Project Proposal

This proposal aims to improve the existing “Discard old builds” feature with an Advanced Build Discard Strategy plugin.

Existing plugin and scenarios discussion

The current “Discard old builds” feature in Jenkins has only one strategy: “Log Rotation”. The features of this strategy are limited and has the following problems or limitations:

1) It cannot keep the last N builds AND age builds at the same time:

> *These two options can be active at the same time, so you can keep builds for 14 days, but only up to a limit of 50 builds, for example. If either limit is exceeded, then any builds beyond that limit will be discarded.*

For example, if you want to keep the last 100 builds forever, and at the same time delete builds after the 100th build after 14 days, you cannot do it with the existing “Log Rotation”. As soon as the 14 days condition is met, all builds older than 14 days are deleted, even the most recent 100 builds you want to keep.

2) It is not able to delete the external workspaces created by the external workspace manager.

3) Users are not able to keep builds that are part of a branch that lead to production (see JENKINS-50887)

Another scenario which the Advanced Build Discard Strategy plugin should support is the following:

> *If you have build pipelines, which contain stage name like “Deploy to Test” and “Deploy to Production” only few of your instances reach Production because of failed builds or quality gates. All of the others are not relevant for documentation for an auditor and so could be deleted by the discarder to free up space. All instances which reach Production should be kept. Maybe there are other scenarios, so maybe the pipeline instances should set explicitly set a filter value. The build discarder should provide a pipeline step to set the filter value and the build discarder should contain several detail configuration sections for handling each of this filter values, as the discarding options could be different, eg. builds with value “production” should be kept for ever, builds with value “qa” should be kept for one year and so on. If the section for the value does not exist, the main filter section is used for discarding the build. Non discarding the build has to be one option of the detail configuration section. Detail section could later be changed from non discarding to another definition.*
>
> *An example of the use case described in the previous paragraph is this Jenkinsfile: https://github.com/tkleiber/de.kleiber.devserver.jenkins.build-discarder*

4) Workspace of Multi Branch Pipeline are not deleted by Build Discarder.

It is important to highlight that the build discarder is concerned with completed builds, meaning build that have run to their completion. Build that are “paused” (for example with an input step), should never be discarded. That responsibility falls under the Milestones plugin. Note that the milestones plugin has bugs of its own, (for example JENKINS-49447). However the Milestones plugin is not in the scope of this project.

Proposed new plugin discussion

The buildDiscarder feature is a Jenkins Extension, so it can be implemented as a plugin.

Here is a list of the desired features:

- Be compatible with Jenkins Configuration as Code
- Many ways to indicate builds or build elements (artifacts, reports, external workspaces) to keep:
    - Keep builds or build elements based on status
    - Keep builds or build elements based on a build property (set by the build itself like a flag)
    - Keep builds or build elements based on their position in the build history (e.g. the most recent 100)
    - Keep builds or build elements based on their branch
    - Keep builds or build elements based on their age
    - others as suggested by the community and by the student
- Many ways to indicate builds to discard
    - Discard builds or build elements based on status
    - Discard builds or build elements based on a build property (set by the build itself like a flag)
    - Discard builds or build element based on a build property
    - Discard builds or build elements based on their position in the build history
    - Discard build or build elements based on their branch (e.g. all builds leading to a branch that is meged could be deleted)
    - Discard builds based on their age
    - others as suggested by the community and by the student (build parameter)
- The ability to specify what to discard:
    - the build itself (including it artifacts, external workspaces, etc.)
    - only the artifacts of the build
    - only the external workspace of the build
    - others? html reports? (maybe html reports are included in build artifacts)
- The user should be able to chain and re-order multiple discard strategies rather than picking just one. Flexibility for user is of paramount importance in this project.

Possible implementation

It may be possible to implement the plugin as linear system. The linear system would consist of a series of filters, followed a build discarder. The filters would pick or block builds from the history based on some conditional (age of the build, build position in history, flag, build parameter value, etc.), and the output of the filters would be a list of builds. This list enters the build discarder where they are subject to the build discard actions. The user would control the filters, the filter order, and the discarder actions.

It should be possible for the user to define multiple filter sequences:
Filter sequence 1: filter A, filter B, discard
Filter sequence 2: filter C, discard
Filter sequence N: ...

Another possible implementation is to let the user pick filters in any order, and whatever builds come out of the filtering chain are then subject to the discard actions. The basic principle of a filter is to either let a build pass through the filter, or block the build, based on the filter criteria.

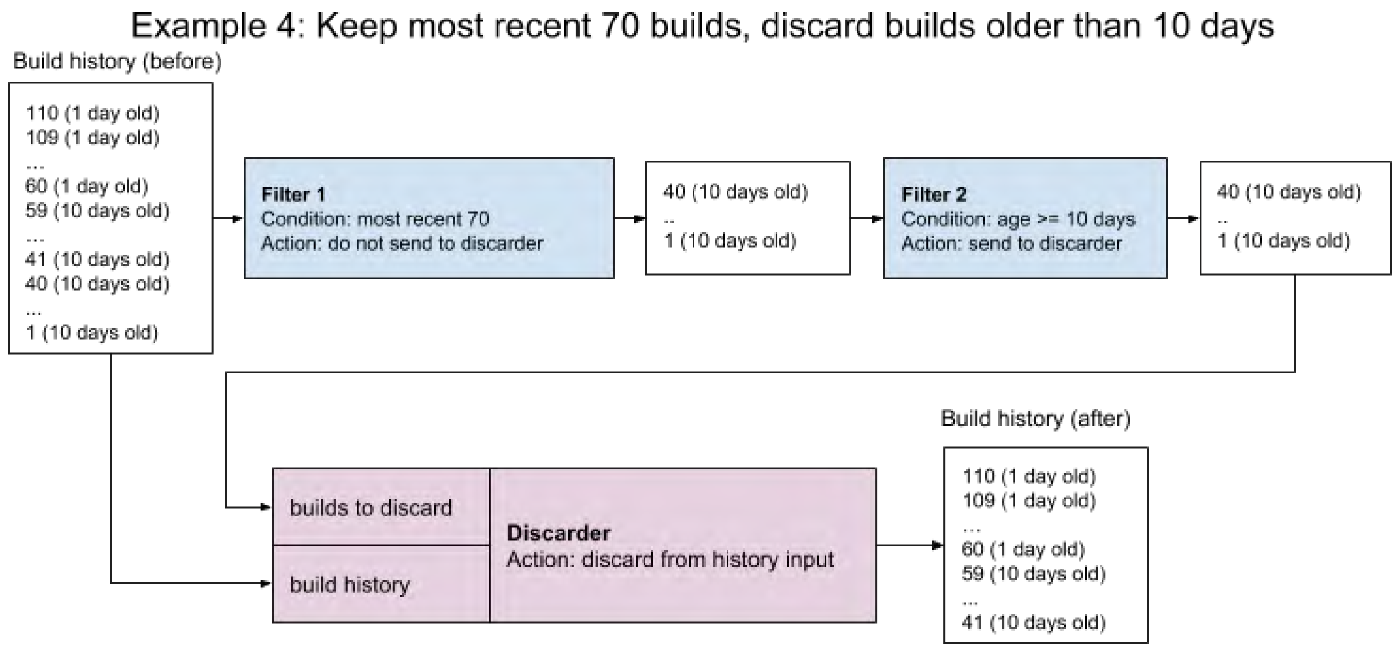
Example 1: “discard failed builds artifacts”
Build history: { 1 (succeed), 2 (fail), 3 (succeed), 4 (succeed), 5 (fail) }
Filter condition: “build status = succeed”
Filter action: “block”
After the filter: { 2 (fail), 5 (fail) }
Discard action: discard artifacts
After discard, the build history is: { 1 (succeed), 2 (fail, artifact deleted), 3 (succeed), 4 (succeed), 5 (fail, artifacts deleted) }

Example 2: “discard builds older than 5 days, but always keep the 3 most recent builds no matter their age”
Build history: { 1 (4 days old), 2 (11 days old), 3 (12 days old) , 4 (13 days old), 5 (14 days old) }
Filter condition: “build number <= 3”
Filter action: “block”
After the filter: { 4, 5 } (builds 1, 2, 3 are blocked by the filter)
Discard action: discard builds older than 5 days
After discard, the build history is: {1, 2, 3 } (only builds 4 and 5 are discarded because the discard action only applies to the list after the filter.

Example 3: “discard builds with a flag=false only if they are older than 5 days, always keep the most recent build regardless of its age or flag value” (Note: the flag is a build input parameter)
Build history: {
  #1 (flag=false, age=0 day)
  #2 (flag=false, age=3 days),
  #3 (flag=true, age=4 days),
  #4 (flag=false, age=6 days)
  #5 (flag=true, age=7 days)
}
For this example we will need three filters:
Filter 1 condition: “build number <= 1”
Filter 1 action: block
After filter 1: { 2, 3, 4, 5 } (always keep the most recent build means we do not want to pass build #1 to the discarder)
Filter 2 condition: flag=false
Filter action: pass through
After filter 2: { 2, 4 }
Filter 3 condition: build age >= 5 days
Filter 3 action: pass through
After filter 3: { 4 }
Discard action: discard build
After discard, the build history is: { 1, 2, 3, 5 }

The point of these examples is to show that the filters (condition and action), and the ordering of the filters, should be under the control of the user. The plugin should not impose a discard strategy, it should let the user build the discard strategy. Of course, the plugin needs to have some pre-built filter conditions, filter actions (block or pass through), and pre-build discard actions (delete build, delete build artifacts, etc.).

Example 4:



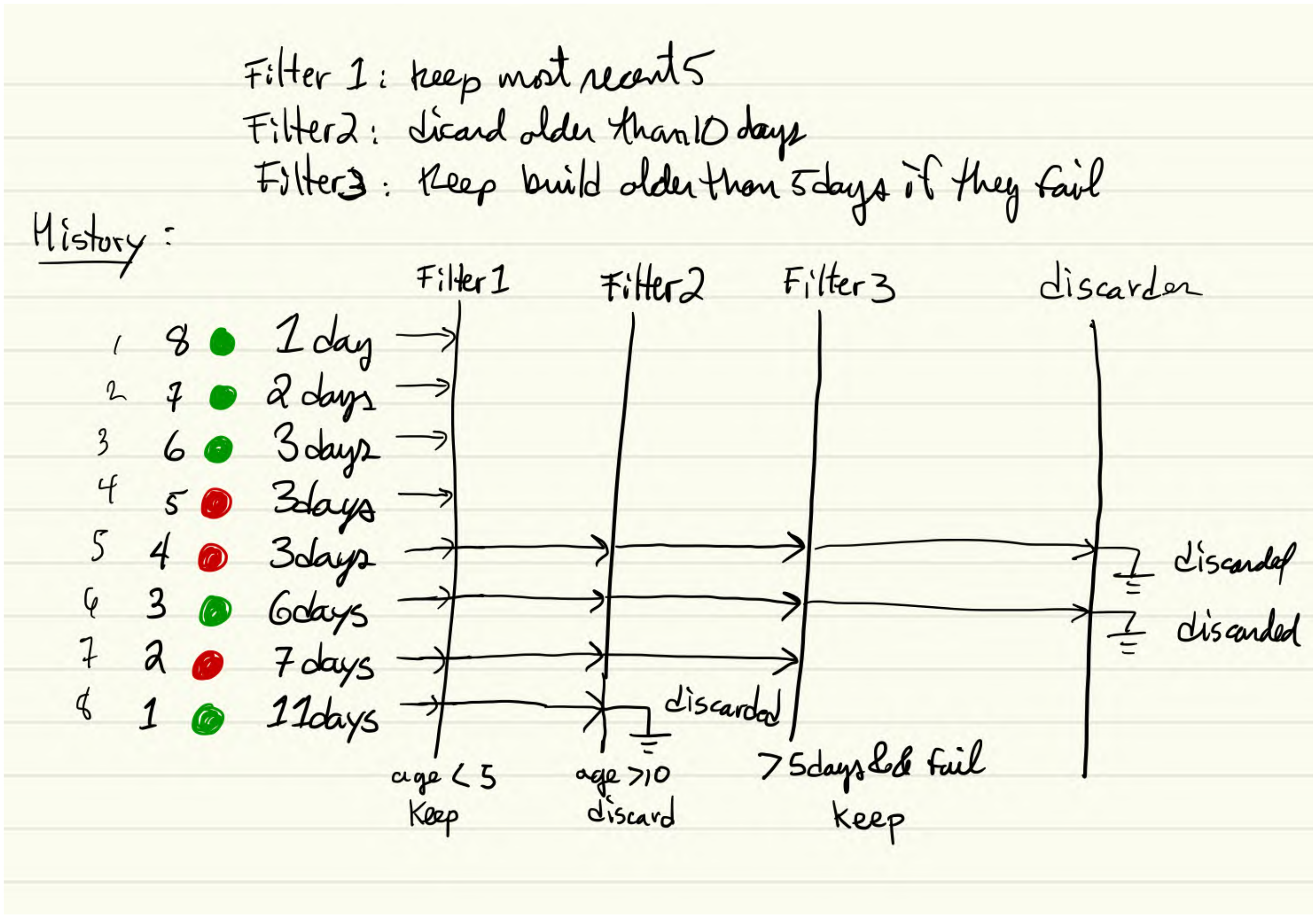
Example 5:
FILTER 1 - Keep most recent 5 builds
FILTER 2 - Discard builds older than 10 days
FILTER 3 - Keep builds older than 5 days only if they had failed

Build history: { 1(succeed-1 days old), 2(succeed-2 days old), 3(succeed-3 days old), 4(failed-3 days old), 5(failied-3 days old), 6(succeed-6 days old), 7(failed-7 days old), 8(succeed,11 days old)}

After FILTER 1, set of builds that will not go to discarder - ***{1,2,3,4,5}***
After FILTER 2,(which is basically the discarder), set of builds discarded - ***{8}***

NOTE HERE - we are left with build number 6 and 7 which have neither been selected for preservation nor for discard. So,
After FILTER 3, set of builds that will not be discarded - ***{7}***
After FILTER 3, set of builds that will be discarded - ***{6}***

FINAL RESULT - ***{1,2,3,4,5,7}***



Expectation

The student is expected to experiment with the current discard old build feature to understand how it works, with and without artifacts. The student is expected to study the materials and write a proposal on how this plugin can be implemented. Having diagrams and a mock UI for how it would look in Jenkins would make the proposal much better.

Comparable solutions

There exists a "Discard Old Builds plugin", but it is old and it is not implemented as an extension point to the buildDiscarder feature. It is implemented as a post-build step.

There exists a "Enhanced old build discarder" plugin. It is implemented as an extension point to the buildDiscarder. But it implements a single feature only. Users need additional features as listed above.

Quickstart

Study the source code for the enhanced old build discarder for a simple example on how to implement an extension point as a plugin.

Contribute to existing related plugins:

- Run Selector (upgrade the parent pom, upgrade the jenkins core version, upgrade to java 8)

Study Jenkins extension points:

- Extension Index
- Defining a new extension point
- Search the Jenkins developer mailing list to find more about extension points
- Search for Jenkins extension points in Google to find blog posts that talk about it

Study the code in the core of Jenkins, specifically:

- BuildDiscarder.java and similarly named files in the same folder.
- LogRotator.java
- Documentation on using buildDiscarder in Pipeline (search for buildDiscarder under the options)

Here is a discard build performance issue that was raised by a user:

- JENKINS-19939 - Discard Old Builds I/O Performance Issue

Links

- build discarder discussions in the developer mailing list (search for build discard too - get ideas from those discussions)
- Prior proposals (for reference only - those proposals have problems and mistakes):
    - This 2018 proposal and this 2019 proposal were abandoned because a Pipeline Step is an incorrect solution.
- Office hours discussion on the build discarder plugin on this youtube link

Newbie-friendly issues

- JENKINS-13878- [Max # of builds to keep] can't be configured as global variable

Skills to improve/study

- Java
-

Project Metadata

Created on: Jan 30, 2019
Goal: Improve the existing “Discard old builds” feature with an Advanced Build Discarder Strategy plugin
Champion: Martin d’Anjou
Champion Github Id and link: https://github.com/martinda

Champion Jenkins JIRA/LDAP id: deepchip

Champion Time Zone: UTC-4
Champion Role: **mentor**

Technical advisor: <If somebody is helping with the technical side, fill in information about the advisor>
Advisor Github Id and link: https://github.com/< id here and test the link please>
Advisor Time Zone: <enter the advisor's UTC timezone here (optional)> (use https://www.timeanddate.com/time/map/ to find it, this help schedule meetings)

Project Category: plugin
SIG: GSoC
Sub-project: n/a
Project Gitter chat room: https://gitter.im/jenkinsci/gsoc-build-discarder-project

Potential Mentors

The potential mentors are:

1) Martin d'Anjou (https://github.com/martinda)
2) Arnab Banerjee (https://github.com/arnab1896)
3)