Using the Pipeline Plugin to Accelerate Continuous Delivery -- Part 1

*apemberton - 24 May 2016*

[inShare](javascript:void(0);)4

Jenkins is a powerful, open source automation tool with an impressive plugin architecture that helps development teams automate their software lifecycle. Jenkins is used to power many industry-leading companies’ software development pipelines. Jenkins Pipeline is a powerful, first-class feature for managing complex, multi-step pipelines. Jenkins Pipeline, a set of open source plugins and integrations, brings the power of Jenkins and the plugin ecosystem into a scriptable Domain Specific Language (DSL). Best of all, like Jenkins core, Pipeline is extensible by third-party developers, supporting custom extensions to the Pipeline DSL and various options for plugin integration.

The Pipeline plugin was formerly known as the Workflow plugin prior to version 1.13 of the plugin.



*Image 1: Pipeline Stage View UI*

In this blog series, we will provide an introduction and step-by-step guide on how to use the Pipeline plugin.

Installing the Pipeline Plugin

It is assumed that you have already installed Jenkins – either via the CloudBees Jenkins Platform or from the Jenkins open source project website. Jenkins Version 1.609.1+ is required.

* Open Jenkins in your web browser
* Navigate to Manage Jenkins > Manage Plugins
* Navigate to the Available tab, filter by Pipeline
* Select the Pipeline plugin and install
* Restart Jenkins

This blog series was written using Pipeline version 1.13. Installing the Pipeline plugin installs all necessary Pipeline dependencies and a new job type called Pipeline.

Creating a Pipeline

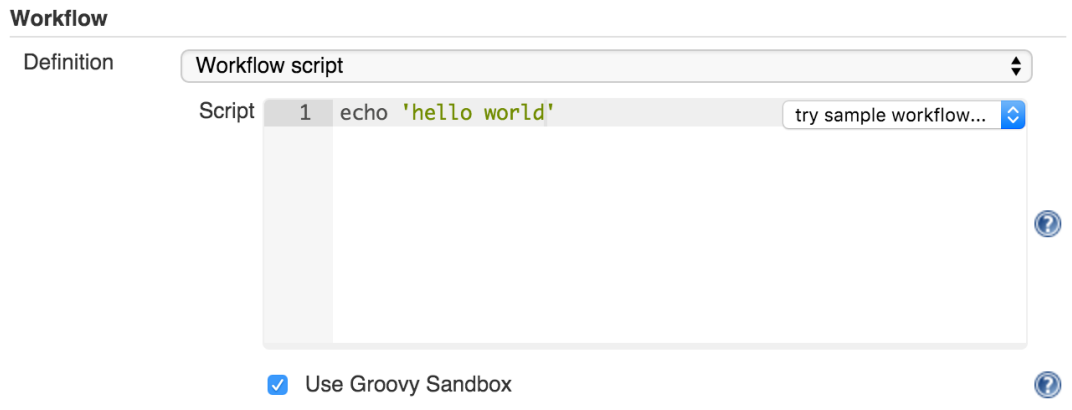
Now that you have Jenkins running and have installed the Pipeline plugin, you are ready to create your first pipeline. Create a new pipeline by selecting New Item from the Jenkins home page. First, give your pipeline a name, e.g.: “hello-world-flow.” Pipelines are simple Groovy scripts, so let’s add the obligatory Hello World. Add a pipeline to the Pipeline script textarea:

echo 'Hello world'

Now save your pipeline, ensuring the Use Groovy Sandbox option is checked (more details to follow on this setting). Click Build Now to run your pipeline.

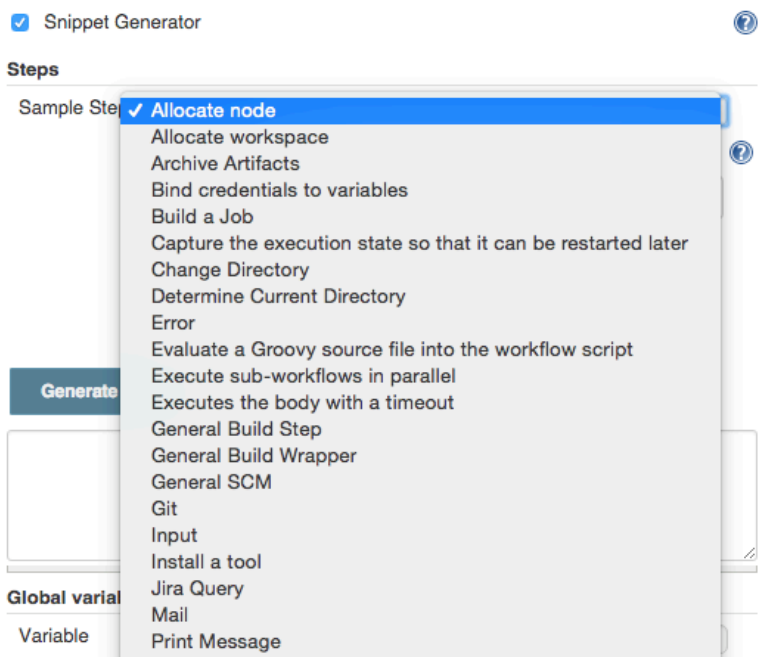
Editing Your Pipeline

Because pipelines are simple text scripts, they are easy to edit. As you’ve seen, pipelines can be edited directly in the Jenkins UI when configuring your pipeline.



Using the Snippet Generator

To make editing your pipelines easier, use the Snippet Generator. The Snippet Generator is dynamically populated with the latest Pipeline steps. Depending on the plugins installed in your environment, you may see more available steps.



Loading External Pipeline Scripts

Because pipelines are text assets, they are ideal to store in a source control system. Pipelines can be edited in your external IDE then loaded into Jenkins using the Pipeline Script from SCM option.

Building Your Pipeline

Now that you’ve created a pipeline, let’s continue to build on it. For a complex flow, you should leverage Jenkins’ job scheduling queue:

node{ sh 'uname' }

The concept of a node should be familiar to Jenkins users: node is a special step that schedules the contained steps to run by adding them to Jenkins’ build queue. Even better, requesting a node leverages Jenkins’ distributed build system. Of course, to select the right kind of node for your build, the node element takes a label expression:

node('unix && 64-bit'){ echo 'Hello world' }

The node step also creates a workspace: a directory specific to this job where you can check out source code, run commands and do other work. Resource-intensive work in your pipeline should occur on a node. You can also use the ws step to explicitly ask for another workspace on the current agent, without grabbing a new executor slot. Inside its body, all commands run in the second workspace.

Checking Out Code

Usually, your pipelines will retrieve source code from your source control server. Pipeline has a simple syntax for retrieving source code, leveraging the many existing SCM plugins for Jenkins.

checkout([$class: 'GitSCM', branches: [[name: '\*/master']], userRemoteConfigs: [[url: '[http://github.com/cloudbees/todo-api.git](https://github.com/cloudbees/todo-api.git)']]])

If you happen to use a Git-based SCM - for example, GitHub - there’s an even further simplified syntax:

git '<https://github.com/cloudbees/todo-api.git>'

Running Your Pipeline

Because pipelines are built as Jenkins jobs, they can be built like other jobs. You can use the Build Now feature to manually trigger your build on-demand or set up triggers to execute your pipeline based on certain events.

Adding Stages and Steps

Stages are usually the top most element of Pipeline syntax. Stages allow you to group your build step into its component parts. By default, multiple builds of the same pipeline can run concurrently. The stage element also allows you to control this concurrency:

stage 'build'

node{ … }

stage name: 'test', concurrency: 3

node{ … }

stage name: 'deploy', concurrency: 1

node{ … }

In this example, we have set a limit of three concurrent executions of the test stage and only one execution of the deploy stage. You will likely want to control concurrency to prevent collisions (for example, deployments).

Newer builds are always given priority when entering a throttled stage; older builds will simply exit early if they are preempted.

General Build Steps

Within your stages, you will add build steps. Just like with Freestyle Jenkins jobs, build steps make up the core logic of your pipeline. Jenkins Pipeline supports any compatible Build Step and populates the snippet generator with all available Build Steps in your Jenkins environment.

step([$class: 'JavadocArchiver', javadocDir: 'target/resources/javadoc', keepAll: false])

step([$class: 'Fingerprinter', targets: 'target/api.war'])

Scripting Jenkins

Pipeline supports executing shell (\*nix) or batch scripts (Windows) just like freestyle jobs:

sh 'sleep 10'

bat 'timeout /t 10'

Scripts can integrate with various other tools and frameworks in your environment - more to come on tools in the next blog post.

[*Using the Pipeline Plugin to Accelerate Continuous Delivery — Part 1*](https://www.cloudbees.com/blog/using-pipeline-plugin-accelerate-continuous-delivery-part-1)[*Using the Pipeline Plugin to Accelerate Continuous Delivery — Part 2*](https://www.cloudbees.com/blog/using-pipeline-plugin-accelerate-continuous-delivery-part-2) *Using the Pipeline Plugin to Accelerate Continuous Delivery — Part 3*