

Top 10 Best Practices for Jenkins Pipeline Plugin

apemberton - 27 Jun 2016

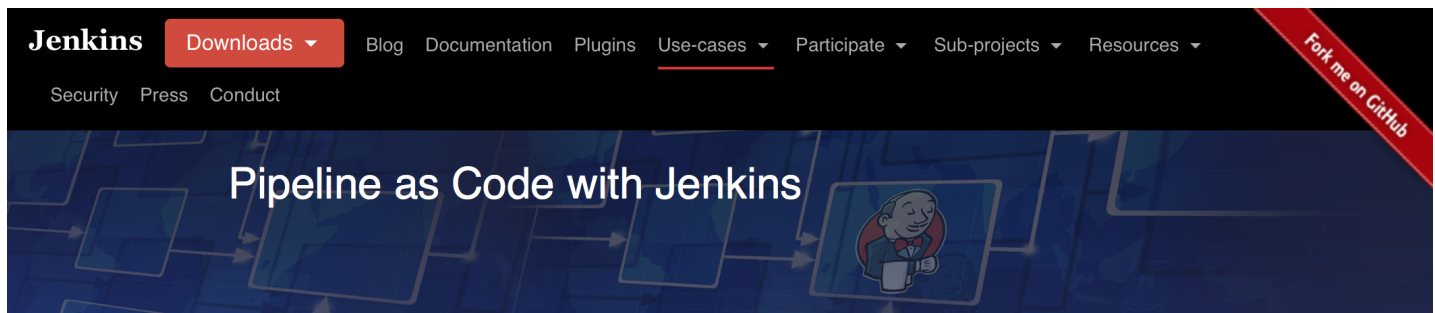
Tweet

Like 76

Share

G+

The Jenkins Pipeline plugin is a game changer for Jenkins users. Based on a Domain Specific Language (DSL) in Groovy, the Pipeline plugin makes pipelines scriptable and it is an incredibly powerful way to develop complex, multi-step DevOps pipelines. This document captures some definite Do's and Don'ts of writing Jenkins Pipelines - with code examples and explanations.



The default interaction model with Jenkins, historically, has been very web UI driven, requiring users to manually create jobs, then manually fill in the details through a web browser. This requires additional effort to create and manage jobs to test and build multiple projects, it also keeps the configuration of a job to build/test/deploy separate from the actual code being built/tested/deployed. This prevents users from applying their existing CI/CD best practices to the job configurations themselves.

Pipeline

With the introduction of the [Pipeline plugin](#), users now can implement a project's entire build/test/deploy pipeline in a [Jenkinsfile](#) and store that alongside their code, treating their pipeline as another piece of code checked into source control.

The Pipeline plugin was inspired by the [Build Flow plugin](#) but aims to improve upon some concepts explored by Build Flow with features like:

- the ability to suspend/resume of executing jobs.
- checking the pipeline definition into source control ([Jenkinsfile](#))
- support for extending the domain specific language with additional, organization specific steps, via the "global library" feature

In an adjacent space is the [Job DSL plugin](#) which is worth mentioning as well.

Jenkins ♥ Continuous Delivery Articles


[Getting started with Jenkins Workflow](#)
udaypal.com

[Multibranch Workflows in Jenkins](#)
jenkins-ci.org

Continuous Delivery plugins for Jenkins

 **Pipeline plugin**
allows creating Pipeline scripts for defining build/test/deploy stages of a delivery pipeline

 **Pipeline Utility Steps plugin**
provides a number of additional, useful, steps to the Pipeline DSL

 **Job DSL plugin**
creates a DSL to orchestrate job creation

1. Do: Use the real Jenkins Pipeline

Don't use older plugins like Build Pipeline plugin or Buildflow plugin. Instead, use the real Jenkins Pipeline suite of plugins (<https://wiki.jenkins-ci.org/display/JENKINS/Pipeline+Plugin>).

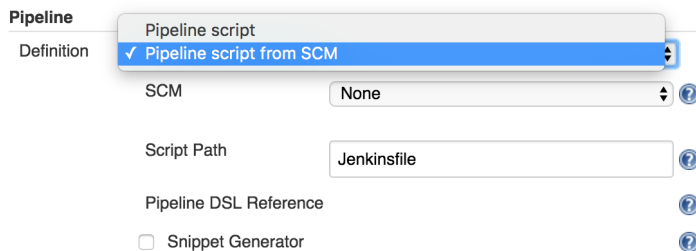
- Why? The Pipeline plugin is a step change improvement in the underlying job itself. Unlike freestyle jobs, Pipeline is resilient to Jenkins master restarts and also has` built-in features that supersede many older plugins previously used to build multi-step, complex delivery pipelines.

More information on getting started is available at <https://jenkins.io/solutions/pipeline/> (<https://jenkins.io/solutions/pipeline/>).

2. Do: Develop your pipeline as code

Use the feature to store your Jenkinsfile in SCM then version and test it like you do other software.

- Why? Treating your pipeline as code enforces good discipline and also opens up a new world of features and capabilities like multi-branch, pull request detection and organization scanning for GitHub and BitBucket.



You should also call your Pipeline script the default name: Jenkinsfile and start the following script header, so your IDE, GitHub and other tooling recognize it as Groovy and enable code highlighting:

```
#!/groovy
```

3. Do: All work within a stage

Any non-setup work within your pipeline should occur within a stage block.

- Why? Stages are the logical segmentation of a pipeline. Separating work into stages allows separating your pipeline into distinct segments of work.

Example:

```
stage 'build'
//build
stage 'test'
//test
```

And better still: the Pipeline Stage View plugin visualizes stages as unique segments of the pipeline:



4. Do: All material work within a node

Any material work within a pipeline should occur within a node block.

- Why? By default, the Jenkinsfile script itself runs on the Jenkins master, using a lightweight executor expected to use very few resources. Any material work, like cloning code from a Git server or compiling a Java application, should leverage Jenkins distributed builds capability and run on an agent node.

Example:

```
stage 'build'
node{
    checkout scm
    sh 'mvn clean install'
}
```

5. Do: Work you can within a parallel step

Pipeline offers a straight-forward syntax for branching your pipeline into parallel steps. Use it!

- Why? Branching work in parallel will allow your pipeline to run faster, shifting your pipeline steps to the left, and getting feedback to developers and the rest of your team faster.

Example:

```
parallel 'shifting':{
    //everything
}, 'left':{
    //I can
}
```

Bonus tip: use the Parallel Test Executor plugin (<https://wiki.jenkins-ci.org/display/JENKINS/Parallel+Test+Executor+Plugin>) to have Jenkins automatically determine how to run your xUnit compatible tests in optimally parallel buckets! Read more information on Parallel Test Execution on the CloudBees Blog (<https://www.cloudbees.com/blog/parallelism-and-distributed-builds-jenkins>).

6. Do: Acquire nodes within parallel steps

Why? One of the main benefits of parallelism in a pipeline is: to do more material work (see Best Practice #4)! You should generally aim to acquire a node within the parallel branches of your pipeline.

Example:

```
parallel 'integration-tests':{
    node('mvn-3.3'){ ... }
}, 'functional-tests':{
    node('selenium'){ ... }
}
```

7. Don't: Use input within a node block

While you can put an input statement within a node block, you definitely shouldn't.

- Why? The input element pauses pipeline execution to wait for an approval - either automated or manual. Naturally these approvals could take some time. The node element, on the other hand, acquires and holds a lock on a workspace and heavy weight Jenkins executor - an expensive resource to hold onto while pausing for input.

So, create your inputs outside your nodes.

Example:

```
stage 'deployment'
input 'Do you approve deployment?'
node{
    //deploy the things
}
```

8. Do: Wrap your inputs in a timeout

Pipeline has an easy mechanism for timing out any given step of your pipeline. As a best practice, you should always plan for timeouts around your inputs.

- Why? For healthy cleanup of the pipeline, that's why. Wrapping your inputs in a timeout will allow them to be cleaned-up (i.e., aborted) if approvals don't occur within a given window.

Example:

```
timeout(time:5, unit:'DAYS') {  
    input message:'Approve deployment?', submitter: 'it-ops'  
}
```

9. Don't: Set environment variables with env global variable

While you can edit some settings in the env global variable, you should use the withEnv syntax instead.

- Why? because the env variable is global, changing it directly is discouraged as it changes the environment globally, so the withEnv syntax is recommended.

Example:

```
withEnv(["PATH+MAVEN=${tool 'm3'}/bin"]) {  
    sh "mvn clean verify"  
}
```

10. Do: Prefer stashing files to archiving

Before the stash capability was added to Pipeline DSL, archives were the best way to share files between nodes or stages in a pipeline. If you just need to share files between stages and nodes of your pipeline, you should use stash/unstash instead of archive.

- Why? Stash and unstash are designed for sharing files, for example your application's source code, between stages and nodes. Archives, on the other hand, are designed for longer term file storage (e.g., intermediate binaries from your builds).

Example:

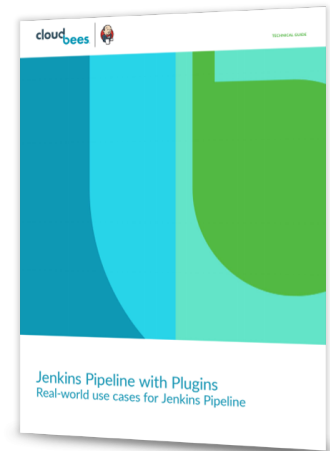
```
stash excludes: 'target/', name: 'source'  
unstash 'source'
```

Summary

Thanks for reading! The new Jenkins Pipeline plugin is gaining more and more traction, especially with the launch of Jenkins 2.0 (<https://jenkins.io/2.0/>). I'm sure there will be more and more best practices to come as developers across the world develop their DevOps pipelines with Jenkins. We will post follow-up blogs and more examples soon!

Learn More

(<https://pages.cloudbees.com/Jenkins-Pipeline-with-plugins-Technical-guide>) Jenkins is one of the preeminent automation tools. Jenkins is extensible by design, using plugins. Plugins are what give Jenkins its great flexibility for automating a wide range of processes on diverse platforms. Jenkins Pipeline builds on that flexibility and rich plugin ecosystem while enabling Jenkins users to write their Jenkins automation as code. This technical guide (<https://pages.cloudbees.com/Jenkins-Pipeline-with-plugins-Technical-guide>) will show a number of common use cases for plugins with Jenkins Pipeline.



These use cases include:

- Publishing HTML reports
- Sending notifications
- Continuous delivery using Docker
- Test result interpretation and reporting

Download this technical guide (<https://pages.cloudbees.com/Jenkins-Pipeline-with-plugins-Technical-guide>) to learn more about and get started with Jenkins Pipeline!

[apemberton's blog \(/blogs/apemberton\)](#)

Related Content

Driving Digital Transformation with DevOps (</blog/driving-digital-transformation-devops>)

Viktor Farcic - 08 May 2018

Tweet

Like 0

Share



Read more (</blog/driving-digital-transformation-devops>) [Viktor Farcic's blog \(/blogs/viktor-farcic\)](/blogs/viktor-farcic)

Recipe for Running a Successful Product Portfolio Planning Summit (/blog/recipe-running-successful-product-portfolio-planning-summit)

Harpreet Singh - 15 Mar 2018

Tweet

Like 12

Share



Read more (/blog/recipe-running-successful-product-portfolio-planning-summit)

Harpreet Singh's blog (/blogs/harpreet-singh)

Predicting admissions into UCLA with a Neural Network (/blog/predicting-admissions-ucla-neural-network)

Harpreet Singh - 15 Feb 2018

Tweet

Like 0

Share



Read more (/blog/predicting-admissions-ucla-neural-network)

Harpreet Singh's blog (/blogs/harpreet-singh)



Get our blogs straight to your inbox

Subscribe to our blog posts. No spam, just our best blog posts each week!

(<https://chrome.google.com/webstore/detail/rss-subscription-extensio/nlbjncdgjeocebhnmkbbbdekmmmcbfjd>)

Recent Blogs

What's New in Declarative Pipeline 1.3: Sequential Stages (/blog/whats-new-declarative-pipeline-13-sequential-stages)

Women in DevOps: Paola Moretto (/blog/women-devops-paola-moretto)

Tuning the Engine - CI/CD Platform Monitoring with CloudBees DevOptics Run Insights (/blog/tuning-engine-cicd-platform-monitoring-cloudbees-devoptics-run-insights)

The Kubernetes Oligopoly and Jenkins X (/blog/kubernetes-oligopoly-and-jenkins-x)

CloudBees Core on Azure Kubernetes Service is now GA (/blog/cloudbees-core-azure-kubernetes-service-now-ga)

Jenkins is Showing the CI/CD Way! (/blog/jenkins-showing-cicd-way)

The Lure of the Monolith (/blog/lure-monolith)

Las Vegas, The Grand Canyon and Medallia's Project 24 (/blog/las-vegas-grand-canyon-and-medallia%E2%80%99s-project-24)

DevOps Radio: Laura Frank, CloudBees – When the Stars, Coffee, DockerCon and Code Align (/blog/devops-radio-laura-frank-cloudbees-%E2%80%93-when-stars-coffee-dockercon-and-code-align)

CloudBees Jenkins Enterprise is now GA on Elastic Container Service for Kubernetes (/blog/cloudbees-jenkins-enterprise-now-ga-elastic-container-service-kubernetes)

Guide: Get Started with Jenkins Pipeline

X

Download the Guide (<http://join.cloudbees.com/l/272242/2017-04-17/3sf8v>)

Subscribe to our Jenkins Newsletter

X

Subscribe Now (<https://pages.cloudbees.com/l/272242/2017-04-03/3r4j6>)



CloudBees is building the world's first end-to-end automated software delivery system. We provide smart solutions for continuous development, integration and delivery. By making the software delivery process more productive, manageable and hassle-free, CloudBees puts companies on the fast track to transforming great ideas into great software and delivering value to their business more quickly.

[Learn More > \(/company\)](#)

COMPANY (/company)

[Blog \(/blog\)](#)

[Careers \(/careers\)](#)

[Contact Us \(/contact-us\)](#)

[News Room \(/company/news-room\)](#)

[Company Store \(https://cloudbeesapiary.myshopify.com/\)](https://cloudbeesapiary.myshopify.com/)

SOLUTIONS (/devops)

[What is Continuous Delivery? \(/devops/continuous-delivery\)](#)

[CD with Docker \(/devops/continuous-delivery/jenkins-docker\)](#)

[CD for Government \(/devops/continuous-delivery/government\)](#)

[CD with Jenkins Pipeline \(/devops/continuous-delivery/pipeline\)](#)

[The CD Journal \(/devops/cd-journal\)](#)

PRODUCTS (/products)

[CloudBees Core \(/products/cloudbees-core\)](#)

[CloudBees CodeShip \(/products/cloudbees-codeship\)](#)

[CloudBees DevOptics \(/products/cloudbees-devoptics\)](#)

[CloudBees Jenkins Support \(/products/cloudbees-jenkins-support\)](#)

[Pricing \(/products/pricing\)](#)

RESOURCES (/resources)

[Webinars \(/resources/webinars\)](/resources/webinars)

[Whitepapers \(/resources/whitepapers\)](/resources/whitepapers)

[Collateral \(/resources/collateral\)](/resources/collateral)

[DevOps Radio \(/resources/devops-radio\)](/resources/devops-radio)

[Events \(/company/events\)](/company/events)

JENKINS (/jenkins/jenkins-cloudbees)

[About Jenkins \(/jenkins/about\)](/jenkins/about)

[Certification \(/jenkins/jenkins-certification\)](/jenkins/jenkins-certification)

[DevOps World | Jenkins World \(/devops-world\)](/devops-world)

[Newsletter \(/jenkins/newsletter\)](/jenkins/newsletter)

[Security Advisories \(/jenkins/security-advisories\)](/jenkins/security-advisories)

[Training \(/jenkins/training\)](/jenkins/training)

©CloudBees, Inc. 2010– 2018

[Privacy Policy \(/privacy-policy\)](/privacy-policy) | [Terms of Service \(/terms-service\)](/terms-service) | [Security \(/security-policy\)](/security-policy)

<https://www.facebook.com/CloudBees> (<https://twitter.com/cloudbees>)

(<https://www.linkedin.com/company/cloudbees/>)

(<https://plus.google.com/+CloudbeesHive>)

(<https://www.youtube.com/user/CloudBeesTV>)

