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.

**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.

**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.

Exclusion’s 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.

**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.

**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.

**Job Configuration History Plugin**

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.

**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.

**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.

**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.

**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.

**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.

**SERVER MONITORING**

Plugins: Disk Usage, Monitoring

Keeping a watchful eye on your server lets you avoid some sticky situations. Why not employ some plugins to provide basic monitoring of your master and slaves? Use the Disk Usage plugin to keep your disk space in check and the Monitoring plugin to oversee a whole slew of interesting metrics in its very own monitoring page.

**NESTED VIEWS**

Plugin: Nested View

Organizing your jobs into separate views helps keep your Jenkins organized. Larger Jenkins instances, however, are quickly overwhelmed with dozens of views. Sadly, vanilla Jenkins doesn’t allow for grouping of views, but that’s where the Nested View plugin comes in. It allows you to group relevant views together under a parent view, creating comfortably traversed view trees.

**RESTARTING SAFELY**

Plugin: SafeRestart

Jenkins needs to restart every now and then, oftentimes when installing or updating plugins. This plugin allows you to put new jobs on hold and reboot the server once your running jobs are finished. The reboot can be scheduled from both the Jenkins configuration page and the main dashboard.

**GREEN BALLS**

Plugin: Green Balls

Jenkins uses blue balls to mark successful builds for a reason. However, enough people prefer green balls for this plugin, which changes the ball color, to be one of the most popular plugins out there.

**BUILD PIPELINES**

Plugin: Build Pipeline

The Build Pipeline plugin visualizes your up- and downstream connected jobs in a prettified view and helps you get a nice overview of your build process, start to finish. It also comes with the handy ability to add manual triggers to your jobs. This allows you to fit external processes, such as a review before deployment, into your pipeline.

**BUILD BADGES AND PROMOTIONS**

Plugins: Promoted Builds, Build Trigger Badge

Promote your builds as they go and decorate them with informative badges. The Build Trigger Badge plugin decorates your builds with a small badge depicting the build cause. Was it triggered manually, by an SCM change, or did it come from upstream?

Use the Promoted Builds plugin to take it one step further, promoting and decorating your builds throughout your entire pipeline. With every passing step, you can decorate your build with a badge depicting its success. Just a glance at a build will tell you what originally started it and how well it performed further down the pipeline. For the creative users out there, itâ€™s possible to trigger jobs when certain promotion levels are met.

**BUILDING A BETTER DASHBOARD**

Plugins: Build Monitor, Dashboard View, Radiator View

Piece together your dream dashboard and display the status of your key jobs on that big monitor by the coffee machine. Sip your coffee with pride as you bask in its green glow (or scurry back to your desk to fix that last commit instead). Gather and display interesting statistics about your builds on a different dashboard, letting you to identify fragile parts of your build process and pinpoint its ills.

**JOBS AS CODE**

**Plugin: Job DSL**

Storing your job and pipeline configuration as code comes with a myriad of advantages. It allows you to version control your configuration, regenerate your jobs on the fly, spin up and configure new Jenkins instances in a heartbeat, etc. The Jenkins Job DSL is a popular and exceptionally well documented Groovy DSL for configuring pipelines and jobs via scripts. Sinking your teeth into this one will doubtlessly open up some new doors.

**JOB SCRIPTING AND PARAMETERIZATION**

**Plugins: EnvInject, Environment Script**

Reduce redundancy and increase reusability of your jobs by parameterizing their variable values. Use the EnvInject and Environment Script plugin to help you inject necessary values into your jobs. Theyâ€™re surprisingly flexible and can really help you construct generic jobs. As an added bonus, once your jobs are parameterized, it will be trivial to convert them to clean scripts using Job DSL.

**COMPILER WARNINGS**

**Plugin: Warnings**

Keep track of compiler warnings thrown in your builds, they’re a great indicator of potential problems in the code. Don’t panic if your first build returns hundreds of issues, trends are more important than absolute values. You can steadily bring those numbers down with diligence and you’ll rest easy knowing the code gets a little better with every commit.

**TEST RESULTS AND COVERAGE**

**Plugins: xunit, junit, nunit, Test Results Analyzer**

Round up the results of your tests and plot them in helpful graphs and statistics. (You do have tests, don’t you?) Identify fragile tests and invest some time to fortify them (or the code they’re testing). Keep an eye on your test coverage trend, challenge yourself to increase that percentage with every commit.

**TASK SCANNING**

**Plugin: Task Scanner**

Remember that //TODO you left last friday? Or that @deprecated method you still need to replace throughout the code? We know you’ll go back to fix them eventually, but don’t let them pile up too high, use the Task Scanner plugin to keep them in check. It comes with a bunch of handy features like trend reports, prioritization and thresholds.