**Chapter 16. Troubleshooting**

There can be a steep learning curve associated with migrating to Jenkins 2. In this chapter, I’ll attempt to explain some of the common or more complex issues you may run into, or point you to other sections of the book where they are explained.

This is more a varied collection of tips and processes than a consistent flow of information, but this is by design, since the best method to troubleshoot a situation can vary widely depending on the circumstances.

Let’s start out by looking at how we can drill in to get more details about the steps in our pipeline.

**Diving into Pipeline Steps**

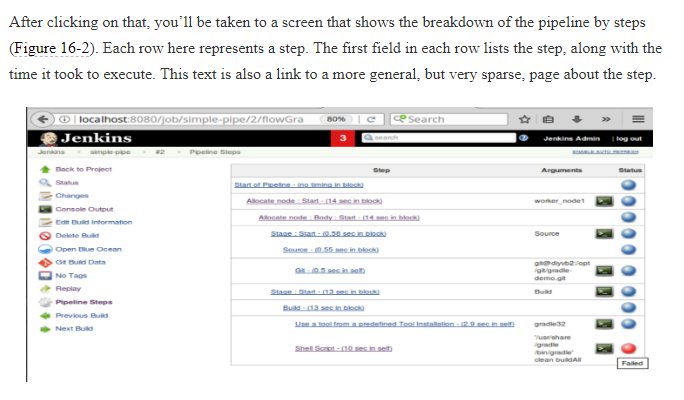
While the Stage View provides a level of separation and detail on pieces of the pipeline, there may be times when it is beneficial to examine processing at an even lower level to troubleshoot an issue. The *Pipeline Steps* view provides this capability.

To get to the Pipeline Steps view, you first need to go into the output screen for a single run of a build. You can use a URL of the form:

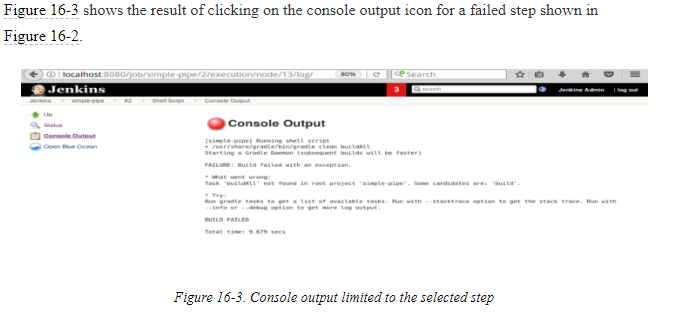
http://<jenkins-location>/job/<job-name>/<build-number>

or simply click on the build number in the Build History section of the Stage View page. This will take you to the specific output page for that build. On that page, in the menu on the left, will be a Pipeline Steps item



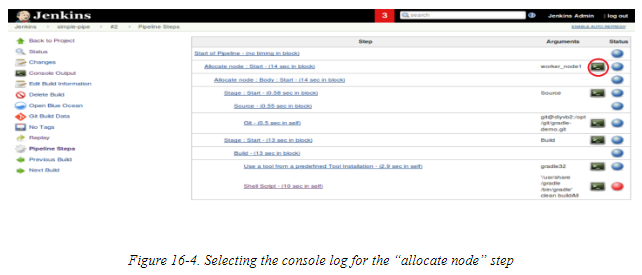


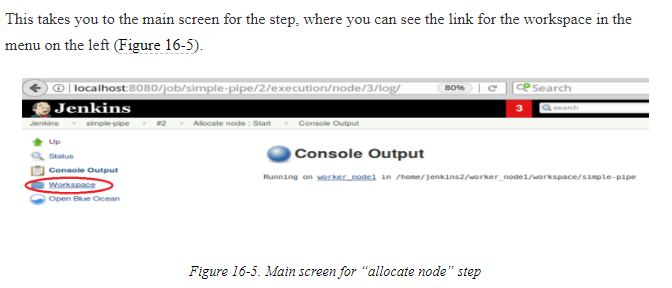
On the righthand side of the row are any arguments that the step received, a screen icon that links to the console output (if that makes sense for the step), and a status indicator of whether the step was successful or not.

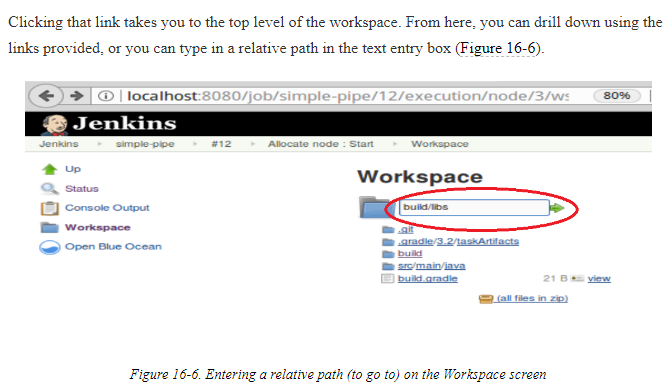
With these data points, you can verify that steps got the expected arguments, see which steps used the most/least time, and view only the portion of the console output that pertain to a particular step. 

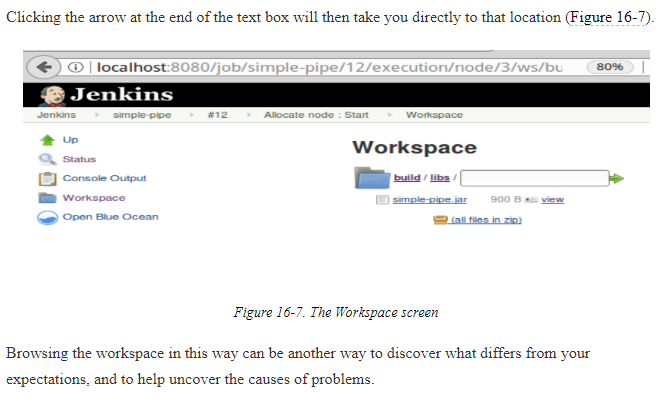
The Pipeline Steps screen is also the way to get to the workspace from the web interface. Prior to Jenkins 2, there was a Workspace link on the output page for a build. This link is not surfaced any longer on that page. Instead, you have to dig deeper through this area to find it.

Since a workspace is associated with a node, you first click on the console output icon for a pipeline step associated with allocating a node









Some problems are not caused by the way the steps are organized or used in the pipeline, but by trying to use steps, methods, or libraries that aren’t serializable—that is, not able to save their state. This violates a requirement of Jenkins 2. Dealing with errors and problems around serialization is the topic of our next section.

# Dealing with Serialization Errors

One of the features of Jenkins pipelines is the ability to recover from restarts. This is implemented in Pipeline by transforming how control flows as the pipeline is executing, and regularly writing the pipeline’s state to disk, so there is data available to restart if needed.

In order for this to work effectively, the pipeline must use objects and methods that are themselves serializable—but since not all methods and objects are. Therefore, you may encounter cases where your pipeline will not execute due to something not being serializable. In this section, we’ll discuss how to handle that situation.

First, it’s helpful to understand a bit about how pipeline flow is handled in Jenkins.