**Part 2: Working with the Blue Ocean Editor**

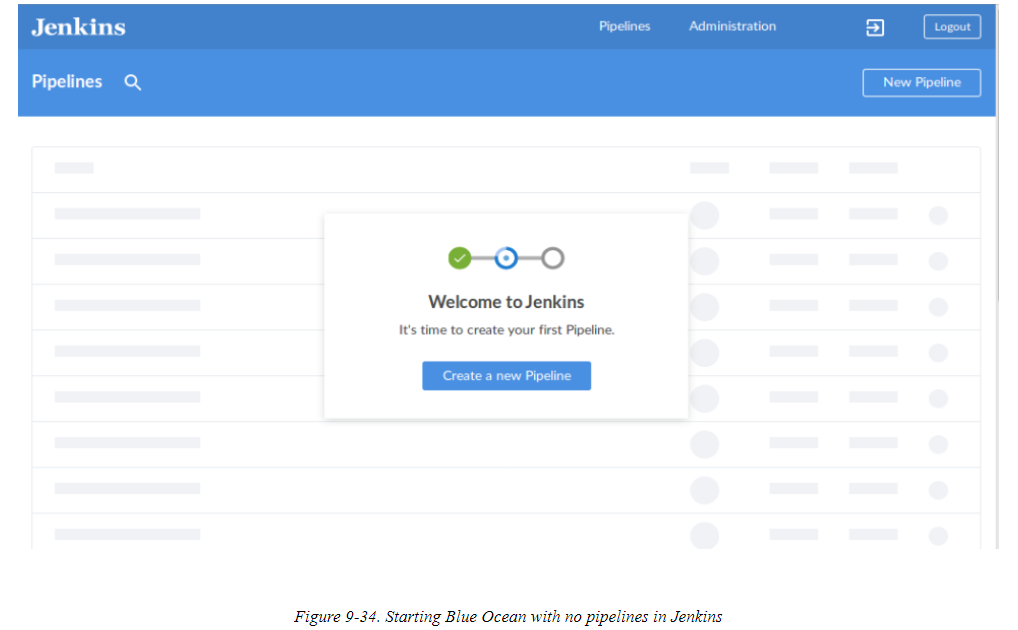
In this second part of the chapter, we’ll discuss the other key aspect of Blue Ocean, the pipeline editor. This editor allows you (within limits) to create and update pipelines, and parts of pipelines, through a more visual interface.

We’ll look at a couple of use cases: creating a new pipeline from a project without an existing Jenkinsfile, and using the editor to add or edit content in an existing pipeline.

## **Creating a New Pipeline Without an Existing Jenkinsfile**

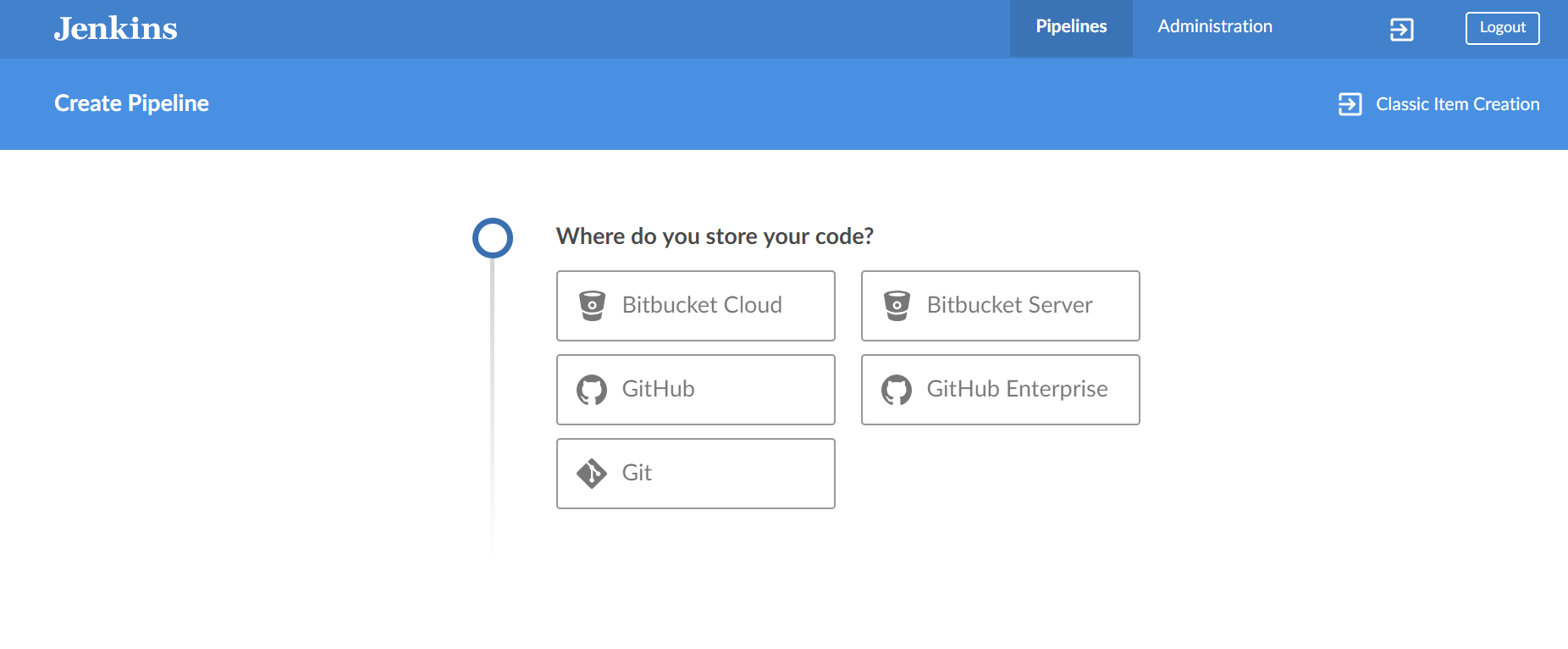
Having an existing Jenkinsfile that defines your pipeline gives you a foundation for making changes or adding functionality through the pipeline editor. But you can also use the pipeline editor to create a completely new pipeline where there wasn’t one before. In fact, this is the default workflow when you use the Blue Ocean interface to create a new pipeline. Let’s see how this works.

If we were to start out with a Jenkins instance that did not have any pipelines yet and opened Blue Ocean, we’d get a screen like the one in [Figure 9-34](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_start_BO_nopipe). To create a new pipeline, we could then simply click the button in the dialog.

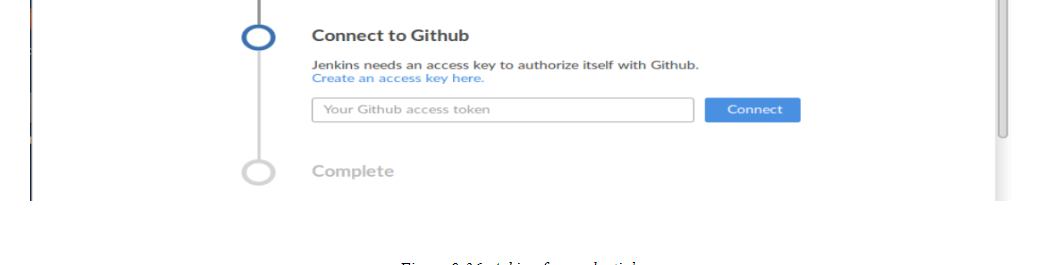


For the case where we already have existing pipeline projects in Jenkins, those will show up in the pipelines list on the dashboard (as previously described). In that case, we can create a New Pipeline by clicking the New Pipeline button on the Blue Ocean dashboard (as shown in the upper-right corner of above pic )

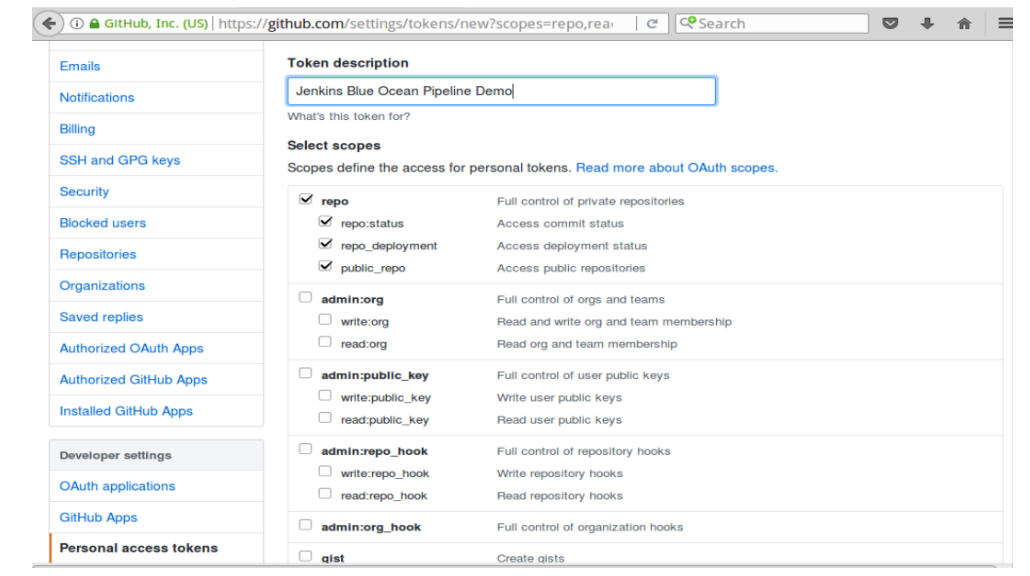
From there, Blue Ocean prompts us to choose where the source repository is that we want to use Below. Currently, the options include Git (meaning a Git repo we have access to), GitHub (the public Git hosting service), GitHub Enterprise, Bitbucket Cloud, and Bitbucket Server. For our example here, we’re going to start with a set of code on GitHub, so we would click the GitHub button.



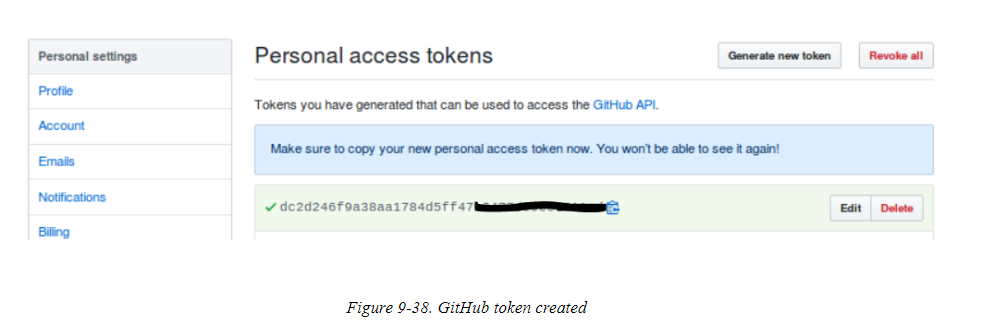
To connect to GitHub, Jenkins needs an access token ([Figure 9-36](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_ask_for_creds)). This is a token that you generate on GitHub yourself.



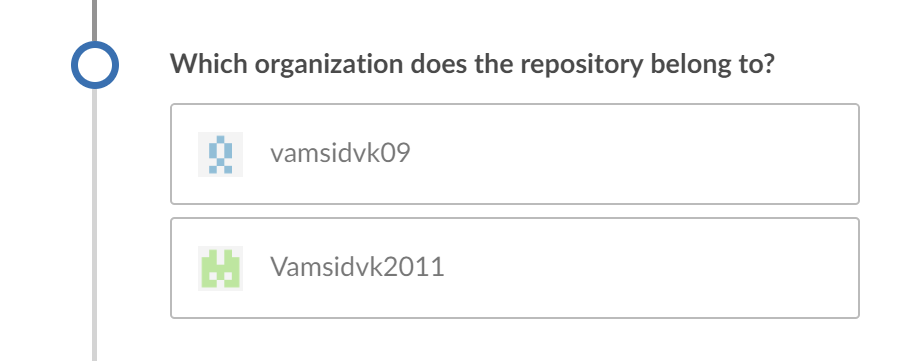
If you already have a token, you can paste it in here. If not, you can click the “Create an access key here” link to be taken directly to GitHub to create one. If you do that, you’ll be prompted for your GitHub login information and then taken to the screen for creating a token ([Figure 9-37](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_create_github_token)).

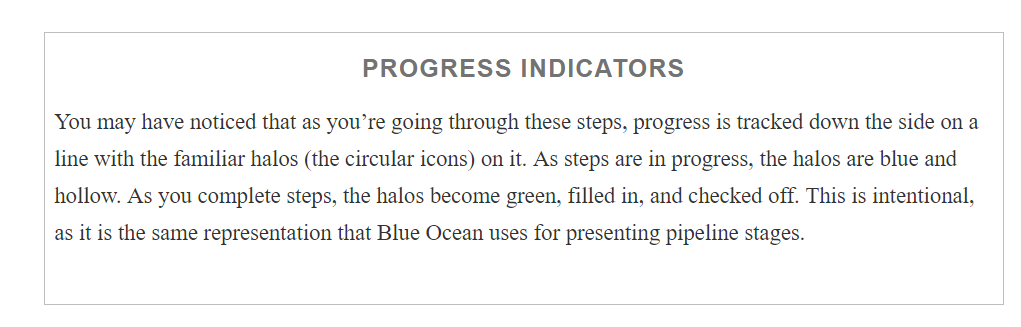
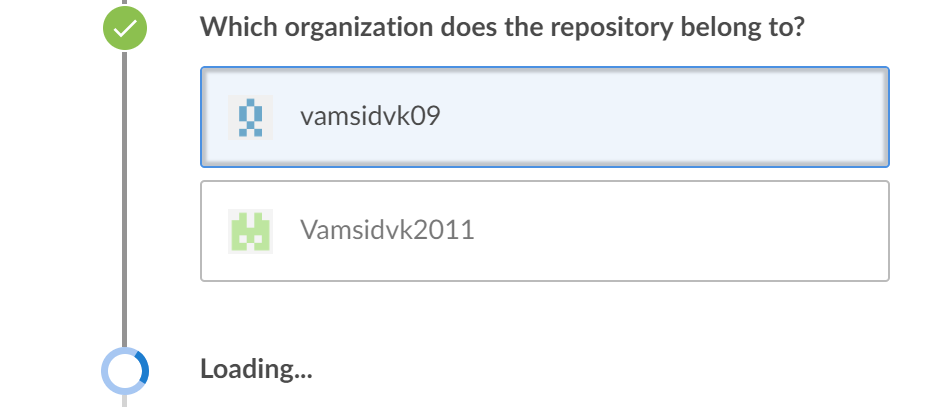


The nice thing about this is that the permissions that Jenkins needs are already selected on this screen. To complete the process, you need to enter something in the “Token description” field, scroll to the bottom, and click the “Generate token” button. Then a token will be generated for you

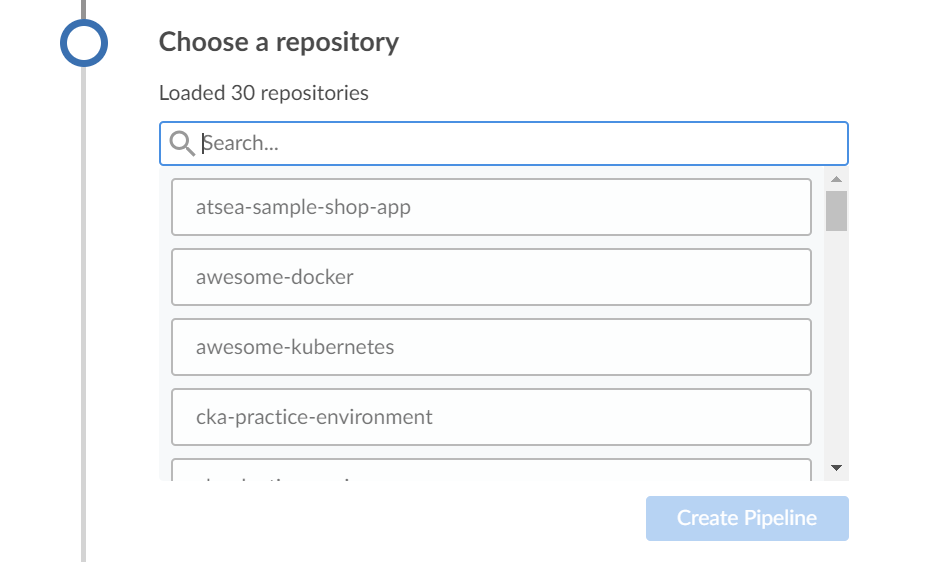


Copy this token and then paste it back into the Jenkins screen. Click the “Connect” button, and Jenkins will access your GitHub account and present a list of organizations for you to choose from, if you have more than one

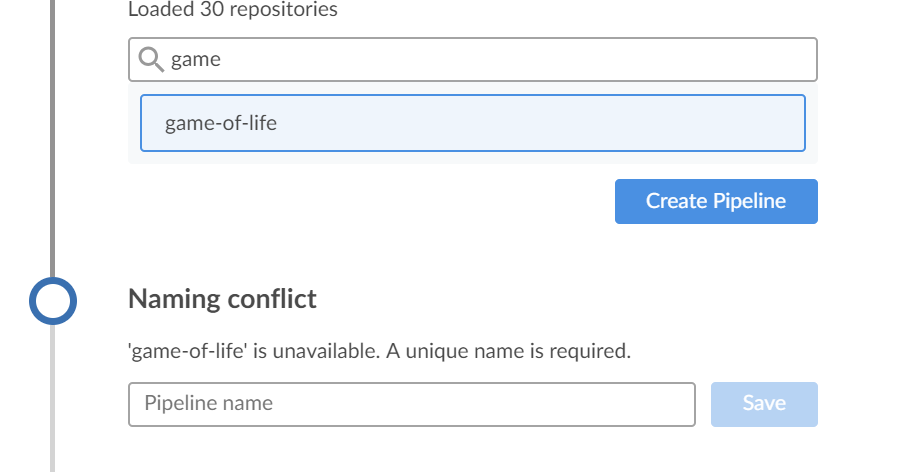




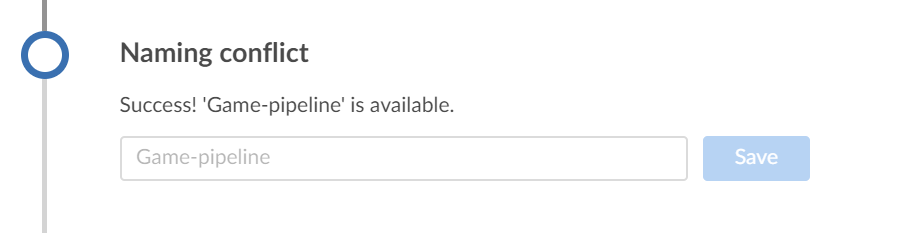
After selecting an organization, you are presented with a list of repositories within that organization ([Figure 9-40](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_select_repo_from_GHO)). There is a search field to filter the list if desired. From the list, you can choose a repository and then click the Create Pipeline button to get started.



In our example, we’re going to choose the “game-of-life” repository. If the repository you choose already has a Jenkinsfile in it, then Jenkins will automatically attempt to create an instance of the pipeline, defined in that Jenkinsfile, and run it for you. In this case, there is no initial Jenkinsfile in this project, so Jenkins just reports that and provides a button to create a new pipeline



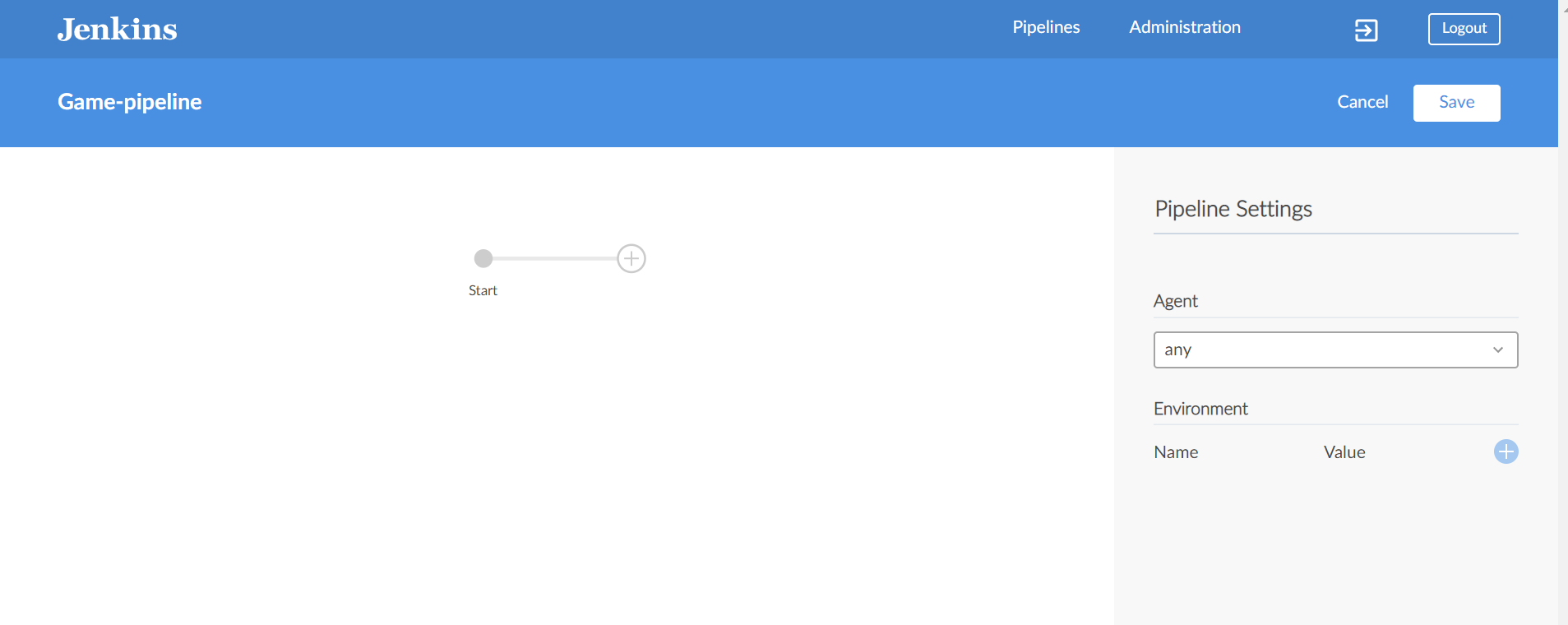
After entering the pipeline which has naming conflict then we have given a new name as below



Clicking the Create Pipeline button takes us into the Blue Ocean pipeline editor, which we’ll discuss next.

## **Working in the Editor**

After telling Jenkins which GitHub project we want to base our new pipeline on, Jenkins puts us into the Blue Ocean pipeline editor. The initial screen is shown in [Figure 9-42](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_init_scrn_editor). The basic idea here is that instead of just typing in all of our Declarative Pipeline code, we’ll use a combination of GUI elements (such as selecting items from a list) and typing to create the main parts of our pipeline. Then, when we save our changes, Jenkins will fill in the necessary syntax to incorporate our changes into a Declarative Pipeline in a Jenkinsfile. That Jenkinsfile can then be committed and pushed back into our project’s repository.



Let’s take a quick tour of the elements on the screen. The top row contains the standard Jenkins “primary” links that we talked about earlier in the chapter. In the next row is the name of our repository (not a link, by the way), a Cancel button to take us out of the editor, and a Save button to save the changes we make.

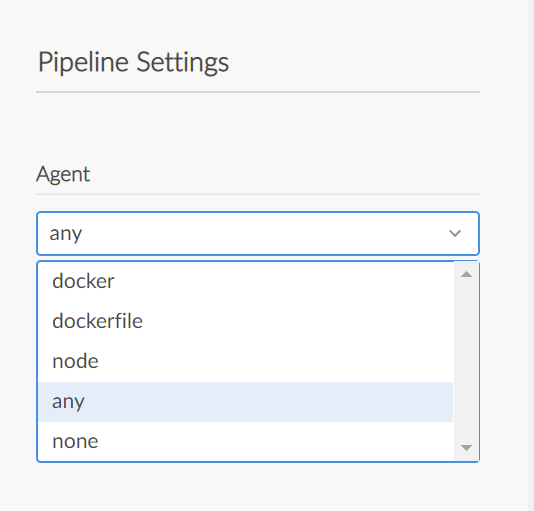
On the lefthand side of the main part of the screen, we have a Start halo with a line connecting it to an empty halo. The + in the second halo means we can click on that to add a new stage to our pipeline.

On the righthand side, of this main part of the screen, is the area where we can specify pipeline elements. Furthermore, we can choose—or type in—values for them to make up the parts of our pipeline.

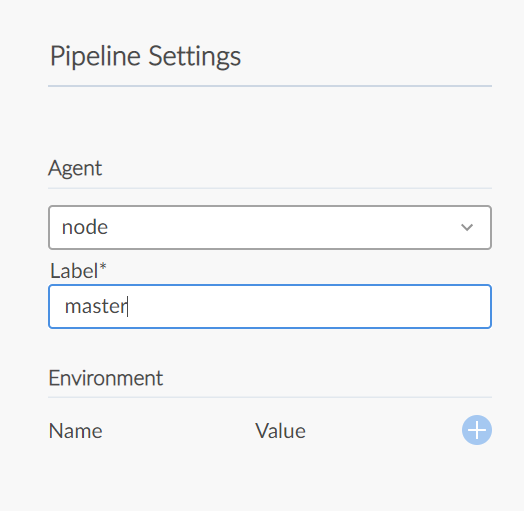
From here, some examples will best serve to demonstrate how to use the editor.

### **SPECIFYING GLOBAL PARTS OF THE PIPELINE**

We’re going to now create a pipeline using the editor. We’ll start out by specifying a particular agent (via the agent’s label) that we want to run the main part of the pipeline on. In the Pipeline Settings section, in the Agent field, we have a drop-down list that we can select the type of agent from



In this case, we’re going to use a standard node that we have available, so we select “node.” A new Label\* text box pops up for us to put the label in. In this case, we’ll use a node labelled master

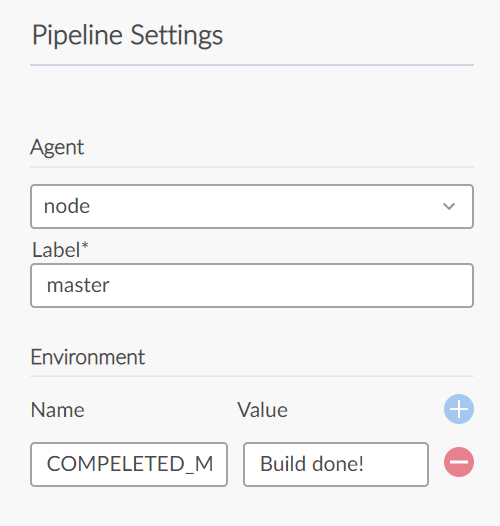


This would correspond to code like this in a Jenkinsfile:

pipeline {

   agent{label 'master'}

Under that, we have an option to add environment variables. For illustration purposes, we’ll set a variable named COMPLETED\_MSG to the value “Build done!” To do this, we simply click on the circle with the + sign to the right of the Name and Value labels. Text fields then pop up for us to enter the name and value for the environmentvariable, as shown in below pic



This then would make our pipeline code:

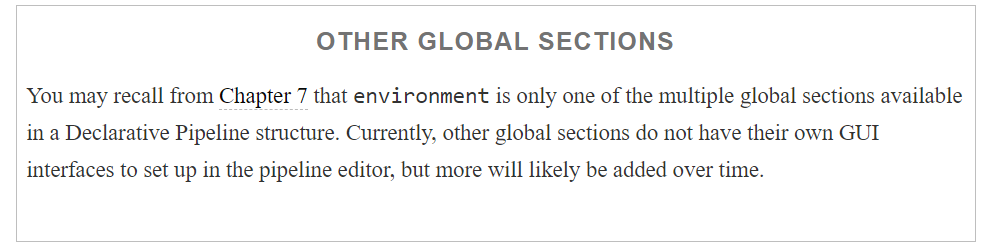
pipeline {

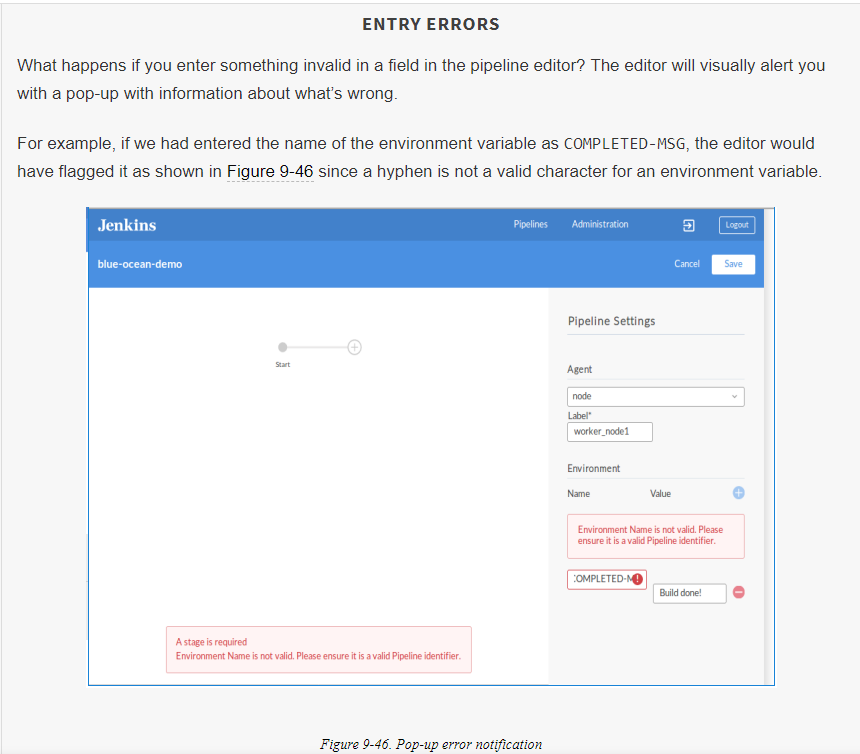
   agent{label 'master'}

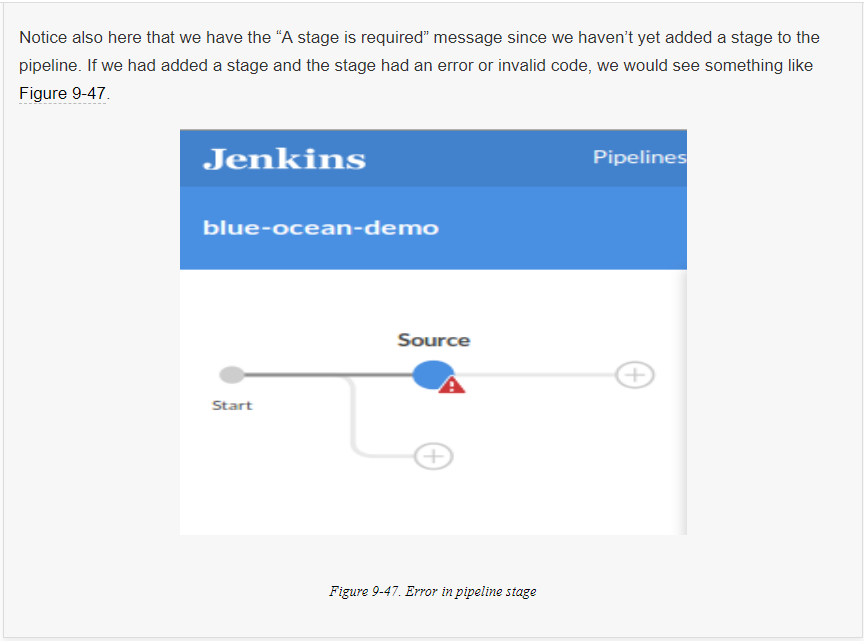
   environment {

      COMPLETED\_MSG = "Build done!"

  }

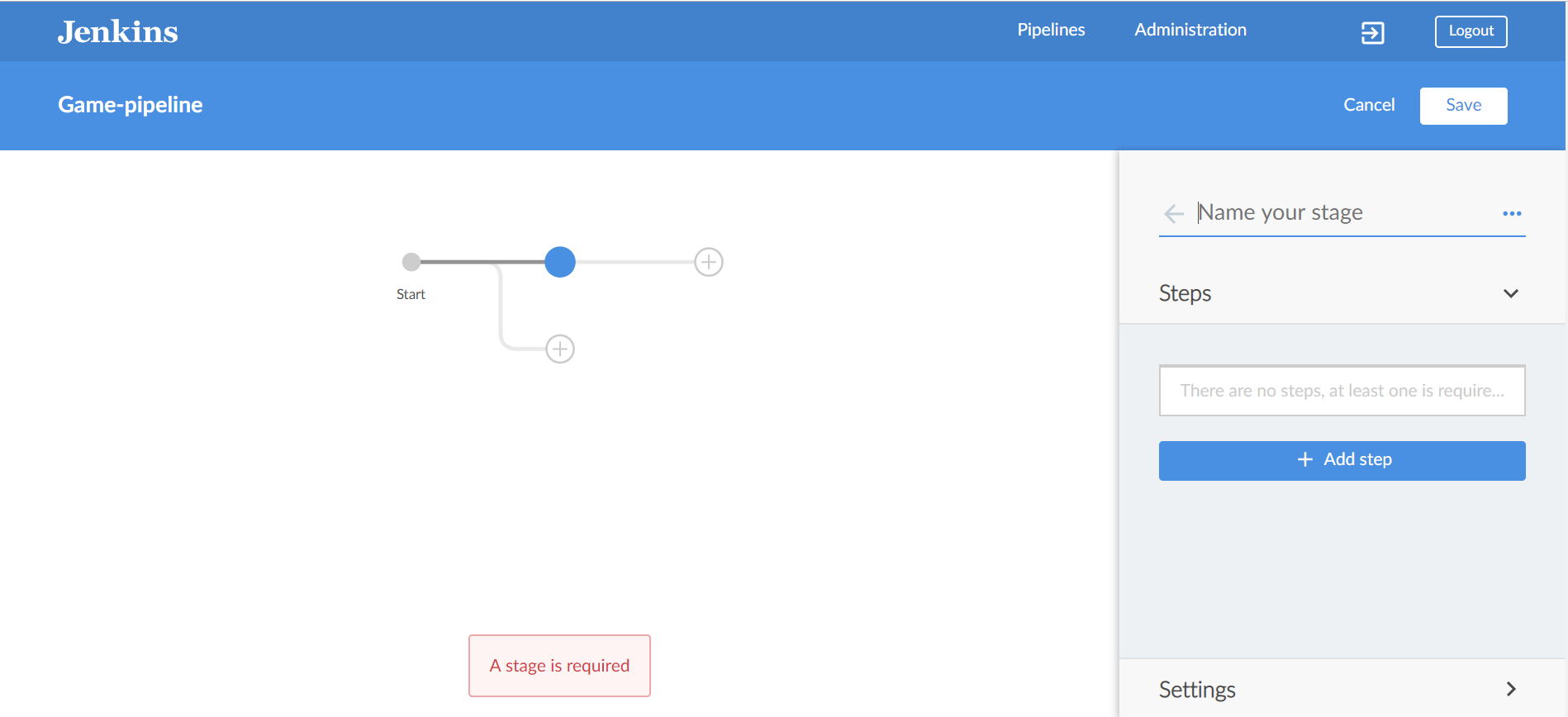






### **ADDING A NEW STAGE**

To add a new stage to our pipeline, we just need to click on the halo with the + inside it in the left section of the main screen. This results in the screen shown in



Let’s discuss what happened when we clicked on that halo.

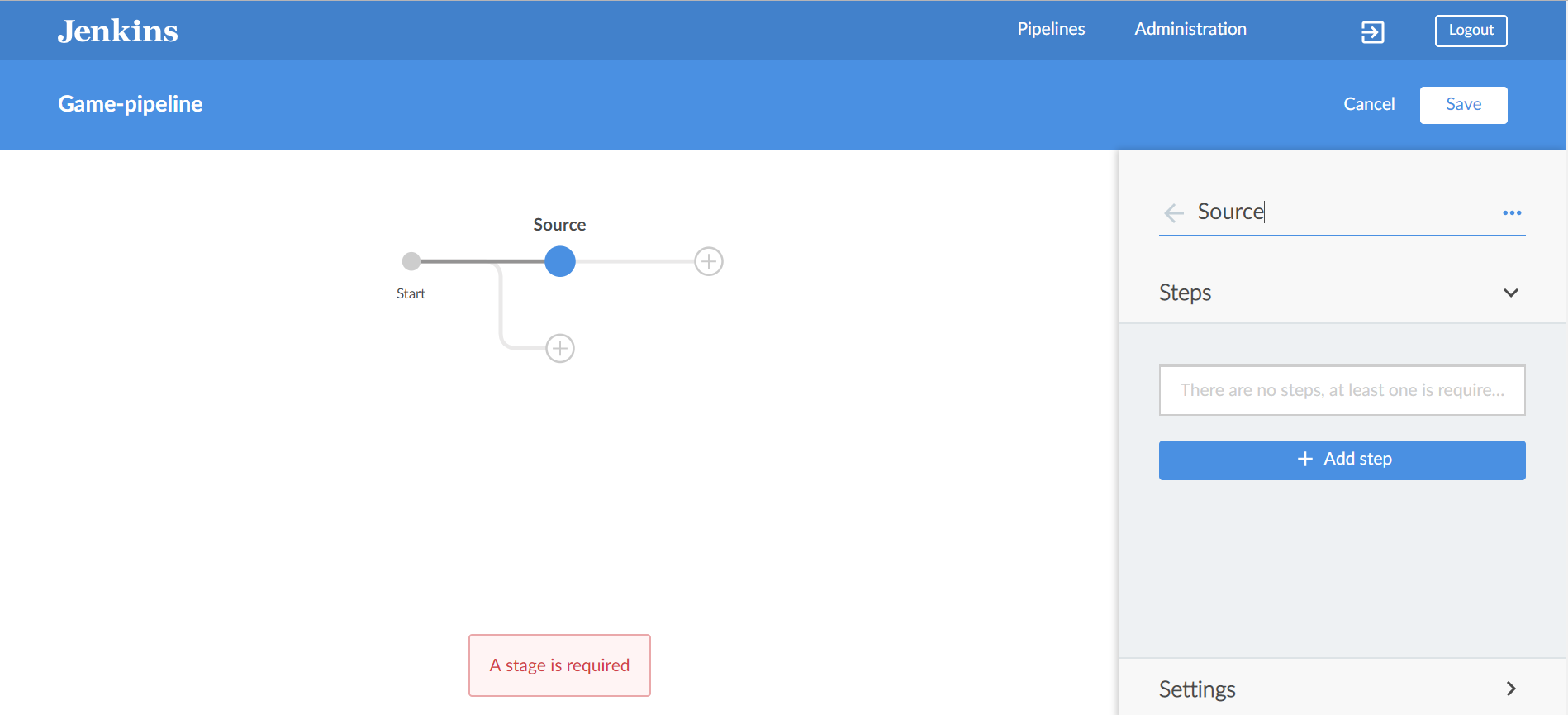
In the left part of the screen:

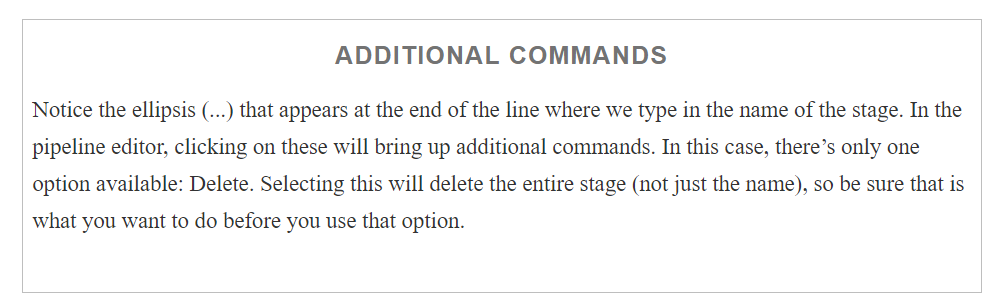
* Jenkins highlighted the halo we clicked on, by filling it in and turning it blue. That indicates it’s the currently selected stage that we’re editing.
* A new halo with a + was added *under* that one. This is a way to add steps to run in parallel if we need them.
* A new halo with a + was added *to the right* of our selected stage. That gives us a means to add another stage in our pipeline if needed.

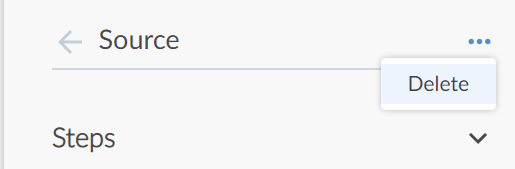
In the right part of the screen:

* A new entry area was set up for us to type in the name of the stage.
* A button was added so that we can add a step to the stage.

For our purposes here, we’ll go ahead and just define a simple stage to get the source for our project. So, we’ll type in “Source” for the name





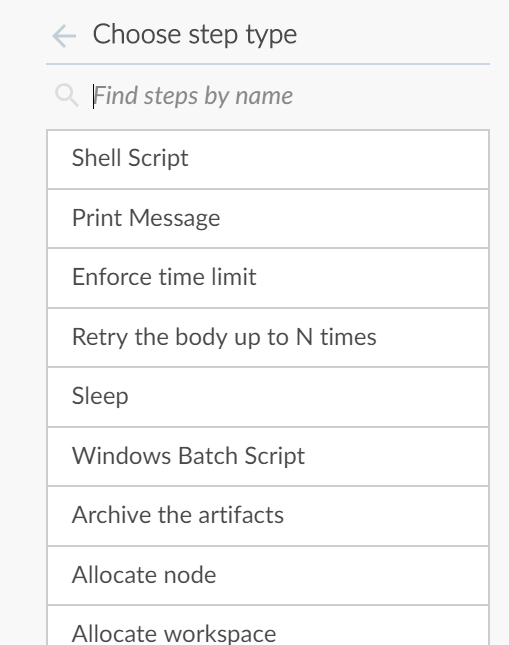


Now we’re ready to add one or more steps to our stage

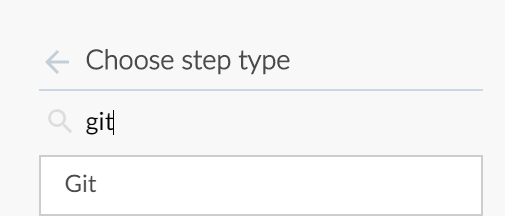
### **ADDING A STEP TO A STAGE**

Each stage in a Jenkins pipeline must have at least one step. If you attempt to move on from the stage definition without adding a step, the pipeline editor will display an error indication.

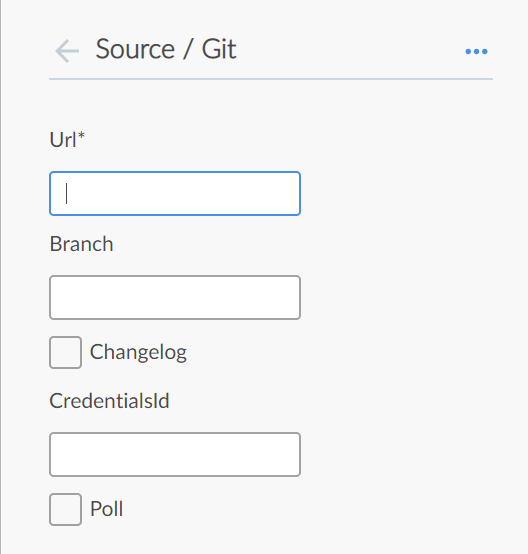
To add a new step to a stage, we just click the “+ Add step” button. Once we do that, the selection pane on the right turns into a list of available step types to pick from below pic.We can scroll through the list to find the step type we want, or we can type it into the search area.

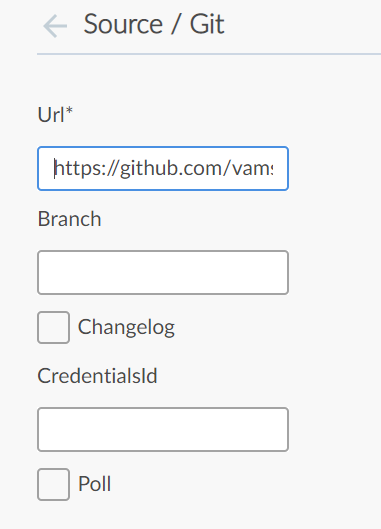


Here, we’ll use the GitSCM step for our Source stage to pull down our source code. In the search box for the steps, we can type “git” and quickly find the “Git” step



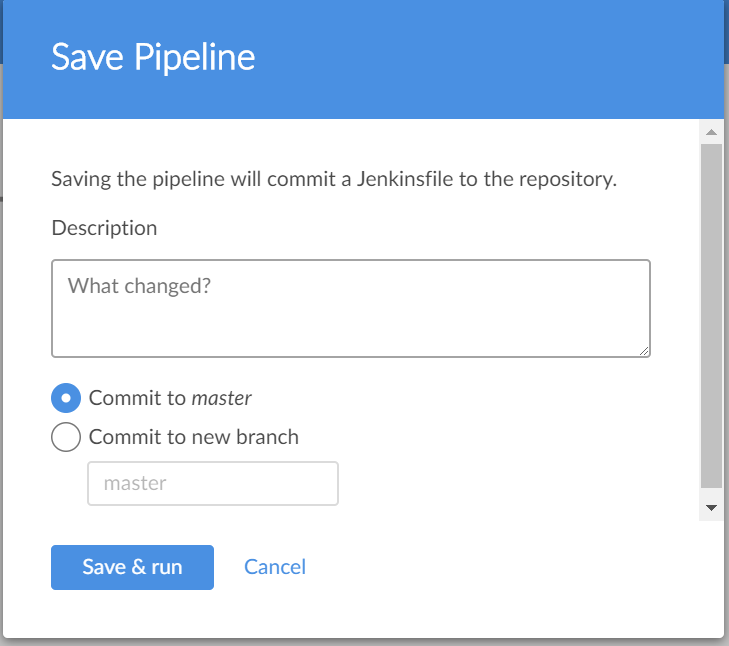
Selecting this step brings up a set of text fields that we can fill in to specify the parameters. Note the \* next to the Url field, meaning that it’s required. The main piece of information we need to put in is the path to our repository, as shown in below





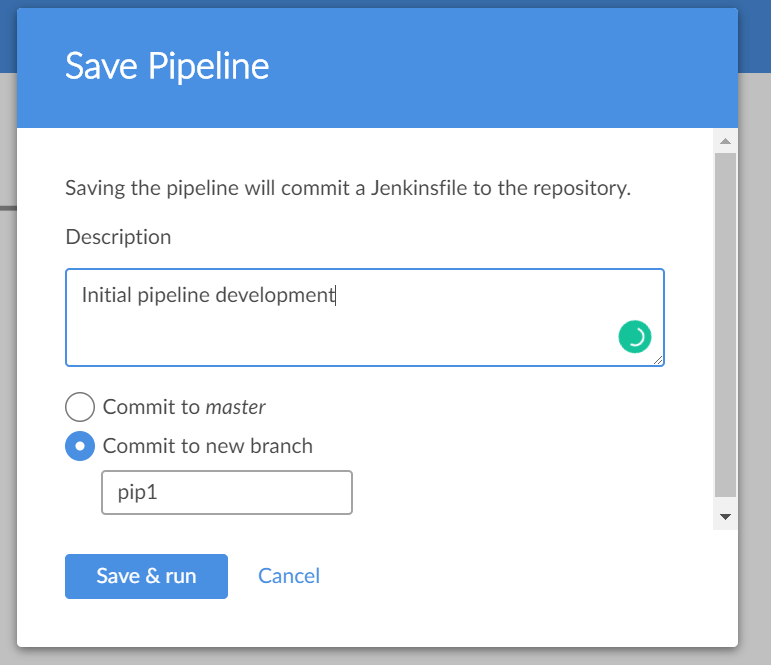
### **SAVING AND COMMITTING PIPELINE CHANGES**

Clicking the Save button in the pipeline editor brings up a dialog like the one shown in  below

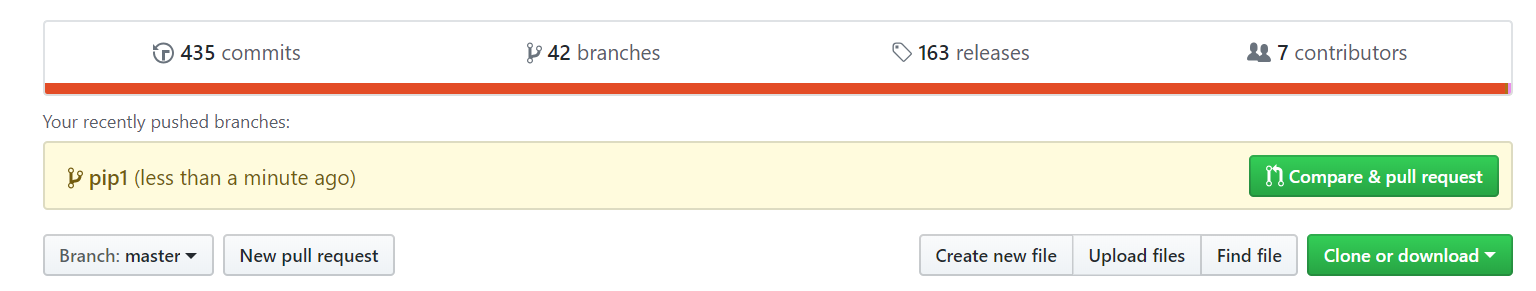


The fields here are self-explanatory. We’ll enter a simple description and choose to commit this to a new branch just because we’re still developing this pipeline. We can always merge it back into master later.

After filling in the fields, our save dialog looks like



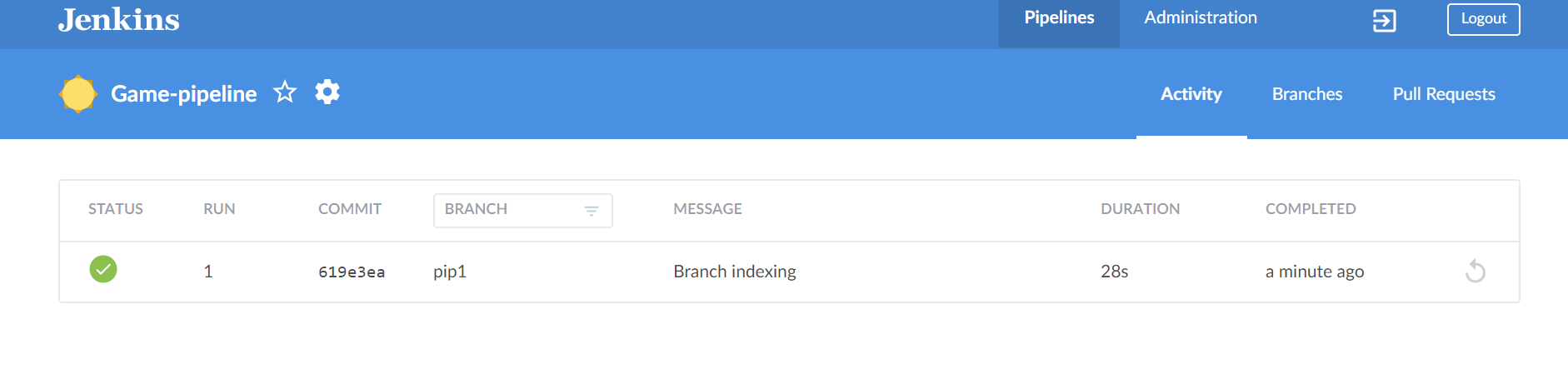
After we click the “Save & run” button, Jenkins will spin for a few moments while it updates the code and commits and pushes it over to GitHub. If we look in GitHub after the save, we can see the new branch there with the new Jenkinsfile



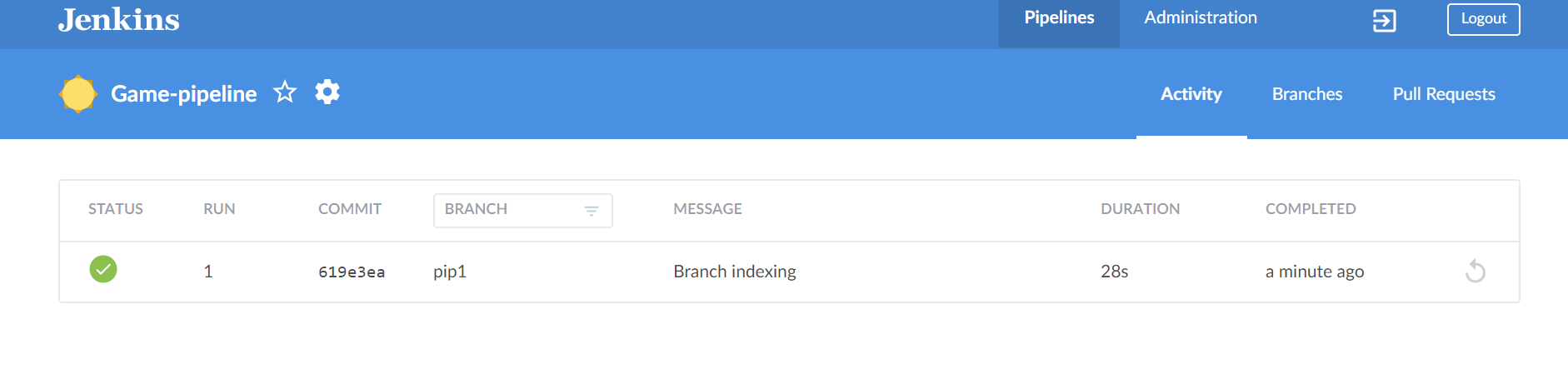
If we open up the Jenkinsfile on GitHub, we can see the code that was generated by our actions in the pipeline editor. This is shown in below



As soon as the code is updated, Jenkins will spin up a build for it. While it’s running, this looks like [Figure 9-57](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_build_in-progress). Note that the leftmost column features the halo being updated as the build progresses, and on the far right is the icon to stop the build if needed.



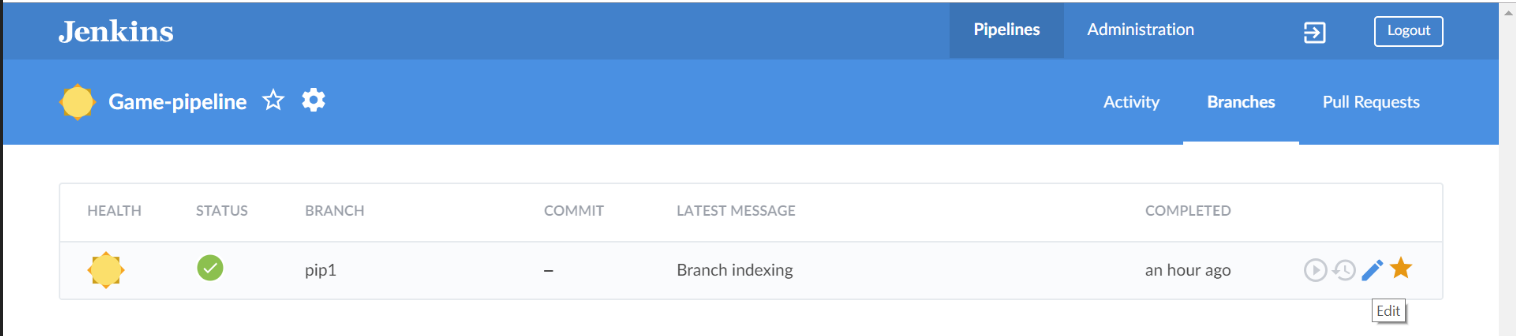
When the build is completed, the screen changes to look like [Figure 9-58](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_BO_completed_build). Notice the halo in the left column is now filled in, green, and checked to indicate successful completion. Also, the icon in the rightmost column has changed into a circular arrow, indicating it can be used to execute a rerun.



Our simple pipeline works, but it doesn’t do anything significant with its one stage. Let’s add another stage to do a build operation.

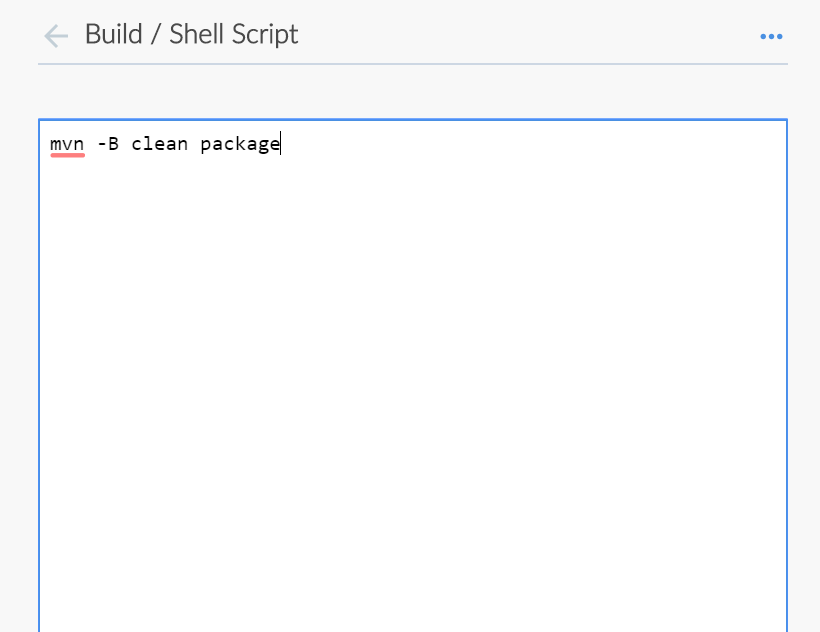
## **Editing an Existing Pipeline**

To edit an existing pipeline for a multibranch project, we switch to the Branches view and click on the next-to-last icon (the one that looks like a pencil) in the row for the branch we want to update. [Figure 9-59](https://www.safaribooksonline.com/library/view/jenkins-2-up/9781491979587/ch09.html#fig_sel_2edit_piplin_Bview) shows the Branches view for our project.

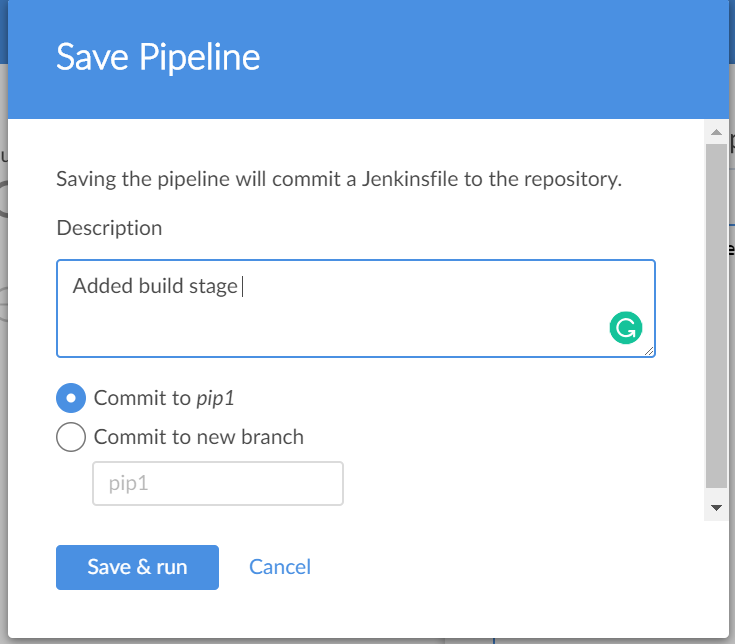


Once we click on the pipeline editor icon, we will be taken back to the pipeline editor screen. To add a new stage after (and not parallel to) the Source stage, we simply click the circle with the + sign in it that’s to the right of the Source stage. Then, on the righthand side of the screen, we’ll name this stage “Build”

For this step, we’re going to want to invoke our  Maven build instance as below

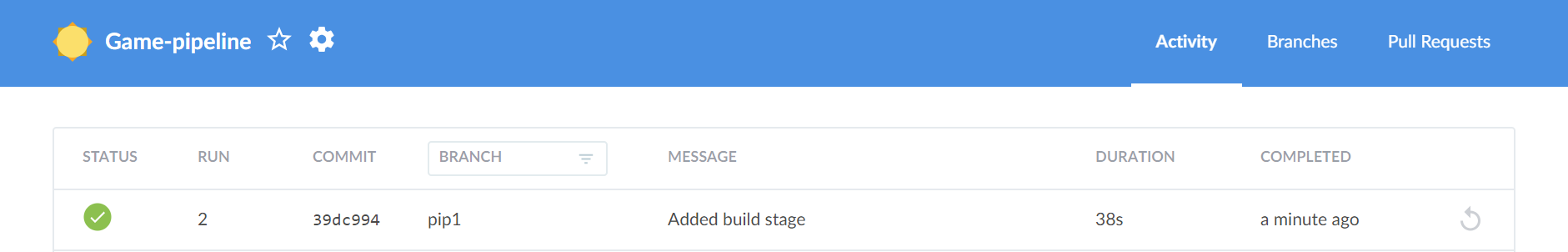


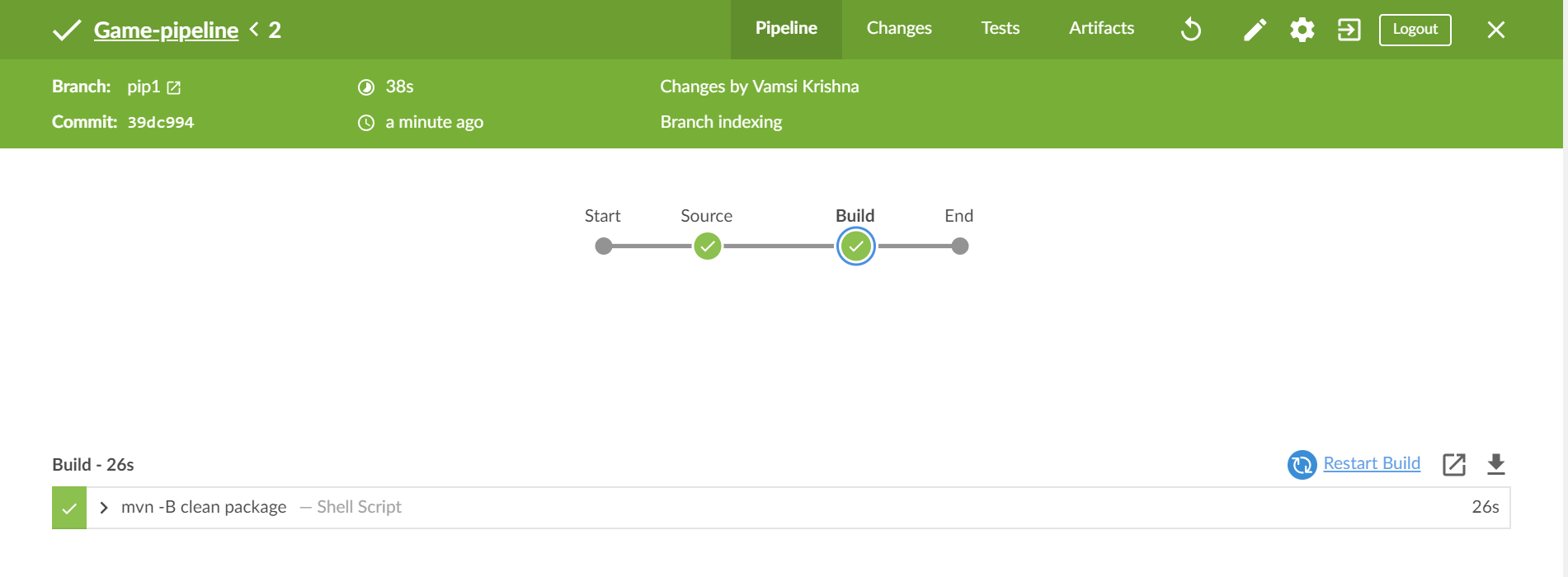
We can now simply save our pipeline and have it committed and pushed to the pipe1 branch again



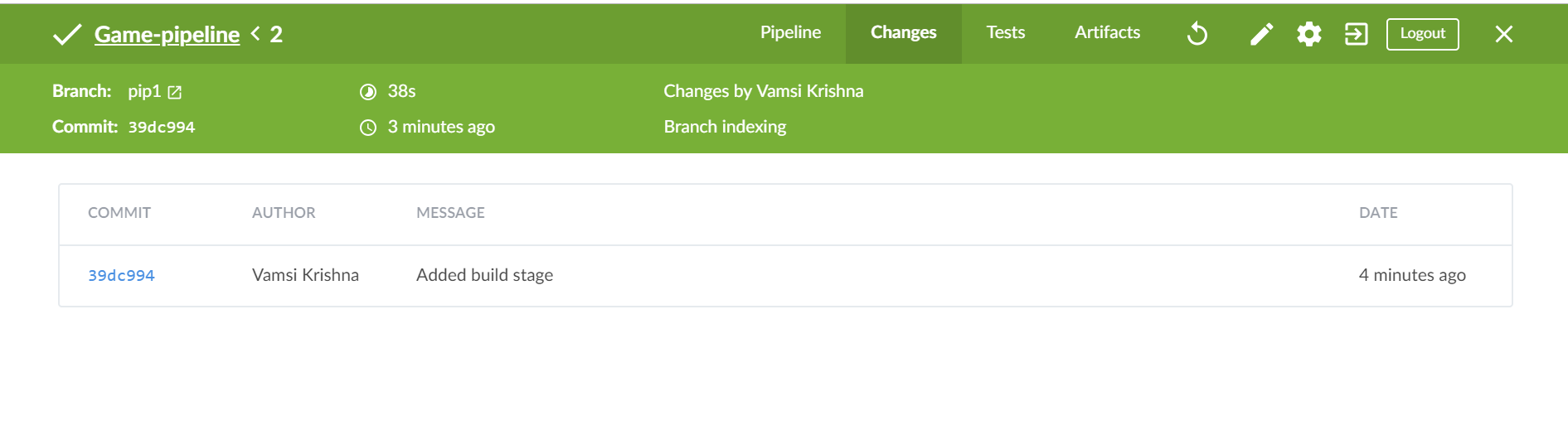
After committing the changes it would invoke the pipeline along with updated jenkinsfile







As we have committed the changes in order to update the new jenkinsfile we can see the commit info in the changes section in the job #n homepage



Which would redirect to the github page of the commit as below

