossary/#build) or [Pipeline](https://jenkins.io/doc/book/glossary/#pipeline) run completes unless specific Workspace cleanup policies have been put in place on the Jenkins [Master](https://jenkins.io/doc/book/glossary/#master).

=====================================================================================

Table of terms used in Jenkins

| **Term used in Jenkins** | **Description** |
| --- | --- |
|  |  |
| Job/Project | Jenkins seems to use these terms interchangeably. They all refer to runnable tasks that are controlled / monitored by Jenkins. |
| Build | Result of one run of a Project. |
| Node/Slave | Slaves are computers that are set up to build projects for a master. Jenkins runs a separate program called "slave agent" on slaves. When slaves are registered to a master, a master starts distributing loads to slaves. Term Node is used to refer to all machine that are part of Jenkins grid, slaves and master. |
| Cloud | Handles creation of Nodes to dynamically expand/shrink the number of slave machines |
| Executor | Separated stream of builds to be run on Node in parallel. Node can have 1 or more Executors. Special executors can be created dynamically (one-off executors) to run lightweight jobs used mostly for orchestration purposes. |
| Workspace | Disposable directory on Node used as a working directory for building. It is preserved on best effort bases after build completion. |
| Stable build | A build is stable if it was built successfully and no publisher reports it as unstable. |
| Unstable build | A build is unstable if it was built successfully and one or more publishers report it unstable. For example if the JUnit publisher is configured and a test fails then the build will be marked unstable. |
| Successful build | A build is successful when the compilation reported no errors. |
| Broken build  Failed build | A build is broken if it failed during building. That is, it is not successful. |
| Completed build | A build is completed, if it was started and finished with any result, including failed builds. |
| Upstream project | A project can have one or several upstream projects, which means that a build for the current project may be scheduled when an upstream build is finished. Per default every *stable* upstream build will schedule a build in the downstream project, but there are several options and plugins which can customize this behaviour. |
| Downstream project | A project can have one or several downstream projects. The current project is then known as an *upstream project* of the downstream project. See *Upstream project* for what this means regarding scheduling of builds. |
| (Un)Stable project | A project is (un)stable if its most recent (completed) build is (un)stable. |
| Broken project | A project is broken if its most recent (completed) build is broken. |
| Publisher | A publisher is part of the build process other than compilation, for example JUnit test runs. A publisher may report *stable* or *unstable* result depending on the result of its processing. For example, if a JUnit test fails, then the whole JUnit publisher may report *unstable*. |

=======================================================================

JENKINS CI TERMINOLOGY

As you continue on and read more about the Jenkins Continuous Integration Server, you will see certain terms used over and over. Below are the more common Jenkins terms you should know:

* **server** – The Server is what serves the Jenkins UI  to your favorite web browser and manages all the configuration settings and jobs.
* **client/agent/slave** – The “client” (aka agent or slave) is the build machine that can be use for your software builds.
* **executor** – An executor is the number of processes an “agent” machine can run concurrently. If an agent has 4 executors, it can run 4 separate builds (jobs) at the same time.
* **master** – This refers to the Jenkins server host. If you are using your Jenkins server as an agent, you may not see this because you only have one system being used for both. Once you configure additional agents, your agents will have hostnames assigned to them and the “master” will be your server. You typically don’t run jobs on your master when you have dedicated build machines (agents). I will run small quick jobs that don’t require a lot of processing power on my master and other jobs that I may need access to files/dirs/resources on the Jenkins server machine. Other than that, you should run all builds on your dedicated build machines (agents).
* **job** – A job is something you create in Jenkins that performs some action or set of actions. For example a job could check out source code from a source code repository (i.e GIT), then compile source files into binary files, run tests, package these up, maybe deploy to some environment etc. Jobs can be simple and small or large and complex. It all depends on what the job needs to do.
* **plugins** – Almost everything Jenkins does, from connecting to source code repositories, compiling code, generating reports to deploying artifacts, get processed by plugins. There’s a Subversion plugin that knows how to operate with Subversion servers and repositories, a Git plugin for Git servers, Artifactory, Maven, Ant, Junit and the list goes on. There are hundreds of Jenkins plugins to choose from. As you get to know Jenkins and create jobs, you’ll start realizing you’re doing the same steps for multiple jobs and realize there must be a plugin for this and chances are, you’re correct. Any pain points or processes people do over and over, chances are there are plugins available to make these much easier and idiot proof. I like idiot proof!

================================================================================

https://xebialabs.com/glossary/ (Complete)

================================================================================

# Useful Jenkins Plugins.

### Use these plugins to get the most out of Jenkins.

##### **Atish Narlawar**

December 3, 2014

The [Jenkins CI tool](http://jenkins-ci.org/) is backed by a very strong and active open source community, which has developed hundreds of useful plugins. These plugins eliminate the need to create fancy, half tested logic to accomplish tasks, solving common problems with minimal pain and also promote reusability across projects.

While the number and quality of plugins grow every day, we wanted to share a list of helpful plugins that we've discovered along the way.

[**Build Flow.**](https://wiki.jenkins-ci.org/display/JENKINS/Build+Flow+Plugin)Managing a pipeline in Jenkins requires moderate configurations across all the jobs participating in the flow. Configurations can combine different types, including parameterized build, parallel builds, joins or downstream wait. With the addition of more jobs, it gets quite complicated for further updates. The build process is then scattered across all of those jobs and is very complex to maintain. Individual jobs are meant to carry the responsibility for their unique task, but with this pipeline configuration they also carry connection details which pollute their goal. In this scenario, Build Flow Pipeline comes to the rescue. This plugin is designed to handle complex build workflows (aka build pipelines) as a dedicated entity in Jenkins. Build Flow enables you to define an upper level Flow item to manage job orchestration and link up rules, using a dedicated DSL. This DSL makes the flow definition very concise and readable. Main jobs don’t know anything about the pipeline as everything is externalized to the flow.

The only thing missing with this plugin is it doesn't have much support for graphical visualization on monitors. There is a Build Graph View Plugin but it wasn’t intended for this use. If graph view doesn't matter to you, then this can be the best plugin to try out. Otherwise go with regular scattered configuration.

[**Build Monitor.**](https://wiki.jenkins-ci.org/display/JENKINS/Build+Monitor+Plugin)Visibility is the core aspect of the Jenkins CI tool. It being understood that CI is a part of collective ownership, everyone should know that if something breaks, some part of it is not working consistently. To make this a reality, having that projected in a monitor visible to all is a welcome move. Build Monitor Plugin provides a highly visible view of the status of selected Jenkins jobs. It easily accommodates different computer screen sizes and is ideal as an Extreme Feedback Device to be displayed on a screen on your office wall.

[**Build Name Setter.**](https://wiki.jenkins-ci.org/display/JENKINS/Build+Name+Setter+Plugin)Every build comes with a build number, but most of the time its hard to identify a particular build with a build number. Many types of builds get executed with an update from revision number by source code, so it would be great deal easier to use revision number as a unique identifier. In  order to visualize the actual revision numbers within Jenkins, the Build Name Setter Plugin can be used instead. This makes it easier to identify builds by revision number instead of by build number.

[**Build Pipeline.**](https://wiki.jenkins-ci.org/display/JENKINS/Build+Pipeline+Plugin)As stated earlier, visibility is the core aspect of Jenkins pipeline to be successful in terms of its value.This plugin helps renders upstream and downstream connected jobs that typically form a build pipeline in a very nice fashion on the monitor, big monitors especially. It also offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins. This would help other decision making teams(Non Developers like QA, BA or other kind of authority) to continue the pipeline.

[**Build Timeout.**](https://wiki.jenkins-ci.org/display/JENKINS/Build-timeout+Plugin)Build-timeout is simple, but necessary.

[**Clone Workspace SCM.**](https://wiki.jenkins-ci.org/display/JENKINS/Clone+Workspace+SCM+Plugin)This plugin makes it possible to archive the workspace from builds of one project and reuse them as the SCM source for another project. This is useful in terms of avoiding space issues, and avoiding copying over the same source code which is not absolutely needed in a couple of cases, resulting in faster execution and feedback.

[**Cobertura.**](https://wiki.jenkins-ci.org/display/JENKINS/Cobertura+Plugin)This plugin integrates Cobertura coverage reports to Jenkins. If testing framework in your build job creates reports aligned to xUnit standards, this can be a handy plugin to project the progress in nice charts and detailed format.

[**Cucumber Test Result.**](https://wiki.jenkins-ci.org/display/JENKINS/Cucumber+Test+Result+Plugin)This plugin allows you to show the results of Cucumber tests within Jenkins.This plugin really helped the team to accommodate the BDD development style. Cucumber plugin formats the raw reports into very nice and visible charts, which can be easily understood by Non Tech folks like BA and Product owners.

[**Delivery Pipeline.**](https://wiki.jenkins-ci.org/display/JENKINS/Delivery+Pipeline+Plugin)This plugin visualizes Delivery Pipelines (Jobs with upstream/downstream dependencies). This is somewhat similar to Build pipeline plugin but has avery lean design. This is the favorite plugin across the teams here in my company.

[**Email Extension.**](https://wiki.jenkins-ci.org/display/JENKINS/Email-ext+plugin)This plugin is a replacement for Jenkins's email publisher. This is more advanced plugin with better features for sending build status messages. Please make sure you set up very granular and effective means of communication to the relevant team members, rather than whole team.

[**Git.**](https://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin)This plugin integrates GIT with Jenkins. This plugin also adds the feature of notifying the build about git code base updates using REST API. This is a very useful feature where Git Hooks can notify the builds immediately, rather than the build waiting for the Git poll interval.

[**Heavy Job**.](https://wiki.jenkins-ci.org/display/JENKINS/Heavy+Job+Plugin) During concurrent execution of jobs, Heavy Job Plugin can be used to allocate all available executors on that node in order to ensure exclusive access to all the local repositories.

[**HipChat.**](https://wiki.jenkins-ci.org/display/JENKINS/HipChat+Plugin)This plugin is a HipChat notifier that can publish build status to HipChat rooms. This plugin add more visibility and awareness in terms of build status for the team, resulting in a better sense of collective ownership.

[**JaCoCo.**](https://wiki.jenkins-ci.org/display/JENKINS/JaCoCo+Plugin)This plugin allows you to capture code coverage report from JaCoCo. Jenkins will generate the trend report of coverage. This plugin is the fork of the [Emma Plugin]. A big part of the code structure comes from it, however, it is completely refactored. It also includes functionality similar to the [Emma Coverage Column] which allows you to include a column in Dashboards which displays the latest overall coverage numbers and links to the coverage report.

[**Join.**](https://wiki.jenkins-ci.org/display/JENKINS/Join+Plugin)If you are thinking of setting up parallel jobs in pipeline, you may end up in a scenario known commonly as  'diamond' shape project dependency. It means there is a single parent job that should start several downstream jobs. Once those jobs are finished, a single aggregation job runs. This plugin allows a job to be run after all the immediate downstream jobs have completed. In this way,  the execution can branch out and perform many steps in parallel, and then run a final aggregation step just once after all the parallel work is finished. More complex interactions are not possible with this plugin.

[**NodeLabel Parameter.**](https://wiki.jenkins-ci.org/display/JENKINS/NodeLabel+Parameter+Plugin)NodeLabel Parameter Plugin can be used to assign the cleanup jobs to the specific nodes. This plugin adds two new parameter types to job configuration - node and label. This allows it to dynamically select the node where a job/project should be executed.

[**Parameterized Trigger.**](https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Trigger+Plugin)This plugin lets you trigger new builds when your build has completed, with various ways of specifying parameters for the new build. You can add multiple configurations: each has a list of projects to trigger, a condition for when to trigger them (based on the result of the current build), and a parameters section.

[**Performance.**](https://wiki.jenkins-ci.org/display/JENKINS/Performance+Plugin)This is very handy plugin in terms of Performance testing. This plugin integrates JMeter reports, JUnit reports, work output, and Iarge reports into Hudson.

[**Priority Sorter.**](https://wiki.jenkins-ci.org/display/JENKINS/Priority+Sorter+Plugin)In scenarios where multiple pipelines execute in parallel,  build steps of different pipelines are often expected not to get executed in a random order. For example, deploying artifacts of a pipeline to a live server cannot be succeeded by deploying artifacts of another pipeline earlier than a smoke test has run for the already deployed artifacts. This can be guaranteed by assigning a higher priority to the smoke test job using the Priority Sorter Plugin.

[**Rebuilder Plugin.**](https://wiki.jenkins-ci.org/display/JENKINS/Rebuild+Plugin)Sometimes a step in the pipeline might fail because of some technical error that is not related to the associated revision like Jenkins restart, or a memory issue. In order to trigger a rebuild of the failed downstream job the pipeline parameters revision number and build number need to be specified manually, which is a bit awkward. Here the Rebuilder Plugin comes in handy. It facilitates rebuilding a job with the same parameters as the failed build.

[**Shared Workspace.**](https://wiki.jenkins-ci.org/display/JENKINS/Shared+workspace+plugin)This plugin allows you to share workspaces for Jenkins jobs with the same SCM repos. It saves some disk space and repetitive steps, if you have different jobs with identical repos. The importance of this plugin is not well understood by the developers, considering memory is cheap. But following standard practices and promoting reusability can be fruitful in the future.

[**SSH Agent.**](https://wiki.jenkins-ci.org/display/JENKINS/SSH+Agent+Plugin)This plugin allows you to provide SSH credentials to builds via a ssh agent in Jenkins.

[**SSH.**](https://wiki.jenkins-ci.org/display/JENKINS/SSH+plugin)You can use the SSH Plugin to run shell commands on a remote machine via ssh.

[**SSH Slaves.**](https://wiki.jenkins-ci.org/display/JENKINS/SSH+Slaves+plugin)This plugin allows you to manage slaves running on Unix machines over SSH. It adds a new type of slave launch method. This launch method will open a SSH connection to the specified host as the specified username. Once it has a suitable version of java, it copies the latest slave.jar via SFTP. It starts the slave process.

[**Subversion.**](https://wiki.jenkins-ci.org/display/JENKINS/Subversion+Plugin)This plugin adds the Subversion support (via SVNKit) to Jenkins.

[**Thinbackup.**](https://wiki.jenkins-ci.org/display/JENKINS/thinBackup)This plugin simply backs up the global and job specific configurations (not the archive or the workspace). One of the main feature is automated backups. This is far better then Backup Plugin.

[**Throttle Concurrent Builds.**](https://wiki.jenkins-ci.org/display/JENKINS/Throttle+Concurrent+Builds+Plugin)The Throttle Concurrent Builds Plugin can be used to define throttle categories and restrict concurrent execution of jobs by assigning them to the same throttle category.

# [Wall Display.](https://wiki.jenkins-ci.org/display/JENKINS/Wall+Display+Plugin) A wall display that shows job build progress in a way suitable for public wall displays. Rendering is performed using ajax based on REST API calls, so it requires no page refreshes.

# [Workspace Cleanup.](https://wiki.jenkins-ci.org/display/JENKINS/Workspace+Cleanup+Plugin)One of the common recommended task of Jenkin Job is to cleanup itself after build is finished to maintain immunity for next build. I have often seen teams use  ‘rm-rf\*’ in script which works fine, but doesn't cover bad scenarios. This plugin is meant to save us from this granular job. This plugin deletes the project workspace after a build is finished

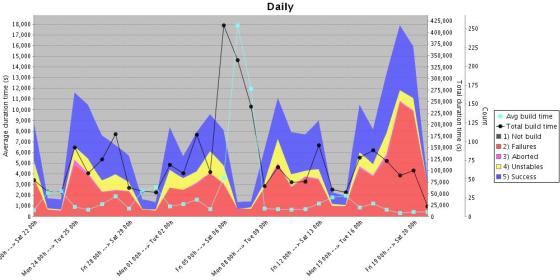
# ========================================================

# 15 must have Jenkins plugins to increase productivity

[vishal sahasrabuddhe](https://devops.com/author/vishal-sahasrabuddhe/)BY [VISHAL SAHASRABUDDHE](https://devops.com/author/vishal-sahasrabuddhe/) ON JANUARY 15, 2015 [6 COMMENTS](https://devops.com/15-must-jenkins-plugins-increase-productivity/#disqus_thread)

Jenkins is a vastly used continuous integration tool for the IT industry which is built and used around plugins. It’s core is the Jenkins tool and there are 100’s of plugins available to enhance its power and usability.

Let’s take a look at a few of the best plugins to increase the productivity.

1. **Global Build Stats Plugin–**It’s essential to know your current capacity, usage and capability to prepare for capacity planning or system requirements, because the very first question is “what is your current capacity?”        [](https://i2.wp.com/devops.com/wp-content/uploads/2015/01/global-build-stats.jpg)

You need to know how many builds are happening on a daily and weekly basis. How much time is taken by various builds, what is the waiting period. This plugin give you enough data to answer the question. This plugin can provide you data as graph which can be further consumed.

1. **Job GeneratorPlugin–**In big or growing organizations, its a bit difficult to maintain the jobs for a project when developers are working on various branches and releases. You want to give access to developers to create their own job and at the same time you don’t want developers to create arbitrary jobs which may not fall under company standards. This plugin gives you flexibility to define templates and developers can create new jobs with the help of job generator template. Configuration access can be disabled via roll-based authorization plugin.
2. **Disable-failed-job –**In rapid development environments failing jobs due to various unnecessary reasons are a pain, It’s difficult to keep track of such jobs and disable/delete them. This plugin can solve the problem by defining the upper number of consecutive failed builds and then disable automatically.
3. **Embeddable-build-status –**This plugin can give you a link which can be pasted anywhere (example github project) to expose the status of the build and users can get the current state of the job while looking at project.
4. **Exclusion –**This plugin enables you to handle conflicts between jobs. You can assign a resource (or lock) in multiple jobs, when build is executed, it will acquire the lock and other builds (if fired) will wait until lock is released.
5. **GitHub/GitLab Pull Request Builder-**This is one of the best plugins I found to provide support to automate code review up to a certain extent in github/gitlab. This plugin does amazing job once defined for the project. For any new pull request this plugin does fox merge and runs the build on the code as well as gathers the necessary static analysis (if configured) or build results and comment back the status to pull request. This helps reviewers get an idea of the health of the code which is going to be merged. You can even define automatic merge if build passes.
6. **Hudson Extended Read Permission Plugin-**During initial days of Jenkins setting configuration access was not available to any developer to make sure they do not change anything but developers used to ask to view configurations to know how job has been setup, they may want to see the build steps. Extended read plugin can provide option to developer to view the configuration without giving them write access.
7. **Post+build+task –**There may be a need for performing some actions on the basis of the results of a build, for example if build passed you may want to upload artifact(ex debian) to some repo (apt) or perform some packaging part or similar. In case of failure you may want to roll back something (like release) . This plugin helps you to define the pass/fail criteria and let’s you decide what to do after that.
8. **JDK Parameter Plugin–**This plugin is useful for an organization where many projects are using different versions of java. This plugin let’s you choose the java version during run time of the build.
9. **Job Configuration History Plugin-**Ah! This is one of my favorite plugins. This plugin lets you keep track of config changes in each build including who did it. You can easily revert back to any previous config if you want.
10. **Multiple SCMs plugin–**Default SCM section provides only one source control tool/URL option, what if you want to check out from more than 2 repos from multiple source control tools (like svn and git). This plugin will come in handy in such scenarios. This plugin will facilitate users to add any number of SCM URLs to checkout the code.
11. **Parameterized Trigger plugin**– Another one of my favorite plugins. This plugin allows you to have user input as a variable and use on run time. This is the most used plugin in dynamic environments where you have lots of options and user-defined values to be used in the build which may keep changing.
12. **Pre SCM BuildStep Plugin-**Just like post build task plugin, you may have requirements to perform some action even before checkout happens for the job, for example you may want to perform merging of the branch before the build and then checkout. This plugin can come in handy in various conditions and gives you flexibility when running the job.
13. **SCM Sync Configuration Plugin-**backup, this is the most important task of any administrator. Without regular backup, the whole system cannot be reliable; this plugin provides you features to backup live Jenkins configs to any source control tool. It will keep committing the config files (including Jenkins and jobs) to SCM repository as soon as there is any change.
14. **Configuration Slicing Plugin –**This plugin comes  in very handy when you want to make bulk changes in multiple jobs. This plugins allows you to change values of various fields like email, timer, shell script, configurations etc at one go.

.

==================================================================

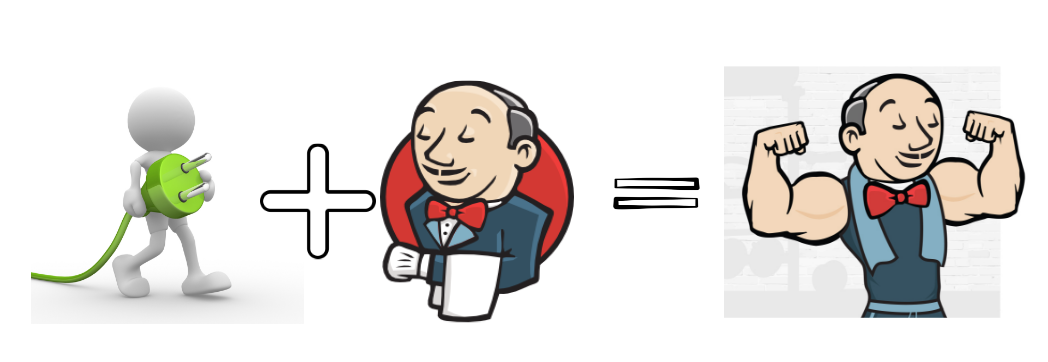
# 7 Must-Have Jenkins Plugins

* [AWS](http://www.tothenew.com/blog/category/technology/aws-2/), [DEVOPS](http://www.tothenew.com/blog/category/technology/devops-technology/), [TECHNOLOGY](http://www.tothenew.com/blog/category/technology/)

29 / JUL / 2016 BY [RAJDEEP SINGH](http://www.tothenew.com/blog/author/rajdeepsingh/) [1 COMMENTS](http://www.tothenew.com/blog/7-must-have-jenkins-plugins/#comments)

Share this blog

EmailTwitterFacebookLinkedInGoogle+

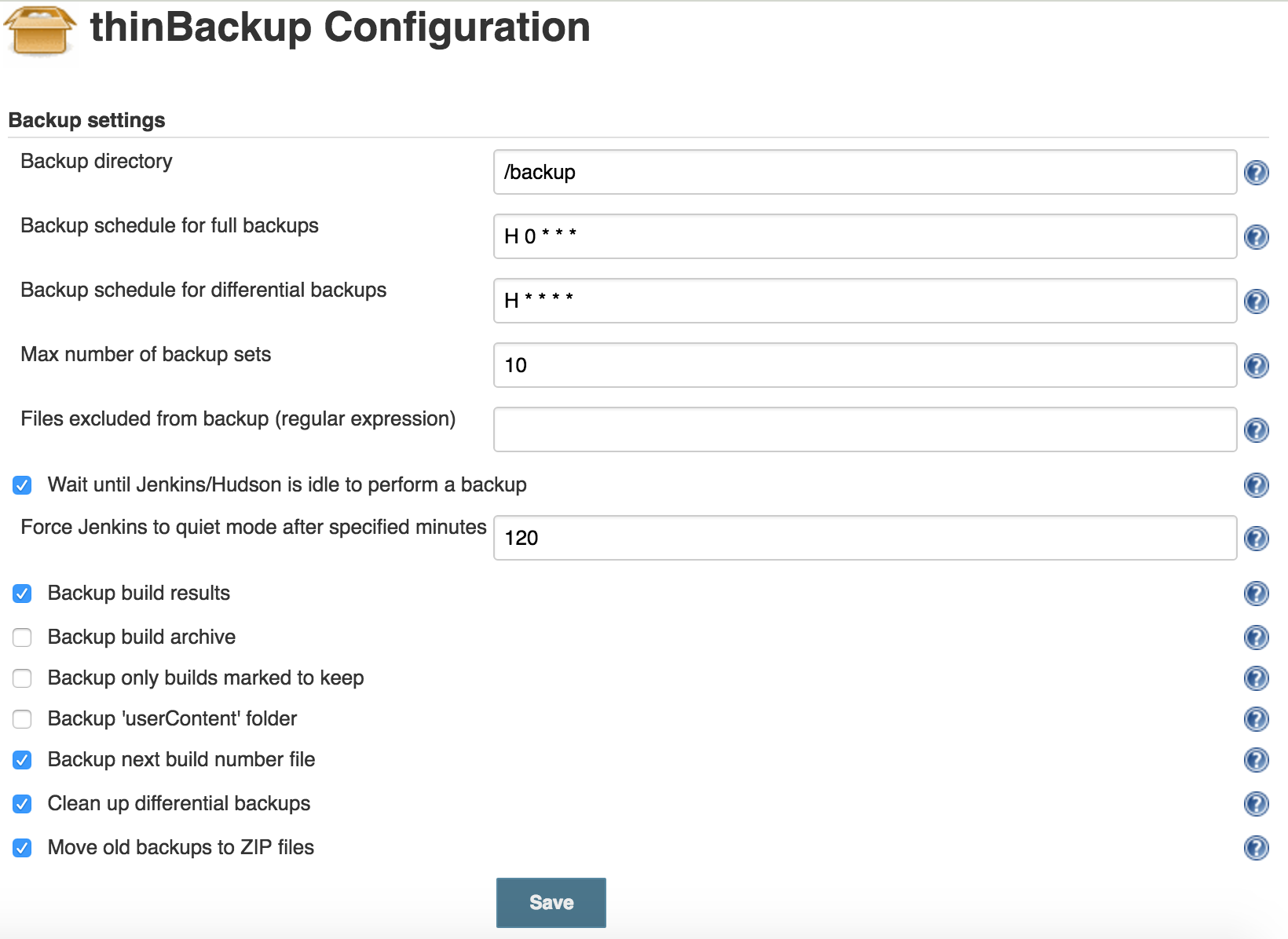


The Jenkins is the best freeware tool to fasten your development cycle. It helps you to increase team productivity with CI (continuous integration) and CD (continuous delivery). In the development phase, you have to test changes more frequently. A single-click build and deploy job reduce time to validate your modified codebase, and you can get rid of all manual steps overhead.

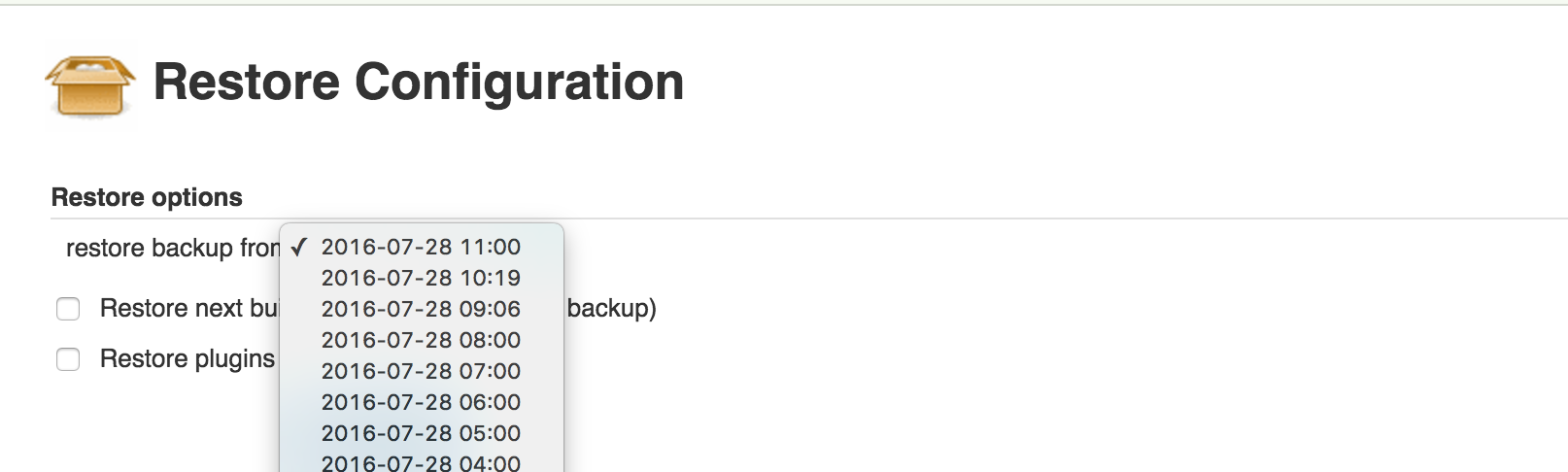
The plugins increase Jenkins capabilities and add some of the cool features which help you to increase productivity make your build space more secure. There are lot’s of plugins available but I would like to share 7 plugins which you must install.

**1. ThinBackup plugin**

This is the first plugin you should install after login into Jenkins. It helps you to backup your jobs, configuration, build history etc. It is easy to configure and you can tweak backup setting with the help of a backup manager. It has the capability to trigger backup job manually or use can schedule daily automated backup. To configure this plugin navigate to **Manage Jenkins**–> **thinBackup** –> **setting:**

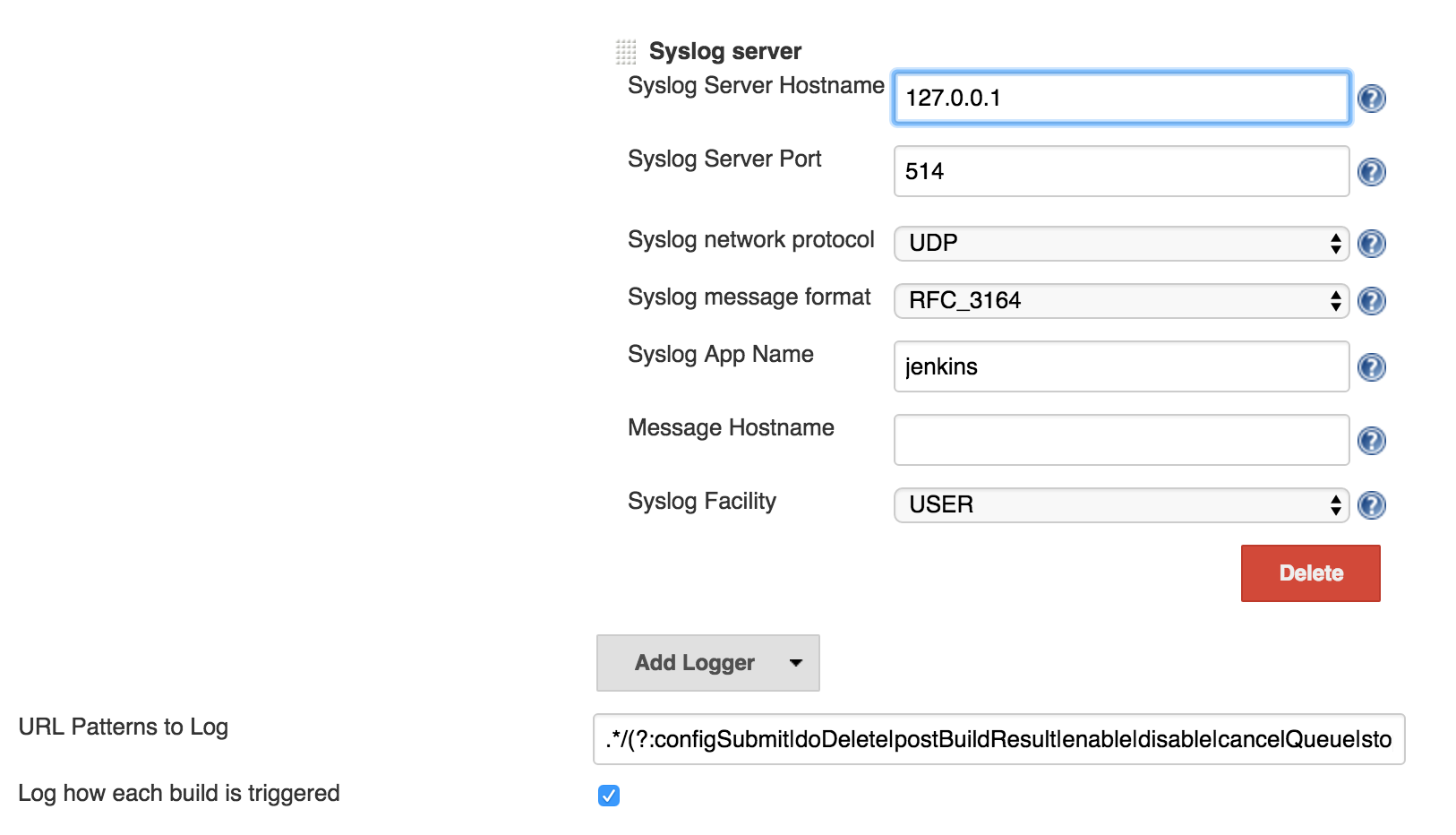


It all depends on how you want to schedule backup as with above setting you will have backup of each hour and whenever you want to restore deleted job select **Jenkins –> thinBackup–> Restore**. In a drop down menu it will list you all backups available to restore:



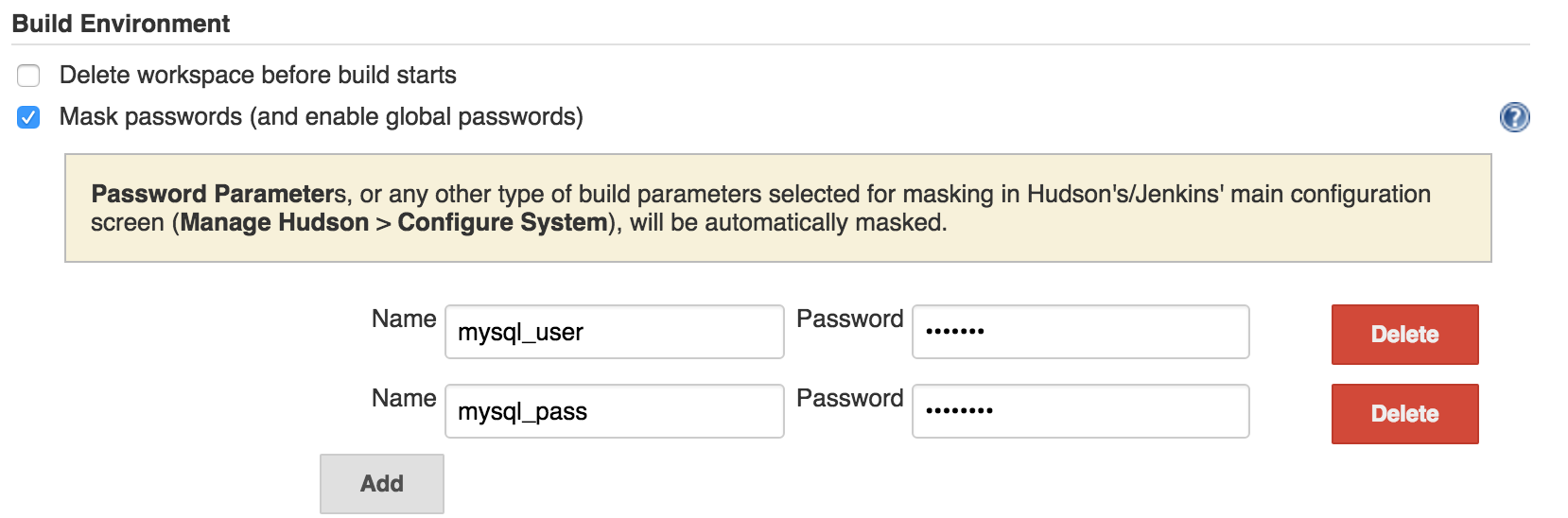
**2. Audit Trail**

When there are multiple users using Jenkins to manage jobs and you want to track footprints for auditing purpose, this plugin will help you. It stores all user activity in a log file. This log file can be stored on local machine or you can archive this to a remote location with Syslog. If you are interested in recording only certain events like create, delete this can also be configurable. To select configuration go to **Manage Jenkins –> Configure System** and on this page look for Audit Trail section:

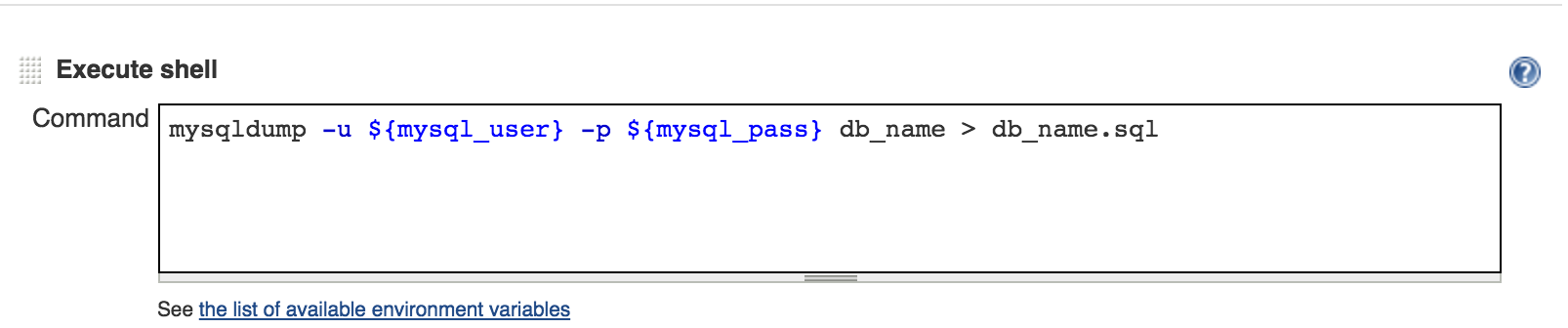


**3. Mask Passwords Plugin**

This plugin masks your credentials which you are defining as plain text format in job configuration console. It helps you to tightly restrict access to sensitive information. While creating a job in **“Build Environment”** section select check-box in front of **“Mask passwords”**and define your credentials:

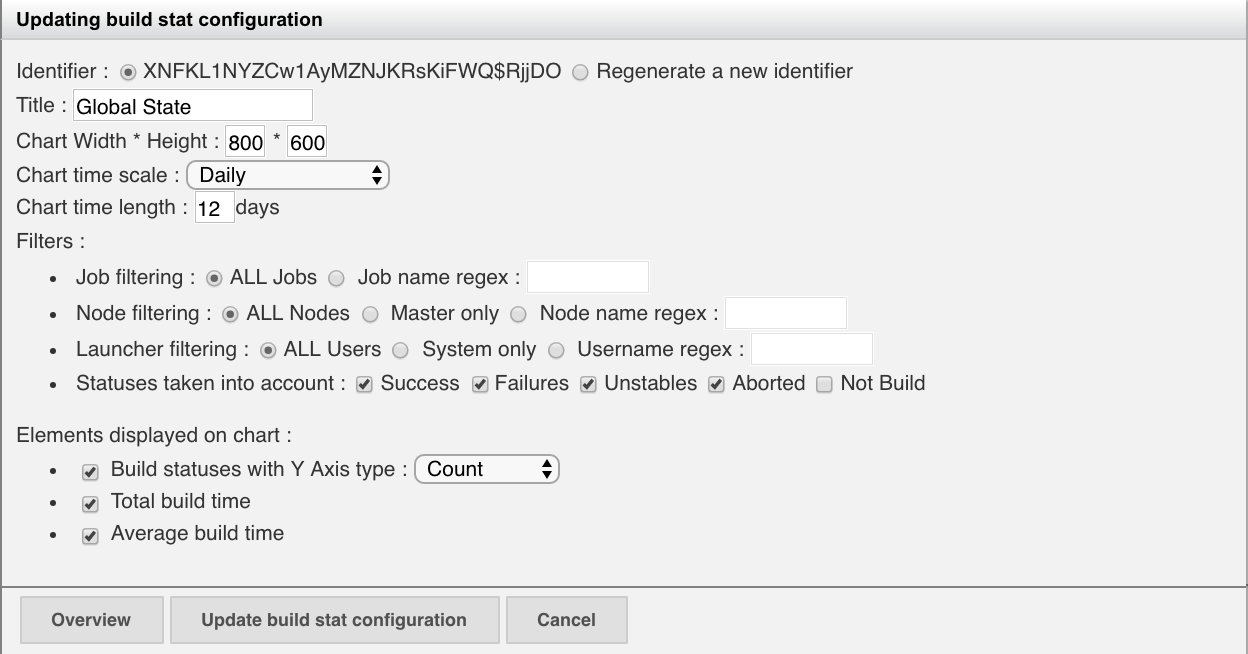


These credentials are invoked in as a variable and ciphered in the job configuration file:

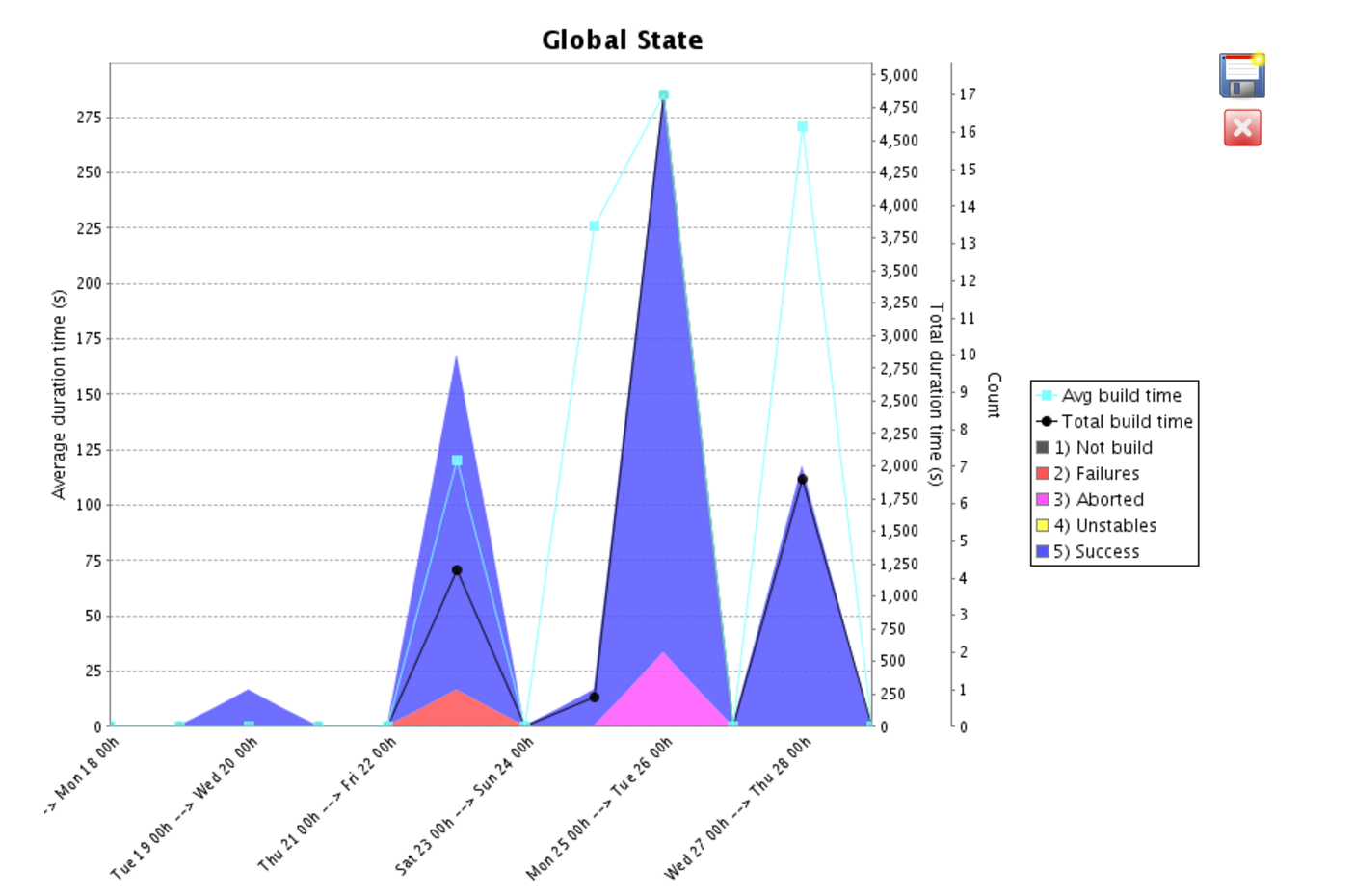


**4. Global Build Stats Plugin**

When you are leveraging Jenkins and with the growth of infrastructure it is most important to keep an eye on its capacity. So you can scale Jenkins before things go out of control. This plugin is useful to create daily/weekly/monthly report of build status. Once you have installed it you need to set graph property to initialize chart first time from **Manage Jenkins –> Global Build Stats –> Create a new chart configuration:**

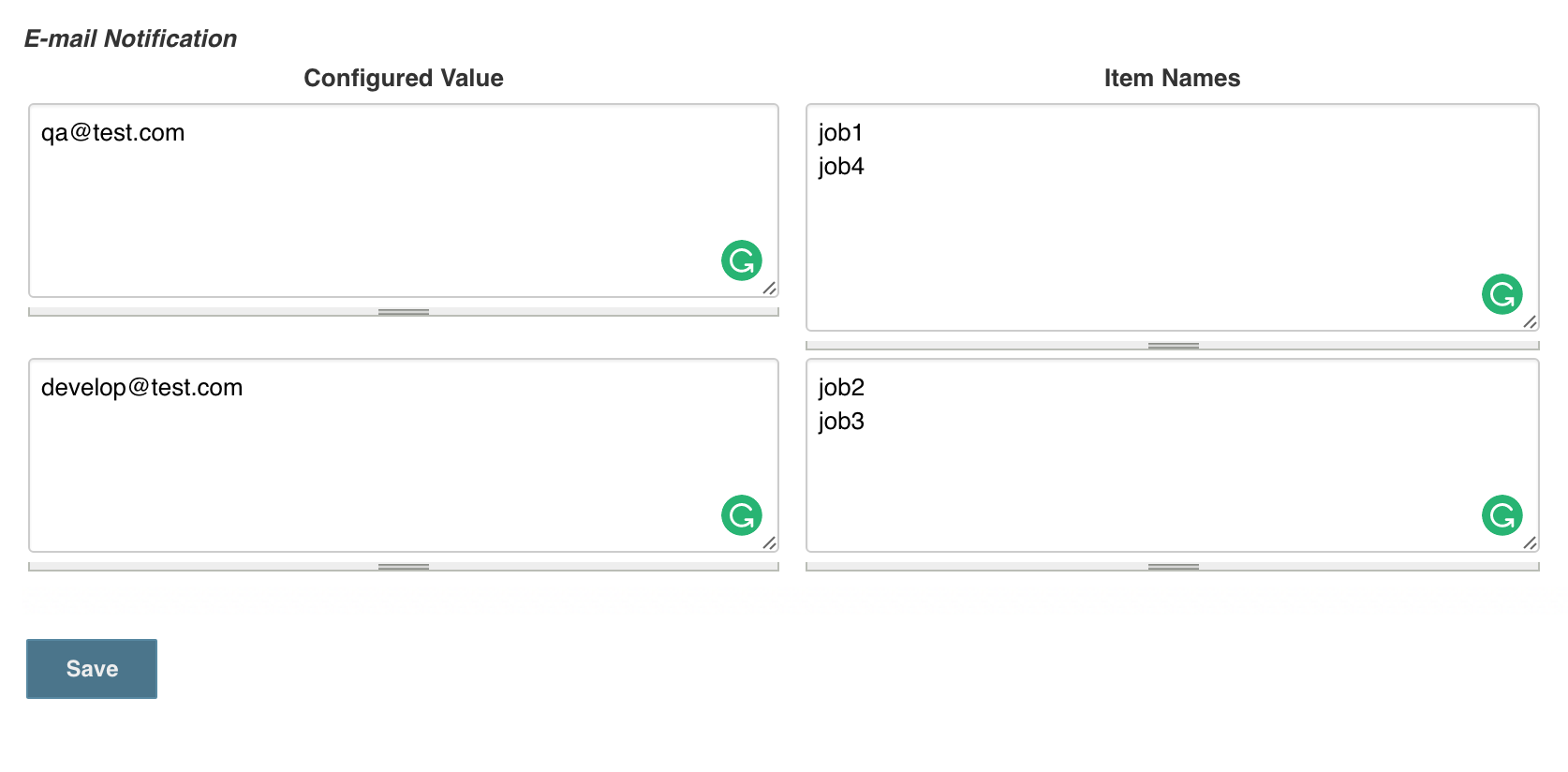


There are multiple options you can set from your choice but don’t forget to enable **“Average build time”** and **“Total build time”**. Once you save this configuration a historical chart will be available and you can plan your system scalability.



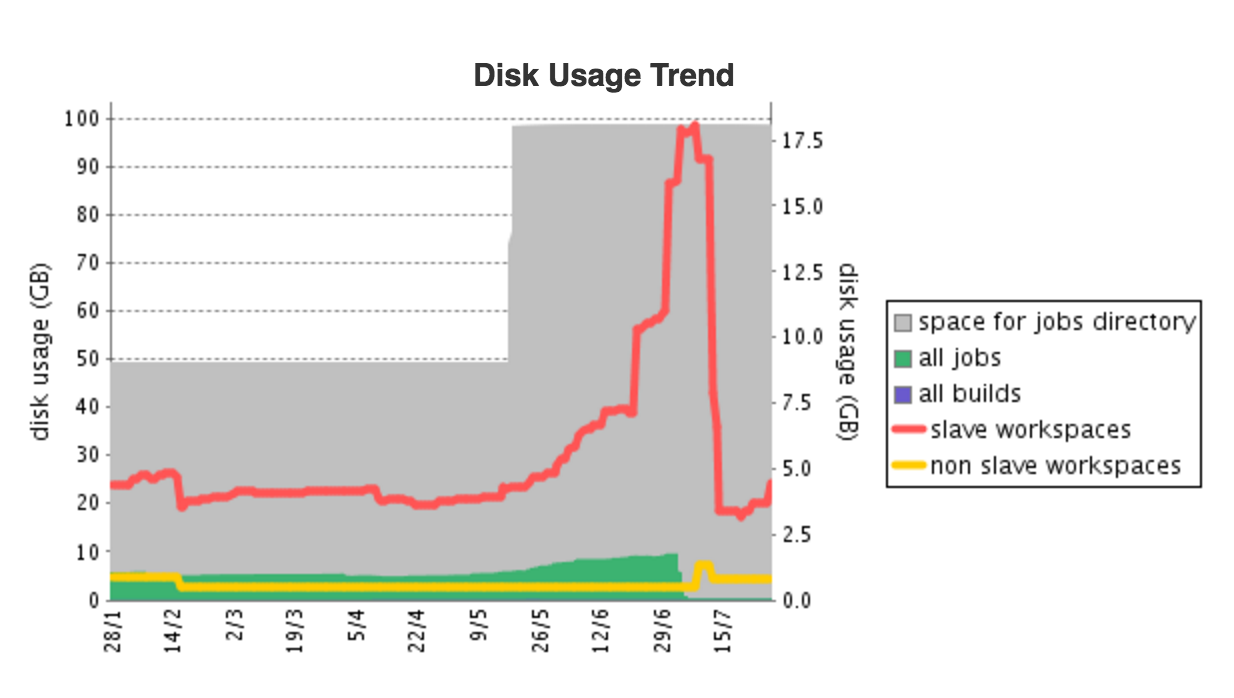
**5. Configuration Slicing Plugin**

You are doing well with your setup and one day you have to modify email id in all jobs then it will take a lot of time to finish your task. You can use this plugin to make your work more productive as it has the functionality to perform mass configuration of project properties. However, as of now it supports only selected configuration changes but still is useful to have in your installed list. It shows up under **“Manage Jenkins” –> “Configuration Slicing”**. In below example, we have changed the email address for multiple jobs at once.

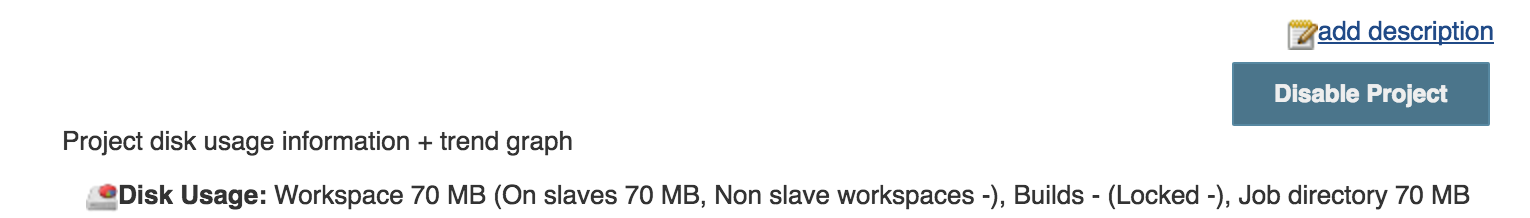


**6. Disk Usages**

This plugin will display you disk used by the project and keep updating the values after 1 hour. To view disk usages trend graph of all jobs you can select **Disk usage** from Jenkins dashboard left panel. You can easily find out sudden change in disk utilization and can correct it if there is something wrong going on.

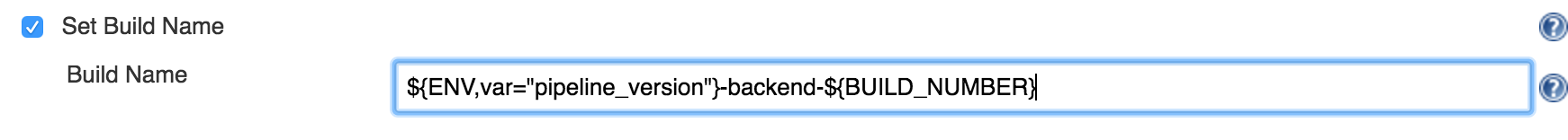


If you want to check usages of a particular job the detail information of used space and it’s workspace also be available on the **project page** of each job.



**7. Build Name Setter Plugin**

When you install this plugin you can change default build name from #1, #2, #3 to something of your choice. Using this you can segregate the multiple jobs based on env/project name. It is configurable in job configuration page under build environment section select **Set Build Name and set name which makes suitable for you.**



I would strongly recommend you to never use a fix name and always include a dynamic identifier like **$(BUILD\_NUMBER)**.

The choice of plugins differ from use case to use case but I found these plugins most useful and make sense for everyone. There are more plugins available and free to use so go ahead and spread your wings.

=======================================================

# Top 5 Jenkins Plugins

### Whether you want to to automate and standardize your Jenkins installations or simply get an overview of your build process — there's a plugin for that.

[**[](https://dzone.com/users/2948971/leakaram.html)**](https://dzone.com/users/2948971/leakaram.html)**by**

[**Lea Maya Karam**](https://dzone.com/users/2948971/leakaram.html)

**·**

**Feb. 09, 17 · [DevOps Zone](https://dzone.com/devops-tutorials-tools-news)**

**Like** [**(3)**](https://dzone.com/articles/5-best-jenkins-plugins-recommended-by-our-team)

**Comment (0)**

**Save**

[**Tweet**](https://dzone.com/articles/5-best-jenkins-plugins-recommended-by-our-team)

**8,239 Views**

Join the DZone community and get the full member experience.

[**JOIN FOR FREE**](https://dzone.com/articles/5-best-jenkins-plugins-recommended-by-our-team)

The Nexus Suite is uniquely architected for a DevOps native world and creates value early in the development pipeline, provides precise contextual controls at every phase, and accelerates DevOps innovation with automation you can trust. [Read how in this ebook](https://dzone.com/go?i=222229&u=https%3A%2F%2Fwww.sonatype.com%2Faccelerate-devops-early-everywhere-at-scale-ebook%3Futm_campaign%3Ddzone%26utm_source%3Dearly%2520everywhere%2520ebook).

[Jenkins](https://jenkins.io/) is an open-source automation server. More specifically, it's a [Continuous Integration](https://apiumhub.com/blog/benefits-of-continuous-integration/) server. This tool is often seen as a middle man between your code and your build server because it regularly looks for changes on your server. Once changes are found, it sends them to the build server. It has become very popular among developers, and that’s why I believe it would be nice to have a short list of the Jenkins plugins that we at [Apiumhub](https://apiumhub.com/) love the most.

## 1. Job DSL Plugin

Last release date: January 15, 2017.

Many of us use the UI of Jenkins to configure jobs, but sometimes, you end up with so many jobs that it becomes difficult to maintain and working with UI becomes slow. The Job DSL plugin is great for this situation — it’s quite popular and exceptionally well-documented.

It allows users to create “projects” by using a Domain-Specific Language (DSL). It enables you to automate your Jenkins installations and standardize them by pushing job creation into a script.

Basically, it uses the jobs that are created and makes templates with them. Let’s say that you create other jobs according to a Groovy script. So, if you create one job, write the script, and run that job, you’ll have the other jobs related to the last one created. Keep in mind that not all jobs that are created will directly run — only the main one created will run.

## 2. Build Pipeline Plugin

Last release date: December 8, 2016.

We’re big fans of Continuous Integration, and we use Jenkins because it provides good support for providing Continuous Delivery. Another one of our favorite Jenkins plugins is the Build Pipeline plugin because it enables us to form a chain of jobs that are based on their dependencies.

The chain of jobs that you create all pass through the quality assurance steps that you can choose to trigger by a bunch of automated steps or manual tests. After they pass through all of this, they are automatically deployed into production. This allows you to integrate external processes such as reviews before deployment.

It’s amazing because it helps you get an overview of your build process. You can see the history, statuses, and where each has arrived in the chain.

## 3. Delivery Pipeline Plugin

Last release date: January 13, 2017.

We are all aware of the fact that in Continuous Delivery, feedback and visualization of the delivery process are very important. As we mentioned earlier, we use (and love) the Build Pipeline Plugin. The third on our topJenkins plugins list is the Delivery Pipeline plugin. It’s nice because you can visualize one or more delivery pipelines.

In fact, a pipeline has various stages. Obviously, building, unit testing, packaging, and analyzing the pipeline can become very long when each job is a stage, so here, you can group jobs into the same stage, making your life much easier.

## 4. Copy Artifact Plugin

Last release date: July 24, 2016.

In some situations, your build jobs don’t really know much about each other because they are isolated (mainly where we are not using a dependency management system).

The copy Artifact plugin will add a build step to copy artifacts from another project. It enables you to copy the needed files from one job to another that can run some tests. You basically choose from which build you want to copy artifacts, and you have good control during that process.

## 5. JIRA Plugin for Jenkins

Last release date: December 20, 2016.

This plugin integrates JIRA to Jenkins. We find it very useful because we use JIRA all the time and we absolutely need it to work on a project in an organized way between the teams.

It’s practical because you can display Jenkins builds inside JIRA. It will help you in maintaining a proper, organized way of working on a lengthy project and in minimizing the mistakes that occur without JIRA.

I will end this list by saying that for people that are looking to increase their productivity, Jenkins is definitely here to help. It has many features and great Jenkins plugins. It's helpful when it comes to reducing workloads through automation and organization. Installations are, in general, quite simple and quick, and you have many many plugins to choose from — whatever suits your needs. If you’re interested, here’s a list of [17 tools used by software developers](https://apiumhub.com/blog/software-development-tools/) on a regular basis.

========================================================================

[5 Jenkins Plugins You Need to Use If You Want to Increase Your Productivity](https://www.tetratutorials.com/blog/528483/5-jenkins-plugins-you-need-to-use-if-you-want-to-increase-your-productivity)

**Manuj Aggarwal**

03 March, 2017

Whether you want to get started in the exciting field of DevOps or you want to enhance your current continuous development, continuous deployment and continuous delivery pipelines - Jenkins is an in indispensable tool for you.

If you are a web developer, mobile developer, and IT engineer or even a technology investor - knowledge about this incredible tool will be invaluable for you.

**What Is Jenkins?**

Jenkins is one of the most popular continuous integration tools used in software development.

It is an open source automation server written in Java. You can use Jenkins to build and test your software projects continuously making it easier for developers to integrate changes to the project, and to make it easier for users to obtain a fresh build.

You can integrate Jenkins with numerous testing and deployment technologies because it can handle any type of automated build or continuous integration - from a small project to large behemoth distributed projects.

**But, What Are The Plugins?**

Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit an organization (or user) specific needs.

There are **over a thousand different plugins** which can be installed on Jenkins to integrate various build tools, cloud providers, analysis tools, and much more.

Although Jenkins have numerous plugins and new ones are added daily, **these are the 5 must-have** plugins that we consider will be helpful for you:

1. [**Global Build Stats Plugin**](https://wiki.jenkins-ci.org/display/JENKINS/Global+Build+Stats+Plugin)**:**This plugin is essential to know your current capacity, usage and capability to prepare for capacity planning or system requirements, because it creates a daily/weekly/monthly report of build status.
2. [**Configuration Slicing Plugin**](https://wiki.jenkins-ci.org/display/JENKINS/Configuration+Slicing+Plugin): It becomes really useful when you want to make a mass configuration change because it allows you to alter values of various fields like email, timer, shell script, and more at one go.
3. [**Job Configuration History Plugin:**](https://wiki.jenkins-ci.org/display/JENKINS/JobConfigHistory+Plugin)It lets you keep track of configuration changes in each build, including the information on who did it. Allowing you to easily revert back to any previous configuration if you want.
4. [Scriptler:](https://wiki.jenkins-ci.org/display/JENKINS/Scriptler+Plugin) Allows you to store/edit groovy scripts and execute them on any of the slaves/nodes or master. You can execute any groovy code in Jenkins context. Thanks to this plugin you can store scripts, share them with other users and use them easily within builds, performing many Jenkins customized actions almost on the fly.
5. [**Build Failure Analyzer:**](https://wiki.jenkins-ci.org/display/JENKINS/Build+Failure+Analyzer)This plugin analyzes the causes of failed builds and presents them on the build page. It does this by using a knowledge base of build failure causes that it’s built from scratch. It’s a very useful plugin that makes you able to see the reasons for particular build failures within a job page and history, after you have created a sufficient knowledge base (which is simply a mapping from build log pattern to a problem name). Additionally, you can move to the problematic build log line with just one click.

============================================================

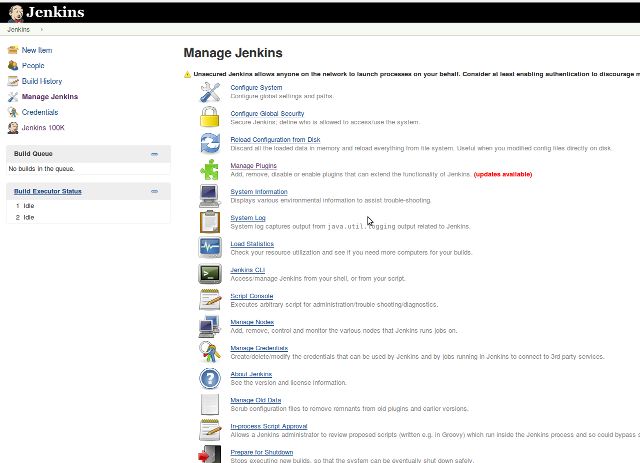
https://www.praqma.com/stories/top-jenkins-plugins/

============================================================

Jenkins plugins you want [1](http://continuousdev.com/2015/04/jenkins-plugins-you-want/#tc-comment-title)

[*2 Apr, 2015*](http://continuousdev.com/2015/04/02/)*in*[*Configuration Management*](http://continuousdev.com/category/configuration-management/)*/*[*System Integration*](http://continuousdev.com/category/system-integration/)*by [Piotr Oktaba](http://continuousdev.com/author/octos/" \o "View all posts by Piotr Oktaba)*

Share it](http://twitter.com/share?url=http://continuousdev.com/2015/04/jenkins-plugins-you-want/&text=Jenkins+plugins+you+want+by+%40poktaba)[](http://www.facebook.com/sharer.php?u=http://continuousdev.com/2015/04/jenkins-plugins-you-want/)[](https://plus.google.com/share?url=http://continuousdev.com/2015/04/jenkins-plugins-you-want/)[](http://www.linkedin.com/shareArticle?mini=true&url=http://continuousdev.com/2015/04/jenkins-plugins-you-want/)[](mailto:?subject=Jenkins%20plugins%20you%20want&body=%20http://continuousdev.com/2015/04/jenkins-plugins-you-want/)[](http://reddit.com/submit?url=http://continuousdev.com/2015/04/jenkins-plugins-you-want/&title=Jenkins%20plugins%20you%20want)

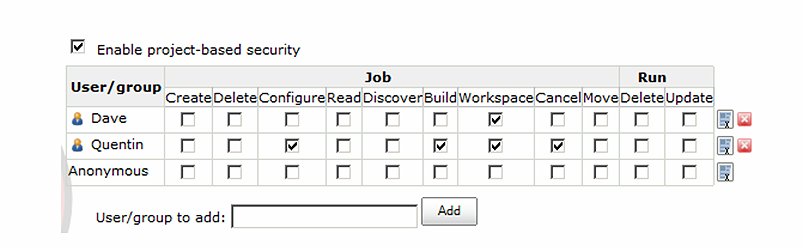
[](http://continuousdev.com/wp-content/uploads/2015/04/jenkins.jpg)

Jenkins is a very popular Continuous Integration server. Easy to install, easy to configure, easy to just use to build and test your project (if you have not used it before it is worth checking out for sure). However, whilst your project and company grow and/or you get more projects to maintain, its core features are not sufficient. Fortunately, a bunch of plugins can help. Installation and updates are automatic and simple. For a complete list of plugins, checkout the [official page](http://continuousdev.com/wp-admin/%20https:/wiki.jenkins-ci.org/display/JENKINS/Plugins). There are more than a thousand available. It is easy to just lose your mind whilst digging through them.

Below you can find the list of plugins that I have found the most useful. Some of them are must have, some just nice to have. It is a good place to start and explore all the rich Jenkins possibilities.

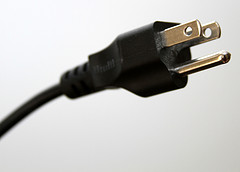
**Must have**

* *Job configuration History plugin* – “saves a backup of the configuration file of a job and of the system configuration for every change made. You can also see what changes have been made by which user if you configured a security policy”. Total must be – allows you to see what exact changes has been done to the configuration, by who, and revert them with just one click!
* *Scriptler* – “allows you to store/edit groovy scripts and execute them on any of the slaves/nodes or master”. You can execute any groovy code in Jenkins context. Thanks to the plugin you can store scripts, share them with other users and use them easily within builds. You can perform many Jenkins customized actions almost on the fly.
* *Build Failure Analyzer* – “This plugin analyzes the causes of failed builds and presents the causes on the build page. It does this by using a knowledge base of build failure causes that is built from scratch”. Very useful. After you have created a sufficient knowledge base (which is simply a mapping from build log pattern to a problem name) you are able to see reasons for particular build failures within a job page and history. Additionally, you can move to the problematic build log line with one click.
* *Nested view plugin* – “View type to allow grouping job views into multiple levels instead of one big list of tabs”. Definitely a must be if you are maintaining hundreds of jobs.
* *Build-timeout plugin* – “This plugin allows you to automatically abort a build if it’s taking too long. Once the timeout is reached, Jenkins behaves as if an invisible hand has clicked the abort build button.” Assures that one unfortunate developer change will not block a slave until someone notices the problem.
* *Matrix Authorization strategy plugin* – pre-installed with current version. Allows you to assign custom permissions to specific users, both globally and per job. Check out the below screen:

[](http://continuousdev.com/wp-content/uploads/2015/04/matrix-authentication.png)

**Nice to have**

* *Active Directory Plugin* – “With this plugin, you can configure Jenkins to authenticate the username and the password through Active Directory.” No need to maintain the users list separately.
* *LDAP Plugin* – pre-installed with current version. “Security realm based on LDAP authentication.” If you would like to use LDAP authentication of course.
* *Conditional build step plugin* – “A buildstep wrapping any number of other buildsteps, controlling their execution based on a defined condition.” Instead of putting complicated logic inside one build step, you can create clear and transparent configuration. You can also conditionally trigger other builds.
* *Email-ext plugin* – “This plugin allows you to configure every aspect of email notifications. You can customize when an email is sent, who should receive it, and what the email says.” Very flexible configuration. If you want to craft your Jenkins notification into a masterpiece and make sure the recipients do not ignore messages, you should check it out for sure.
* *HTML Publisher plugin* – “Html Publisher plugin is useful to publish the html reports that your build generates to the job and build pages.” One of its core features is that HTML reports are persisted per build. As long as you do not delete job history, you will have access to all published reports.
* *Multi-slave config plugin* – “This plugin allows administrators to configure, add and delete several dumb slaves at the same time.” If you maintain Jenkins with dozens of nodes, it is a must have. By allowing to perform operations on several nodes at the same time saves a lot of time.
* *Multiple SCMs plugin* – “Allows a job to check out sources from multiple SCM providers.” Not fully stable, but very useful. In many complicated environments checking out a few projects into one workspace might simplify the configuration and building process.
* *Naginator plugin* – “This plugin allows you to automatically reschedule a build after a build failure.” If your build depends on any external resources (it does for sure: DB, network, repository, etc) you can find it useful to not have to wait until a new change is coming to build. You can of course configure your job to build periodically but then you lose all the advantages of pool scm trigger.
* *TextFinder plugin* – “This plugin lets you search keywords in the files you specified and uses that to downgrade a successful build to be unstable or a failure.” If you encounter situations when your tools are not using exit code properly and you are not able to change it, you can workaround it with this plugin.
* *View Job Filters* – “Create smart views with exactly the jobs you want. Your smart views can automatically include or exclude jobs by using things like the SCM path or type, the job type, build statuses or trends or triggers, relevance to the logged-in user, email recipients, Maven configuration, job parameterization, and user permissions. Mix and match filters to narrow down to exactly what you want.” Self explanatory.
* *Cluster Statistics plugin* – “This plugin will measure the time in queue, time in build and show these statistics per node.” Simple and easy to use statistics to check your Jenkins overall and per node load.
* *Parameterized Trigger plugin* – “This plugin lets you trigger new builds when your build has completed, with various ways of specifying parameters for the new build. You can add multiple configurations: each has a list of projects to trigger, a condition for when to trigger them (based on the result of the current build), and a parameters section.” Nice to create simple build pipelines.
* *Workflow plugin* – this plugin deserves a separate post which will come soon. I will just advocate it a little by copying its introductory sentences: “Building continuous delivery pipelines and similarly complex tasks in Jenkins using freestyle projects and traditional plugins can be awkward. You need to mix Parameterized Trigger, Copy Artifact, Promoted Builds, Conditional Build Step, and more just to express what should be a simple script. The Workflow plugin suite attempts to make it possible to directly write that script, what people often call a *workflow*(sometimes abbreviated *flow*), while integrating with Jenkins features like slaves and publishers.” The idea is to define complex job pipelines/workflow in single Groovy script using embedded DSL.

[](http://continuousdev.com/wp-content/uploads/2015/04/plug.jpg)

*By*[*Samuel M. Livingston*](https://www.flickr.com/photos/39747297@N05/5229733647)*, on*[*Creative Commons*](https://creativecommons.org/)

19 plugins. Most of them useful in any Jenkins instance, without some I cannot imagine maintaining huge Jenkins servers. Try them yourself and evaluate the possibilities. I encourage you to spend half an hour per week to search for what plugins are available and how they can improve productivity of all people using Jenkins within your company. You will soon figure out that this will pay off.

=================================================================

# Jenkins plugins

**You can read more about each plugin using the SEARCH box at**[**Jenkins homepage**](https://wiki.jenkins-ci.org/display/JENKINS/Home)**.**

* ChuckNorris Plugin: Displays a picture of Chuck Norris (instead of Jenkins the butler) and a random Chuck Norris 'The Programmer' fact on each build page.

## Maintenance

* thinBackup: This plugin simply backs up the global and job specific configurations (not the archive or the workspace).
* JobConfigHistory Plugin: Saves copies of all job and system configurations.

## Monitoring

* Monitoring: Monitoring of Jenkins itself with JavaMelody.
* Disk Usage Plugin: This plugin records local disk usage.

## Security

* Escaped Markup Plugin: This plugin escapes the description of project, user, view, and build to prevent from XSS.
* Mask passwords Plugin: This plugin allows masking passwords that may appear in the console, including the ones defined as build parameters.
* Audit Trail Plugin: Keep a log of who performed particular Jenkins operations, such as configuring jobs.
* Matrix Authorization Strategy Plugin: Offers matrix-based security authorization strategies (global and per-project).
* Role Strategy Plugin: Adds a new role-based strategy to manage users' permissions.

## Pipeline / Flow

* Delivery Pipeline Plugin: Visualisation of Delivery/Build Pipelines, renders pipelines based on upstream/downstream jobs.
* Build Pipeline Plugin: This plugin provides a Build Pipeline View of upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution.
* Join Plugin: This plugin allows a job to be run after all the immediate downstream jobs have completed. The plugin is useful for creating a 'diamond' shape project dependency.
* Multijob Plugin: This plugin gives the option to define complex and hierarchical jobs structure in Jenkins.
* Build Flow Plugin: This plugin allows managing Jenkins jobs orchestration using a dedicated domain-specific language,, extracting the flow logic from jobs.
* Workflow Plugin: Originally inspired by the ﻿Build Flow Plugin.

NOTES: [Join Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Join+Plugin) + [Multijob Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Multijob+Plugin) == [Build Flow Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Build+Flow+Plugin) + [Groovy DSL](http://groovy-lang.org/documentation.html) == [Workflow Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Workflow+Plugin) + Snippet Generator or [Groovy DSL](http://groovy-lang.org/documentation.html)

## Build

* Copy Artifact Plugin: Adds a build step to copy artifacts from another project.
* Parameterized Trigger Plugin: This plugin lets you trigger new builds when your build has completed, with various ways of specifying parameters for the new build.
* Fingerprint Plugin: Adds the ability to generate fingerprints as build steps instead of waiting for a build to complete.
* Matrix Reloaded Plugin: The Matrix Reloaded Plugin allows rebuilding parts of an already built Matrix build.
* Promoted Builds Plugin: This plugin allows you to distinguish good builds from bad builds by introducing the notion of 'promotion'.
* ScriptTrigger Plugin: Makes it possible to monitor an environment with a script.

## SCM

* Git Plugin: This plugin allows use of Git as a build SCM.
* GitHub Plugin: This plugin integrates Jenkins with Github projects.
* Multi-Branch Project Plugin: This plugin adds an additional project type that creates sub-projects for each branch using a shared configuration.
* SCM Sync configuration plugin: Syncs configuration files to a SCM repository and tracks changes done to them

## Configuration management

* Ansible Plugin: This plugin allows to execute Ansible tasks as a job build step.
* saltstack-plugin: This plugin sends a SaltStack API message as a build step

## Mobile

### Android

* Google Play Android Publisher Plugin: Enables Jenkins to upload Android apps (APK files) and related info to Google Play.
* Android Lint Plugin: Parses output from the Android lint tool and displays the results for analysis.
* Android Emulator Plugin: Automates many Android development tasks including SDK installation, build file generation, emulator creation and launch, APK (un)installation, monkey testing and analysis...
* Android Device connector plugin: This plugin lists up all the Android devices connected to the master and all the Jenkins slaves, and provide operations to them.

### iOS

* iOS Device Connector Plugin: This plugin lists up all the iOS devices connected to the master and all the Jenkins slaves, and provide operations to them.
* Xcode Plugin: This plugin adds the ability to call Xcode command line tools to automate build and packaging iOS applications (iPhone, iPad, ...).

## Code analysis

* Cobertura Plugin: This plugin allows you to capture code coverage report from Cobertura. Jenkins will generate the trend report of coverage.
* SLOCCount Plugin: This plug-in generates trend report for SLOCCount and cloc open source tools, that count number of code lines written in many programming languages.
* Unicorn Validation Plugin: This plugin uses W3C's Unified Validator, which helps improve the quality of Web pages by performing a variety of checks.
* Violations: This plug-in generates reports static code violation detectors such as checkstyle, pmd, cpd, findbugs, codenarc, fxcop, stylecop and simian.
* Violation Columns Plugin: This plugin allows you to add columns to your Views with data from the Violations Plugin.

## Help

* Green Balls: Changes Jenkins to use green balls instead of blue for successful builds.
* Modern Status: Alternative set of status and action icons to provide a fresh look to Jenkins and to be friendly for all users i.e. iconic not just color indication.
* HTML Publisher: This plugin publishes HTML reports.
* HTML5 Notifier Plugin: Provides W3C Web Notifications support for builds.

## Testing

* FitNesse Plugin: This plugin can be used to both execute and report on FitNesse tests so that they can be integrated into a Jenkins build.
* seleniumhtmlreport Plugin: This plugin visualizes the results of selenium tests.
* Sauce OnDemand Plugin: This plugin allows you to integrate Sauce Selenium Testing with Jenkins. Specifically, you can: Automate the setup and tear down of Sauce Connect + Integrate the Sauce Labs result videos per test.
* Cucumber Test Result Plugin: This plugin allows you to show the results of Cucumber tests within Jenkins.
* Cucumber Reports Plugin: Publishes pretty html reports showing the results of cucumber-jvm runs.

## Notifications

* Email-ext plugin: This plugin allows you to configure every aspect of email notifications. You can customize when an email is sent, who should receive it, and what the email says.
* Google Calendar Plugin: This plugin publishes build records over to Google Calendar.
* Instant Messaging Plugin: This plugin provides generic support for build notifications and a 'bot' via instant messaging protocols. This plugin itself is of no use for end users. Please use one of the derived plugins like the Jabber or the IRC plugin!
* Twitter Plugin: This plugin posts build results to Twitter.
* Jenkins Sounds plugin: This plugin allows Jenkins to play arbitrary audio clips as build actions and notifications.

## Python

* Python Wrapper Plugin: This plugin provides the runtime library for plugins written in Python.
* Python Plugin: Adds the ability to execute python scripts as build steps.
* pyenv plugin: This plugin runs your jobs in the pyenv
* ShiningPanda Plugin: This plugin adds Python support to Jenkins with some useful builders (Python builder, virtualenv builder, tox builder...) and the ability to use a Python axis in multi-configuration projects (for testing on multiple versions of Python).

## VPS

* Docker Plugin: This plugin allows slaves to be dynamically provisioned using Docker.

## Cloud

* Amazon EC2 Plugin: Allow Jenkins to start slaves on EC2 or Eucalyptus on demand, and kill them as they get unused.
* JClouds Plugin: This plugin uses JClouds to provide slave launching on most of the currently usable Cloud infrastructures.
* Openstack Cloud Plugin: Simple and lightweight openstack-only fork of the JClouds-Plugin

## Others

* EnvInject Plugin: This plugin makes it possible to have an isolated environment for your jobs.
* Multi slave config plugin: This plugin allows administrators to configure, add and delete several dumb slaves at the same time.
* Performance Plugin: This plugin allows you to capture reports from JMeter and JUnit . Jenkins will generate graphic charts with the trend report of performance and robustness. It includes the feature of setting the final build status as good, unstable or failed, based on the reported error percentage.
* Workspace Cleanup Plugin: This plugin deletes the workspace before the build or when a build is finished and artifacts saved.
* CloudBees Folders Plugin: This plugin allows users to create "folders" to organize jobs.

=====================================================================

https://wiki.jenkins.io/display/JENKINS/View+Job+Filters

https://wiki.jenkins.io/display/JENKINS/Dashboard+View

https://wiki.jenkins.io/display/JENKINS/Build+Monitor+Plugin

https://wiki.jenkins.io/display/JENKINS/Status+Monitor+Plugin

https://wiki.jenkins.io/display/JENKINS/Categorized+Jobs+View

https://wiki.jenkins.io/display/JENKINS/Promoted+Builds+Plugin

https://wiki.jenkins.io/display/JENKINS/Build+Pipeline+Plugin

http://www.tothenew.com/blog/continuous-delivery-pipeline-jenkins-delivery-pipeline-view-plugin/

https://www.ibm.com/developerworks/library/d-continuous-delivery-framework-jenkins/index.html

https://www.credera.com/blog/technology-insights/open-source-technology-insights/how-to-create-an-easy-build-once-deploy-many-process-with-jenkins/

=======================================================================