|  |  |
| --- | --- |
| **#** | **Plugin description** |
| 1 | 01   **Plugin ID:** **build-with-parameters**  02   **Wiki URL** (date of last commit to master branch): https://wiki.jenkins.io/display/JENKINS/Build+With+Parameters+Plugin  **Plugins URL:** https://plugins.jenkins.io/build-with-parameters  03   **Latest Release:** 1.4  04   **List of dependencies (if any):**  [Matrix Project v.1.0 (implied)](https://plugins.jenkins.io/matrix-project) [(what's this?)](https://plugins.jenkins.io/build-with-parameters)  [JUnit v.1.0 (implied)](https://plugins.jenkins.io/junit) [(what's this?)](https://plugins.jenkins.io/build-with-parameters)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/build-with-parameters)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/build-with-parameters)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/build-with-parameters)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/build-with-parameters-plugin  06   **Plugins’ Description:**  Allows the user to provide parameters for a build in the url, prompting for confirmation before triggering the job.  07   **Usage (describe or provide snapshot):** The plugin exposes $JENKINS/job/$JOB/**parambuild** url to trigger the build with parameter. Unlike $JENKINS/job/$JOB/buildWithParameters, human must confirm that the parameters are correct before the build is triggered. This is useful if you want to create a list of jobs with parameters to trigger ahead of time, and execute it at some future date (e.g. a deployment plan).  An example triggering of a job with the plugin: |
| 2 | 01   **Plugin ID:** **parameter-separator**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Parameter+Separator+Plugin  **Plugins URL:** https://plugins.jenkins.io/parameter-separator  03   **Latest Release:** 1.0  04   **List of dependencies (if any):**  [Matrix Authorization Strategy v.1.0.2(implied)](https://plugins.jenkins.io/matrix-auth) [(what's this?)](https://plugins.jenkins.io/parameter-separator)  [Windows Slaves v.1.0 (implied)](https://plugins.jenkins.io/windows-slaves) [(what's this?)](https://plugins.jenkins.io/parameter-separator)  [OWASP Markup Formatter v.1.0 (implied)](https://plugins.jenkins.io/antisamy-markup-formatter)[(what's this?)](https://plugins.jenkins.io/parameter-separator)  [Matrix Project v.1.0 (implied)](https://plugins.jenkins.io/matrix-project) [(what's this?)](https://plugins.jenkins.io/parameter-separator)  [JUnit v.1.0 (implied)](https://plugins.jenkins.io/junit) [(what's this?)](https://plugins.jenkins.io/parameter-separator)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/parameter-separator)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/parameter-separator)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/parameter-separator)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/parameter-separator-plugin  06   **Plugins’ Description:** This is a dead simple parameter plugin for Jenkins CI that allows one to clearly differentiate sets of parameters on a jenkins build page. This can be useful for builds with lots of parameters (builds that dynamically build different items pased on the job parameters set at build time, for instance). Specifically, the plugin creates an <hr /> tag with a globally configurable style attribute to better suit one's Jenkins install.  07   **Usage (describe or provide snapshot):**   1. Install this plugin from the Jenkins Update Center 2. Configure the styling for your parameters in the Jenkins global configuration page (Manage Jenkins -> Configure System; Parameter Separator section)  https://wiki.jenkins-ci.org/download/attachments/72777819/parameter_separator_globalconfigstyle1.png?version=1&modificationDate=1402422589000&api=v2 3. Add any number of parameter separator elements to your parameterized job on the job's configuration page  https://wiki.jenkins-ci.org/download/attachments/72777819/parameter_separator_jobconfig1.png?version=1&modificationDate=1402422606000&api=v2 4. Now when you start a build of your job, you will have parameters clearly separated by your stylized separator!   https://wiki.jenkins-ci.org/download/attachments/72777819/parameter_separator_buildwithparams1.png?version=1&modificationDate=1402422644000&api=v2 |
| 3 | 01   **Plugin ID:** **greenballs**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Green+Balls  **Plugins URL:** https://plugins.jenkins.io/greenballs  03   **Latest Release:** 1.15  04   **List of dependencies (if any):**  [External Monitor Job Type v.1.0 (implied)](https://plugins.jenkins.io/external-monitor-job)[(what's this?)](https://plugins.jenkins.io/greenballs)  [LDAP v.1.0 (implied)](https://plugins.jenkins.io/ldap) [(what's this?)](https://plugins.jenkins.io/greenballs)  [PAM Authentication v.1.0 (implied)](https://plugins.jenkins.io/pam-auth) [(what's this?)](https://plugins.jenkins.io/greenballs)  [Mailer v.1.2 (implied)](https://plugins.jenkins.io/mailer) [(what's this?)](https://plugins.jenkins.io/greenballs)  [Matrix Authorization Strategy v.1.0.2(implied)](https://plugins.jenkins.io/matrix-auth) [(what's this?)](https://plugins.jenkins.io/greenballs)  [Windows Slaves v.1.0 (implied)](https://plugins.jenkins.io/windows-slaves) [(what's this?)](https://plugins.jenkins.io/greenballs)  [OWASP Markup Formatter v.1.0 (implied)](https://plugins.jenkins.io/antisamy-markup-formatter)[(what's this?)](https://plugins.jenkins.io/greenballs)  [Matrix Project v.1.0 (implied)](https://plugins.jenkins.io/matrix-project) [(what's this?)](https://plugins.jenkins.io/greenballs)  [JUnit v.1.0 (implied)](https://plugins.jenkins.io/junit) [(what's this?)](https://plugins.jenkins.io/greenballs)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/greenballs)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/greenballs)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/greenballs)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/greenballs-plugin  06   **Plugins’ Description:** Changes Hudson to use green balls instead of blue for successful builds.  07   **Usage (describe or provide snapshot):** |
| 4 | 01   **Plugin ID:** **role-strategy**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Role+Strategy+Plugin  **Plugins URL:** https://plugins.jenkins.io/role-strategy  03   **Latest Release:** 2.9.0  04   **List of dependencies (if any):**  [Matrix Authorization Strategy v.1.7(required)](https://plugins.jenkins.io/matrix-auth)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/role-strategy)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/role-strategy)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/role-strategy)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/role-strategy-plugin  06   **Plugins’ Description:** Adds a new role-based strategy to manage users' permissions.  This plugin adds a new role-based strategy to ease and fasten users management. This strategy allows:   * Creating **global roles**, such as admin, job creator, anonymous, etc., allowing to set Overall, Slave, Job, Run, View and SCM permissions on a global basis. * Creating **project roles**, allowing to set only Job and Run permissions on a project basis. * Creating **slave roles**, allowing to set node-related permissions. * Assigning these roles to users.   07   **Usage (describe or provide snapshot):**  Using the plugin is fairly simple:   1. Activate the **Role-Based Strategy** by using the standard **Manage Jenkins > Configure System** screen: https://wiki.jenkins.io/download/attachments/46335825/role-strategy-01.png?version=1&modificationDate=1284951431000&api=v2 2. Define and assign roles by using the **Manages Roles** item which appears in the **Manage Jenkins** screen: https://wiki.jenkins.io/download/attachments/46335825/role-strategy-02.png?version=1&modificationDate=1284951431000&api=v2 You then get following options: https://wiki.jenkins.io/download/attachments/46335825/Screen%20Shot%202013-08-29%20at%2017.18.24.png?version=1&modificationDate=1377785944000&api=v2#\* **Manage Roles** is the place where to set up roles: https://wiki.jenkins.io/download/attachments/46335825/role-strategy-04.png?version=1&modificationDate=1284951431000&api=v2 There's nothing much to say here, this is self-explanatory. The only tricky field is the **Pattern** one. This field consists in a regular expression aimed at matching the full name (including the folder name, if you're using Cloudbees Folders Plugin) of the jobs which the role will apply to. For example, if you set the field to "Roger-.\*", then the role will match all jobs which name starts with "Roger-". Note that the pattern is case-sensitive. To perform a case-insensitive match, use (?i) notation: upper, "Roger-.\*" vs. lower, "roger-.\*" vs. case-insensitive, "(?i)roger-.\*". If you have a nested folder structure where you want to provide the particular access to the second folder (or deeper), consider having a two-level security structure as well (Say you want to provide exclusive write/ modify type access to foo/bar and not everything else under "foo": First, assign that user/ group to read/ discover permissions with pattern " ^foo.\*"**,** then assign that same user/ group to the more particular permissions with pattern " ^foo/bar.\* " - Similar to what you'd do in a Unix/ Linux environment.    1. #\* **Assign Roles** is the place where to assign the defined roles to users: https://wiki.jenkins.io/download/attachments/46335825/role-strategy-05.png?version=1&modificationDate=1284951431000&api=v2 |
| 5 | 01   **Plugin ID:** **schedule-build**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Schedule+Build+Plugin  **Plugins URL:** https://plugins.jenkins.io/schedule-build  03   **Latest Release:** 0.5.1  04   **List of dependencies (if any):**  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/schedule-build)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/schedule-build)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/schedule-build-plugin  06   **Plugins’ Description:** Adds capability to schedule a build for a later point in time. Asks the user for a date and time and adds the build to the build queue with the respective quiet period.  07   **Usage (describe or provide snapshot):** Scheduling Builds Press the "Schedule Build" link on the project page or use the schedule build action in the list view.  https://wiki.jenkins-ci.org/download/attachments/71435843/Schedule%20Project%20Page.png?version=1&modificationDate=1415391774000&api=v2  https://wiki.jenkins-ci.org/download/attachments/71435843/Schedule%20Action.png?version=1&modificationDate=1415391770000&api=v2 Then select date and time when to schedule the build.  https://wiki.jenkins-ci.org/download/attachments/71435843/Schedule%20Page.png?version=1&modificationDate=1415392107000&api=v2 The build will be added to the build queue with the respective quiet period.  https://wiki.jenkins-ci.org/download/attachments/71435843/Scheule%20Build%20Queue.png?version=1&modificationDate=1415392391000&api=v2 Configure Schedule Build Plugin The configuration of the schedule build plugin is very simple. There are only two parameters on the Jenkins system configuration page.  The default time which is set when a user wants to schedule a build may be configured and time zone used by the plugin, which might differ from the system time zone.  https://wiki.jenkins-ci.org/download/attachments/71435843/image2017-7-10%2023%3A31%3A3.png?version=1&modificationDate=1499722264000&api=v2 |
| 6 | 01   **Plugin ID:** **mask-passwords**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Mask+Passwords+Plugin  **Plugins URL:** https://plugins.jenkins.io/mask-passwords  03   **Latest Release:** 2.12.0  04   **List of dependencies (if any):**  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/mask-passwords)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/mask-passwords)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/mask-passwords)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/mask-passwords-plugin  06   **Plugins’ Description:** This plugin allows masking passwords that may appear in the console  07   **Usage (describe or provide snapshot):**  This plugin allows masking passwords that may appear in the console, including the ones defined as build parameters. This often happens, for example, when you use build steps which can't handle passwords properly. Take a look at the following example. Before Consider you're using an **Invoke Ant** build step to run an Ant target. This target requires a password to achieve its goal. You would end up having a job configuration like this:  https://wiki.jenkins.io/download/attachments/46335226/config-before.png?version=1&modificationDate=1283327974000&api=v2  Of course, you could have created a variable to store the password and use this variable in the build step configuration so that it doesn't appear as plain text. But you would still end with a console output like this:  https://wiki.jenkins.io/download/attachments/46335226/console-before.png?version=1&modificationDate=1283327974000&api=v2 After When activating the **Mask passwords** option in a job, the builds' **Password Parameters** (or any other type of build parameters selected for masking in **Manage Hudson** > **Configure System**) are automatically masked from the console. Furthermore, you can also safely define a list of static passwords to be masked (you can also define a list of static password shared by all jobs in Jenkins' main configuration screen). As such, the passwords don't appear anymore as plain text in the job configuration (plus it is ciphered in the job configuration file):  https://wiki.jenkins.io/download/attachments/46335226/config-after.png?version=1&modificationDate=1283327974000&api=v2  Once done, new builds will have the passwords masked from the console output:  https://wiki.jenkins.io/download/attachments/46335226/console-after.png?version=1&modificationDate=1283327974000&api=v2 User guide First, go to Jenkins' main configuration screen (**Manage Hudson** > **Configure System**) and select, in the **Mask Passwords - Configuration**section, which kind of build parameters have to be automatically masked from the console output:  https://wiki.jenkins.io/download/attachments/46335226/global-settings.png?version=1&modificationDate=1299862924000&api=v2  Notice that, as of version 2.7, you can also define global passwords (defined as pairs of name/password) that can be accessed across all jobs.  Then, for a specific job, activate the **Mask passwords** option in the **Build Environment** section to mask passwords from the console:   1. All the password parameters defined for the job will be automatically hidden. 2. For each other kind of password (that is, static ones) that may appear in the console output, add an entry (by clicking on the **Add**button) and set the **Password** field. You may additionally set the **Name** field. If you do so, the password will then be available as a standard variable. It is then possible to refer to this password using this variable rather than keying it in a field which is not ciphered. Take a look at the screenshots above for an example. |
| 7 | 01   **Plugin ID:** **build-pipeline-plugin**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Build+Pipeline+Plugin  **Plugins URL:** https://plugins.jenkins.io/build-pipeline-plugin  03   **Latest Release:** 1.5.8  04   **List of dependencies (if any):**  [Parameterized Trigger v.2.17 (required)](https://plugins.jenkins.io/parameterized-trigger)  [jQuery v.1.7.2-1 (required)](https://plugins.jenkins.io/jquery)  [Dashboard View v.2.2 (optional)](https://plugins.jenkins.io/dashboard-view)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/build-pipeline-plugin)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/build-pipeline-plugin)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/build-pipeline-plugin)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/build-pipeline-plugin  06   **Plugins’ Description:** 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, e.g. an approval process outside of Jenkins.  07   **Usage (describe or provide snapshot):**  Continuous Integration has become a widely adopted practice in modern software development. Jenkins & Hudson are great tools for supporting Continuous Integration.  **Taking it to the next level:** Continuous integration can become the centerpiece of your [deployment pipeline](http://www.informit.com/articles/article.aspx?p=1621865), orchestrating the promotion of a version of software through quality gates and into production. By extending the concepts of CI you can create a chain of jobs each one subjecting your build to quality assurance steps. These QA steps may be a combination of manual and automated steps. Once a build has passed all these, it can be automatically deployed into production.  In order to better support this process, we have developed the Build Pipeline Plugin. This gives the ability to form a chain of jobs based on their upstream\downstream dependencies. Downstream jobs may, as per the default behaviours, be triggered automatically ,or by a suitable authorised user manually triggering it.  You can also see a history of pipelines in a view, the current status and where each version got to in the chain based on its revision number in VCS. ScreenshotsThe Pipeline View https://wiki.jenkins.io/download/attachments/54723106/bpp1.png?version=2&modificationDate=1340695983000&api=v2 ConfigurationView Configuration  1. Install the plugin using the Hudson\Jenkins Plugin Manager and restart. 2. Create a view of the new type Build Pipeline View. You will then be redirected directly to the configuration page. 3. The table below outlines what each interesting parameter controls:  | **Name** | The name of the Build Pipeline View | | --- | --- | | **Description** | This message will be displayed on the view page. Useful for describing what this view is about, or linking to relevant resources. Can contain HTML tags. | | **Build Pipeline View Title** | Gives a title to the page that displays the view | | **Select Initial Job** | This is the first job in the build pipeline. It will traverse through the downstream jobs to build up the entire build pipeline.  Select from a drop-down list of jobs. | | **No of Displayed Builds** | The number of historical builds to be displayed on a page. | | **Restrict triggers to most recent successful builds** | Select this option to restrict the display of a Trigger button to only the most recent successful build pipelines.  Yes: Only the most recent successful builds displayed on the view will have a manual trigger button for the next build in the pipeline.  No: All successful builds displayed on the view will have a manual trigger button for the next build in the pipeline. | | **Always allow manual trigger on pipeline steps** | Select this option if you want to manually execute or re-execute any step of the pipeline at any time. | | **Show pipeline parameters** | Select this option if you want to display the parameters used to run the first job in the pipeline. |   https://wiki.jenkins.io/download/attachments/54723106/config.png?version=1&modificationDate=1340758239000&api=v2 Job Configuration  * 1. Navigate to the Job configuration page.   2. Scroll down to the Post-build Actions section.      1. For an **Automated** downstream build step; To add a build step that will trigger automatically upon the successful completion of the previous one:         1. Select the Build other projects check-box         2. Enter the name(s) of the downstream projects in the Projects to build field. (n.b. Multiple projects can be specified by using comma, like "abc, def".)         3. For a **Manually Triggered** downstream build step: To add a build step that will wait for a manual trigger:            1. Select the Build Pipeline Plugin -> Manually Execute Downstream Project check-box            2. Enter the name(s) of the downstream projects in the Downstream Project Names field. (n.b. Multiple projects can be specified by using comma, like "abc, def".)            3. Click Save   **Automatic & Manual downstream build steps**  The Build Pipeline Plugin handles the creation of multiple automatic and/or manually triggered downstream build steps on the same project.  https://wiki.jenkins.io/download/attachments/54723106/JobConfig.PNG?version=2&modificationDate=1346302165000&api=v2 |
| 8 | 01   **Plugin ID:** **timestamper**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Timestamper  **Plugins URL:** https://plugins.jenkins.io/timestamper  03   **Latest Release:** 1.8.10  04   **List of dependencies (if any):**  [Pipeline: Step API v.1.7 (optional)](https://plugins.jenkins.io/workflow-step-api)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/timestamper)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/timestamper)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/timestamper)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/timestamper-plugin  06   **Plugins’ Description:** Adds timestamps to the Console Output.  Example output:  21:51:15 Started by user anonymous  21:51:15 Building on master  21:51:17 Finished: SUCCESS  07   **Usage (describe or provide snapshot):**  Enable timestamps within the "Build Environment" section of the build's configuration page.  To enable timestamps for multiple builds at once, use the [Configuration Slicing Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Configuration+Slicing+Plugin) version 1.32 or later. **Instructions:**[**Pipeline Builds**](https://wiki.jenkins-ci.org/display/JENKINS/Pipeline+Plugin) **Since Timestamper 1.8**  Use the timestamps step to wrap the rest of the pipeline script.  timestamps {  // some block  }  **Since Timestamper 1.7**  Prior to Timestamper 1.8, timestamps can only be recorded within a node.  node {  wrap([$class: 'TimestamperBuildWrapper']) {  echo 'hello from Workflow'  }  } **Customization**  * The timestamp format can be configured via the Configure System page. * There is a panel on the left-hand side of the console page which allows either the system clock time or the elapsed time to be displayed. * The time zone used to display the timestamps can be configured by setting a system parameter as described here: [Change time zone](https://wiki.jenkins-ci.org/display/JENKINS/Change+time+zone).  **Scripting** Scripts can read the timestamps from the /timestamps/ URL of each build.  Examples:   * + /timestamps/ By default, display the elapsed time in seconds with three places after the decimal point.   + /timestamps/?time=HH:mm:ss&appendLog Display the system clock time and append the line from the log.   + /timestamps/?elapsed=HH:mm:ss.S&appendLog Display the elapsed time and append the line from the log.   + /timestamps/?time=HH:mm:ss&elapsed=HH:mm:ss.S Display both the system clock time and the elapsed time.   + /timestamps/?currentTime&time=HH:mm:ss Display the current time on the Jenkins master.   Supported query parameters:   * + - **time** (since 1.8) Display the system clock time. Accepts the [JDK SimpleDateFormat](http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html) format. The time zone and locale of the Jenkins server will be used, unless they are configured with the timeZone and locale query parameters.     - **elapsed** (since 1.8) Display the elapsed time since the start of the build. Accepts the [commons-lang DurationFormatUtils](https://commons.apache.org/proper/commons-lang/javadocs/api-2.6/org/apache/commons/lang/time/DurationFormatUtils.html) format.     - **precision** (since 1.3.2) Display the elapsed time in seconds, with a certain number of places after the decimal point. Accepts a number of decimal places or values such as seconds and milliseconds.     - **appendLog** (since 1.8) Display the console log line after the timestamp.     - **startLine** (since 1.8) Display the timestamps starting from a certain line. Accepts a positive integer to start at that line, or a negative integer to start that many lines back from the end.     - **endLine** (since 1.8) Display the timestamps ending at a certain line. Accepts a positive integer to finish at that line, or a negative integer to finish that many lines back from the end.     - **timeZone** (since 1.8) Time zone used when displaying the system clock time. Accepts the [JDK TimeZone](http://docs.oracle.com/javase/6/docs/api/java/util/TimeZone.html) ID format.     - **locale** (since 1.8) Select the locale to use when displaying the system clock time. Accepts a locale in the format recognised by [commons-lang LocaleUtils.toLocale](https://commons.apache.org/proper/commons-lang/javadocs/api-2.6/org/apache/commons/lang/LocaleUtils.html#toLocale(java.lang.String)).     - **currentTime** (since 1.8.8) Display the current time on the Jenkins master instead of reading timestamps from the build.   Reading the timestamps directly from the file system is not recommended, because the format may change. **Java API** **Since Timestamper 1.8**  Other plugins can add a [dependency](https://wiki.jenkins-ci.org/display/JENKINS/Dependencies+among+plugins) on the Timestamper plugin, and then use the TimestamperAPI.read method to retrieve the timestamps. The read method accepts any query string that can be passed to the /timestamps/ URL.  String query = "time=HH:mm:ss";  try (BufferedReader reader = TimestamperAPI.get().read(build, query)) {  // read timestamps here  } |
| 9 | 01   **Plugin ID:** **configurationslicing**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Configuration+Slicing+Plugin  **Plugins URL:** https://plugins.jenkins.io/configurationslicing  03   **Latest Release:** 1.47  04   **List of dependencies (if any):**  [Ant v.1.1 (optional)](https://plugins.jenkins.io/ant)  [Email Extension v.2.37 (optional)](https://plugins.jenkins.io/email-ext)  [Logstash v.1.2.0 (optional)](https://plugins.jenkins.io/logstash)  [build log file size checker v.1.2 (optional)](https://plugins.jenkins.io/logfilesizechecker)  [Maven Integration v.2.14 (optional)](https://plugins.jenkins.io/maven-plugin)  [Mask Passwords v.2.8 (optional)](https://plugins.jenkins.io/mask-passwords)  [Timestamper v.1.2.2 (optional)](https://plugins.jenkins.io/timestamper)  [Jython v.1.9 (optional)](https://plugins.jenkins.io/jython)  [Matrix Project v.1.6 (optional)](https://plugins.jenkins.io/matrix-project)  [Build Timeout v.1.16 (optional)](https://plugins.jenkins.io/build-timeout)  [Gradle v.1.24 (optional)](https://plugins.jenkins.io/gradle)  [Python v.1.2 (optional)](https://plugins.jenkins.io/python)  [Priority Sorter v.1.3 (optional)](https://plugins.jenkins.io/PrioritySorter)  [Config File Provider v.2.7.1 (optional)](https://plugins.jenkins.io/config-file-provider)  [Groovy v.1.9 (optional)](https://plugins.jenkins.io/groovy)  [Claim v.2.3 (optional)](https://plugins.jenkins.io/claim)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/configurationslicing)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/configurationslicing)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/configurationslicing)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/configurationslicing-plugin  06   **Plugins’ Description:** Perform mass configuration of select project properties, including email, timer, discard old builds, and Maven configuration.It has a framework to make it very easy to add a configuration page for a new property.  At present, two types of data can be mass-configured: booleans and strings.  07   **Usage (describe or provide snapshot):**  The plugin shows up in Jenkins' UI on the Manage Jenkins page - seen here near the bottom of the page:   https://wiki.jenkins.io/download/attachments/38142123/ConfigurationSlicing%20-%20Manage%20Hudson.png?version=2&modificationDate=1310821741000&api=v2  The main page of the configuration slicing plugin shows all the properties that can be sliced - select one and you are presented with a screen showing how that value is set across the entire Jenkins instance. Many properties on Jenkins projects are useful to set this way, but the configuration slicing plugin can handle properties on any collection, such as slaves, or builds of a project.  **The following functions are supported**   * [Ant version per project](https://wiki.jenkins.io/display/JENKINS/Ant+Plugin) * Block Build when Downstream Building Slicer (bool) * Block Build when Upstream Building Slicer (bool) * [Build Timeout](https://wiki.jenkins.io/display/JENKINS/Build-timeout+Plugin) (does not support all features) * Custom Workspace Slicer (Advanced Project Options > Use custom workspace) * Discard Old Builds Slicer - Days to keep artifacts * Discard Old Builds Slicer - Days to keep builds * Discard Old Builds Slicer - Max # of builds to keep * Discard Old Builds Slicer - Max # of builds to keep with artifacts * E-mail Notification * [Editable Email Notification](https://wiki.jenkins.io/display/JENKINS/Email-ext+plugin) ([recipient list only](https://issues.jenkins-ci.org/browse/JENKINS-11774)) * [Execute Jython script](https://wiki.jenkins.io/display/JENKINS/Jython+Plugin) * [Execute Python script](https://wiki.jenkins.io/display/JENKINS/Python+Plugin) * Execute shell slicer * Execute Windows batch command slicer * [Gradle version per project](https://wiki.jenkins.io/display/JENKINS/Gradle+Plugin) * [Groovy version per project](https://wiki.jenkins.io/display/JENKINS/Groovy+plugin) * JDK per project * Job Disabled Build Slicer (bool) * Job Disabled Build Slicer (String) * [Job Priority Slicer](https://wiki.jenkins.io/display/JENKINS/Priority+Sorter+Plugin) * [Logfilesizechecker Plugin](https://wiki.jenkins.io/display/JENKINS/Logfilesizechecker+Plugin) * Maven "top-level" targets * Maven Goals and Options (Maven project) * Maven Version (Maven Projects) * MAVEN\_OPTS per Maven project * Parameters * Quiet period * SCM Timer Trigger Slicer * Tied Label Slicer * Timer Trigger Slicer * [Timestamper Slicer](https://wiki.jenkins.io/display/JENKINS/Timestamper) * [Claim Slicer](https://wiki.jenkins.io/display/JENKINS/Configuration+Slicing+Plugin)  Boolean slicing In the case of booleans, the plugin presents a set of checkboxes and names.  The user can then adjust that property and save the changes.  https://wiki.jenkins.io/download/attachments/38142123/ConfigurationSlicing%20-%20Job%20Disabled%20Slice.png?version=2&modificationDate=1310822184000&api=v2 String slicing Most of the slicing uses a GUI much like the following example.  You can move the Item Names (i.e. Jobs) around within the boxes on the right to change which jobs have different settings.  You can also alter the values on the left to change how jobs are configured.  There will always be a blank set of boxes added to the bottom to allow you to create a new setting when you need it.  For most of these screens, a value of "(Disabled)" will indicate that those jobs do not use this configuration at all.  To disable jobs (e.g. for SCM Polling), move those job names into that "(Disabled)" box.  https://wiki.jenkins.io/download/attachments/38142123/ConfigurationSlicing%20-%20SCM%20Timer%20Slice.png?version=2&modificationDate=1310822734000&api=v2 String slicing multiple values Some slicers allow you to configure multiple values at a time. In those cases, the values are separated by a comma, and follow the given example.  https://wiki.jenkins.io/download/attachments/38142123/ConfigurationSlicing%20-%20string-multiple-values.png?version=1&modificationDate=1322526234000&api=v2 String slicing multiple Builders For the Windows batch builder, Shell builders, and "Top-level Maven targets", a job can have multiple builders of each type. To configure jobs like this, you will be presented with an index next to the jobs names like "MyJob[0]" and "MyJob[1]". The index indicates which instance of the builder you are configuring.  https://wiki.jenkins.io/download/attachments/38142123/StringSlicingMultipleBuilders.png?version=1&modificationDate=1337433359000&api=v2  This is available under these links   * + Execute shell slicer   + Execute Windows batch command slicer   + Maven "top-level" targets  Configuring parameters across multiple jobs Job Parameters (aka "This build is parameterized") can be configured across multiple jobs at one time through the "Parameters" link. To indicate which parameter you are configuring, note the "JobName[ParameterName]" syntax.  https://wiki.jenkins.io/download/attachments/38142123/ParametersSlicingItems.png?version=1&modificationDate=1337733717000&api=v2 https://wiki.jenkins.io/download/attachments/38142123/ParametersSlicingStringParameter.png?version=1&modificationDate=1337733736000&api=v2 Slicing by View If you have many jobs, it can be difficult to perform the configuration slicing. To make it more granular, you can configure just the jobs within one view. Assuming you have organized your Jenkins installation to have useful views, this will allow you to configure jobs at the right granularity. To use this feature, first select the type of configuration (in this example "Custom Workspace") and then you will be given a list of views to choose from. You don't have to choose a view, as the default is to show all jobs. If you select one of the views on the left, your list of jobs is filtered down to just the jobs in that view.  https://wiki.jenkins.io/download/attachments/38142123/ConfigurationSlicing%20-%20Views.png?version=1&modificationDate=1318517242000&api=v2 Email Notifications and [Editable Email Notifications](https://wiki.jenkins.io/display/JENKINS/Email-ext+plugin) (from 1.41 on) When editing recipient lists, Email notifications are only completely disabled when set to (Disabled).  Setting a empty recipient list leaves existing email notifications to committers (Checkbox "Notify individuals who broke the build") in place. |
| 10 | 01   **Plugin ID:** **parameterized-trigger**  02   **Wiki URL (date of last commit to master branch):** https://wiki.jenkins.io/display/JENKINS/Parameterized+Trigger+Plugin  **Plugins URL:** https://plugins.jenkins.io/parameterized-trigger  03   **Latest Release:** 2.35.2  04   **List of dependencies (if any):**  [Conditional BuildStep v.1.3.1 (required)](https://plugins.jenkins.io/conditional-buildstep)  [Matrix Project v.1.6 (required)](https://plugins.jenkins.io/matrix-project)  [Script Security v.1.25 (required)](https://plugins.jenkins.io/script-security)  [promoted builds v.2.25 (optional)](https://plugins.jenkins.io/promoted-builds)  [Subversion v.2.5.7 (optional)](https://plugins.jenkins.io/subversion)  [bouncycastle API v.2.16.0 (implied)](https://plugins.jenkins.io/bouncycastle-api) [(what's this?)](https://plugins.jenkins.io/parameterized-trigger)  [Command Agent Launcher v.1.0 (implied)](https://plugins.jenkins.io/command-launcher)[(what's this?)](https://plugins.jenkins.io/parameterized-trigger)  [JDK Tool v.1.0 (implied)](https://plugins.jenkins.io/jdk-tool) [(what's this?)](https://plugins.jenkins.io/parameterized-trigger)  05   **Source Code URL (GitHub):** https://github.com/jenkinsci/parameterized-trigger-plugin  06   **Plugins’ Description:** This plugin lets you trigger new builds when your build has completed, with various ways of specifying parameters for the new build.  07   **Usage (describe or provide snapshot):**  Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:   * [Missing permission check allows building all jobs](https://jenkins.io/security/advisory/2017-07-10/)   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.  There is also a [Parameterized Remote Trigger Plugin](https://wiki.jenkins-ci.org/display/JENKINS/Parameterized+Remote+Trigger+Plugin) in case you want to trigger a build on a different/remote Jenkins Master.  https://wiki.jenkins-ci.org/download/attachments/36602920/screenshot.png?version=1&modificationDate=1249903259000&api=v2  The parameters section can contain a combination of one or more of the following:   * + a set of predefined properties   + properties from a properties file read from the workspace of the triggering build   + the parameters of the current build   + **Subversion revision**: makes sure the triggered projects are built with the same revision(s) of the triggering build. You still have to make sure those projects are actually configured to checkout the right Subversion URLs.   + **Restrict matrix execution to a subset**: allows you to specify the same combination filter expression as you use in the matrix project configuration and further restricts the subset of the downstream matrix builds to be run.   The parameter section is itself pluggable, and other plugins can contribute other sources of parameters.  This triggering mechanism can be used both as a post-build step or as a build step, in which case you can also block for the completion of the triggered builds. This lets you create a "function call" like semantics.  **\*\*\* YOU MUST DEFINE THE PARAMETER IN DOWNSTREAM JOBS VIA  "This project is parameterized". For example, if job1 passes ABC=123 to job2 then in job2 mark the job as "This project is parameterized" and "Add Parameter" named "ABC". \*\*\*** **Usage as a Build step** When using the "Trigger/Call builds on another project" item. If the trigger is configured with the "Block until the triggered projects finish their builds" enabled, the following Environment variables are made available for further build steps  Env variables for future build steps   * + - LAST\_TRIGGERED\_JOB\_NAME="Last project started"     - TRIGGERED\_BUILD\_NUMBER\_<project name>="Last build number triggered" **from version 2.17 onwards**     - TRIGGERED\_JOB\_NAMES="Comma separated list of all triggered projects"     - TRIGGERED\_BUILD\_NUMBERS\_<project name>="Comma separated list of build numbers triggered"     - TRIGGERED\_BUILD\_RESULT\_<project name>="Last triggered build result of project"     - TRIGGERED\_BUILD\_RESULT\_<project name>RUN<build number>="Result of triggered build for build number"     - TRIGGERED\_BUILD\_RUN\_COUNT\_project name>="Number of builds triggered for the project"   From 2.17 onwards  All Project names have characters not a-zA-Z or 0-9 replaced by \_(multiple characters are condensed into a single \_).  Note that with the BuildStep a variable can be used for the project name, I.E. ${projectName}.  Please submit bugs and feature requests to the issue tracker and not (only) in the comments. **Use of the plugin in a Matrix job**Post build task When using the trigger parameterized build as a post build task for a matrix job the triggering will be be done once when all of the different matrix configurations have completed. In this case some of the Environment variables may not be resolvable as passing them to downstream jobs will fail.  You also cannot use a variable for the downstream project name.  If this functionality is needed, the BuildStep must be used.  Environment variables that should be available are the the default shell ones (<yourserver:port>/env-vars.html) and ones defined as Parameters. Variables added by the other plugins as a buildwrappers may not be available. Build step When using the trigger parameterized build as a buildstep it will be called for every different configuration, so if triggering another project with no parameters it will be done the same number of times as you have configurations, possible causing the triggered job to run more than once.  However this also allows you to trigger other jobs with parameters relating to the current configuration, i.e. triggering a build on the same node with the same JDK. |
|  | 01   **Plugin ID:**  02   **Wiki URL (date of last commit to master branch):**  **Plugins URL:**  03   **Latest Release:**  04   **List of dependencies (if any):**  05   **Source Code URL (GitHub):**  06   **Plugins’ Description:**  07   **Usage (describe or provide snapshot):** |