Experiment – Jenkins Build Pipelines

Experiment Overview:

Jenkins pipelines help you align the build process of a project. This is done by specifying tasks and the order in which they are executed. There are all kinds of possible tasks that a Jenkins pipeline can do for you. For example, build assets, send an email on error, send the build artifacts via SSH to your application server, etc.

Setup of Pipelines

Jenkins allows to specify pipelines using a `Jenkinsfile`. This is just a textfile that contains the necessary data for Jenkins to execute the pipeline. It is called `Jenkinsfile` (notice: no file extension) and should be placed in the root of your project.

> This file should be checked into version control as it is needed on your Jenkins instance.

Jenkins supports two different syntaxes.

1. Declarative (since Pipeline version 2.5)
2. Scripted

For this experiment we will focus on the declarative approach.

The following example shows a pipeline with 2 stages, this would go into a project file named “Jenkinsfile”. This is case sensitive and must be in this format. Place this in your GitHub, BitBucket or GitLab account. The example below would use GitHub, but any git environment will work.

```
pipeline {
    agent any

    stages {
        stage('Build Assets') {
            agent any
            steps {
                echo 'Building Assets...'
            }
        }
        stage('Test') {
            agent any
            steps {
                echo 'Testing stuff...'
```

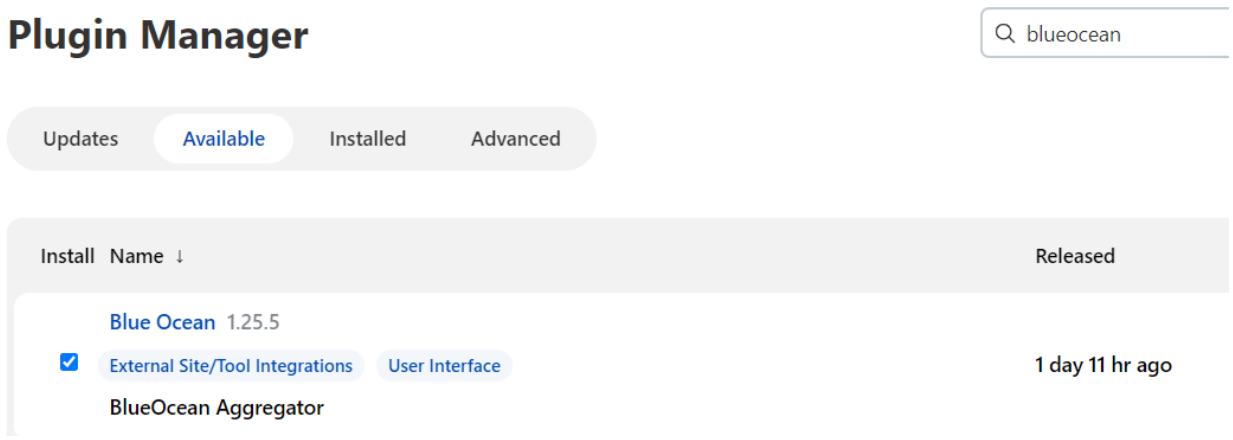
```
        }
      }
    }
}
```

The `agent` directive tells Jenkins to allocate a workspace and an executor for the pipeline. Without it, the pipeline is not valid and therefore required.

Setup using the Blue Ocean Plugin

Blue Ocean Plugin Installation

To install the Plugin we would go to **Manage Jenkins Manage Plugins Available** and select the *Blue Ocean* Plugin.



Select **Install without restart**

After the installation is finished you have an additional menu entry called `Open Blue Ocean` in your main *Jenkins* navigation.

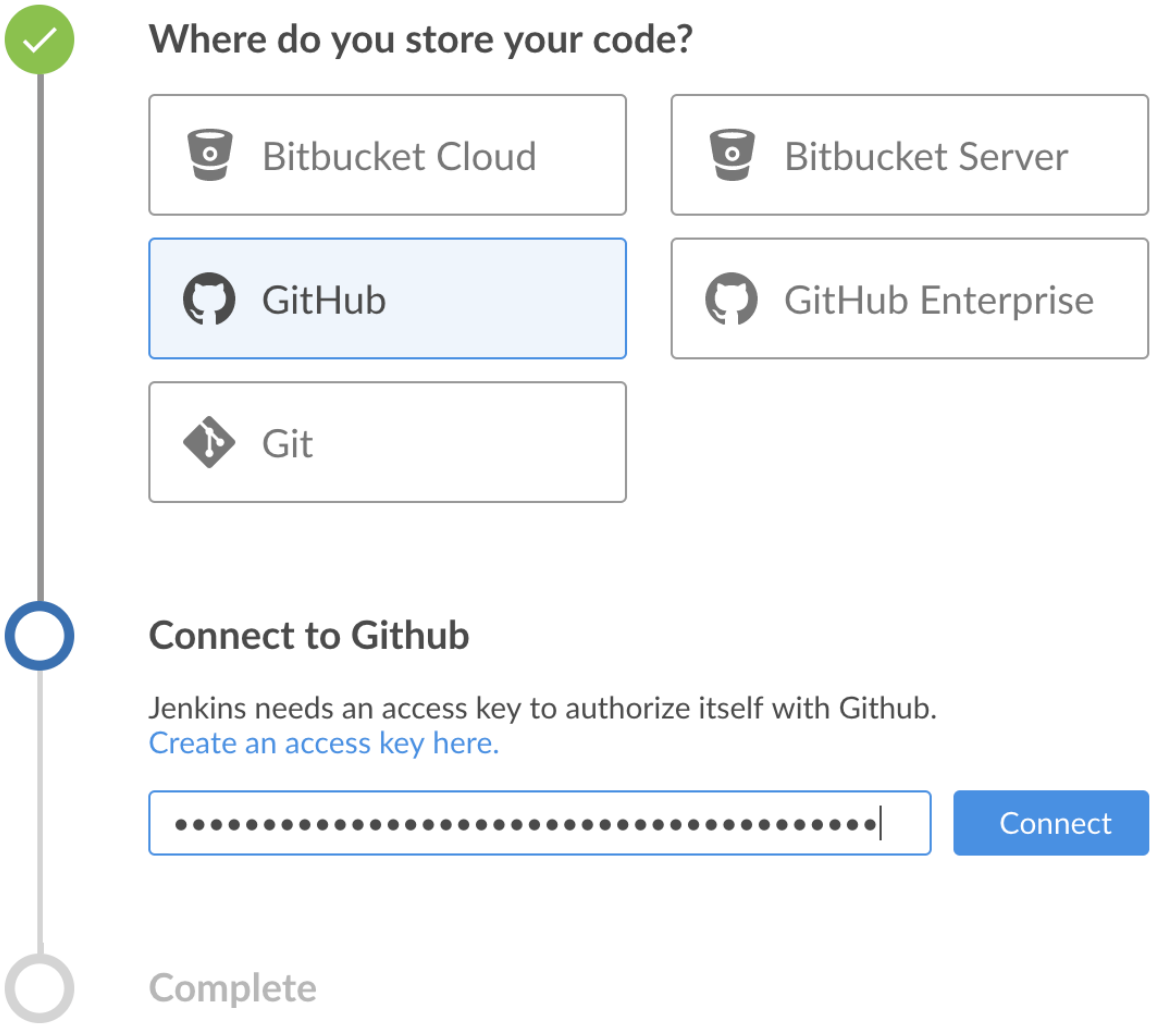
Fork the sample repository on GitHub

Fork the simple "Welcome to React" Node.js and React application on GitHub into your own GitHub account. https://github.com/jenkins-docs/creating-a-pipeline-in-blue-ocean.

1. Ensure you are signed in to your GitHub account. If you don't yet have a GitHub account, sign up for a free one on the GitHub website.
2. Fork the creating-a-pipeline-in-blue-ocean on GitHub into your local GitHub account. If you need help with this process, refer to the Fork A Repo documentation on the GitHub website for more information.

Create your Pipeline project in Blue Ocean

1. Go back to Jenkins and ensure you have accessed the Blue Ocean interface. To do this, make sure you:
    - have browsed to http://**JenkinsHost**:8080/blue and are logged in
      or
    - have browsed to http://**JenkinsHost**:8080/, are logged in and have clicked **Open Blue Ocean** on the left.
2. In the **Welcome to Jenkins** box at the center of the Blue Ocean interface, click **Create a new Pipeline** to begin the Pipeline creation wizard.
   Note: If you don't see this box, click **New Pipeline** at the top right.
3. In **Where do you store your code?**, click **GitHub**.
4. In **Connect to GitHub**, click **Create an access key here**. This opens GitHub in a new browser tab.
   Note: If you previously configured Blue Ocean to connect to GitHub using a personal access token, then Blue Ocean takes you directly to step 9 below.
5. In the new tab, sign in to your GitHub account (if necessary) and on the GitHub **New Personal Access Token** page, specify a brief **Token description** for your GitHub access token (e.g. Blue Ocean).
   Note: An access token is usually an alphanumeric string that respresents your GitHub account along with permissions to access various GitHub features and areas through your GitHub account. This access token will have the appropriate permissions pre-selected, which Blue Ocean requires to access and interact with your GitHub account.
6. Scroll down to the end of the page (leaving all other **Select scopes** options with their default settings) and click **Generate token**.
7. On the resulting **Personal access tokens** page, copy your newly generated access token.
8. Back in Blue Ocean, paste the access token into the **Your GitHub access token** field and click **Connect**.

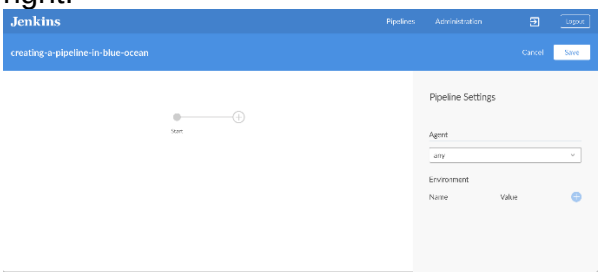


Jenkins now has access to your GitHub account (provided by your access token).

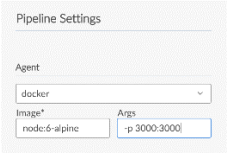
9. In **Which organization does the repository belong to?**, click your GitHub account (where you forked the repository above).
10. In **Choose a repository**, click your forked repository **creating-a-pipeline-in-blue-ocean**.
11. Click **Create Pipeline**.
    Blue Ocean detects that there is no Jenkinsfile at the root level of the repository's master branch and proceed to help you create one. (Therefore, you'll need to click another **Create Pipeline** at the end of the page to proceed.)
    Note: Under the hood, a Pipeline project created through Blue Ocean is actually "multibranch Pipeline". Therefore, Jenkins looks for the presence of at least one Jenkinsfile in any branch of your repository.

Create your initial Pipeline

1. Following on from creating your Pipeline project (above), in the Pipeline editor, select **docker** from the **Agent** dropdown in the **Pipeline Settings** panel on the right.



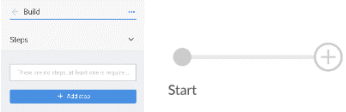
2. In the **Image** and **Args** fields that appear, specify node:6-alpine and -p 3000:3000 respectively.



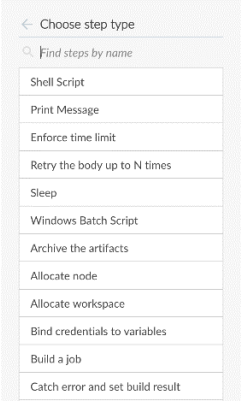
Note: For an explanation of these values, refer to annotations **1** and **2** of the Declarative Pipeline in the ``Create your initial Pipeline…" section of the Build a Node.js and React app tutorial.

3. Back in the main Pipeline editor, click the **+** icon, which opens the new stage panel on the right.

4. In this panel, type Build in the **Name your stage** field and then click the **Add Step** button below, which opens the **Choose step type** panel.

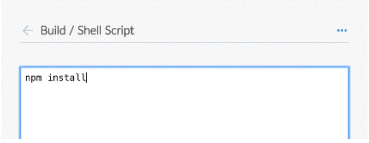


5. In this panel, click **Shell Script** near the top of the list (to choose that step type), which opens the **Build / Shell Script** panel, where you can enter this step's values.



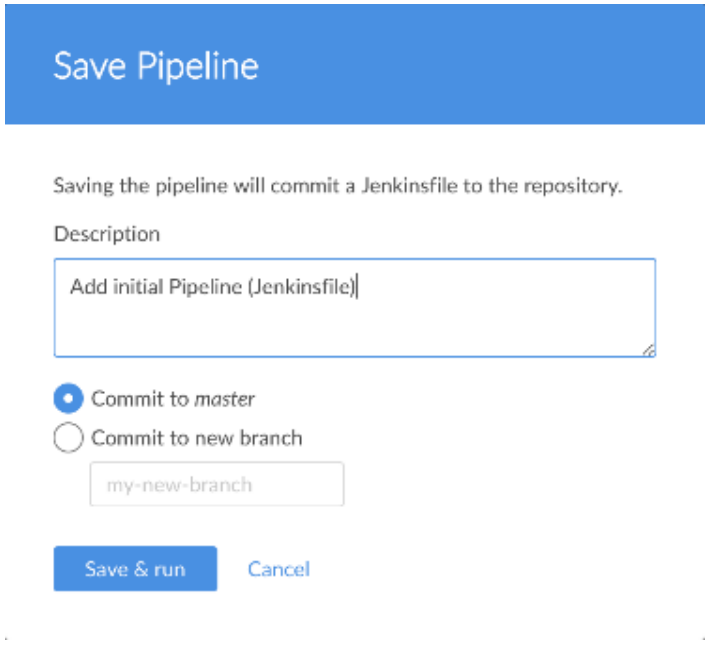
Tip: The most commonly used step types appear closest to the top of this list. To find other steps further down this list, you can filter this list using the **Find steps by name** option.

6. In the **Build / Shell Script** panel, specify npm install.

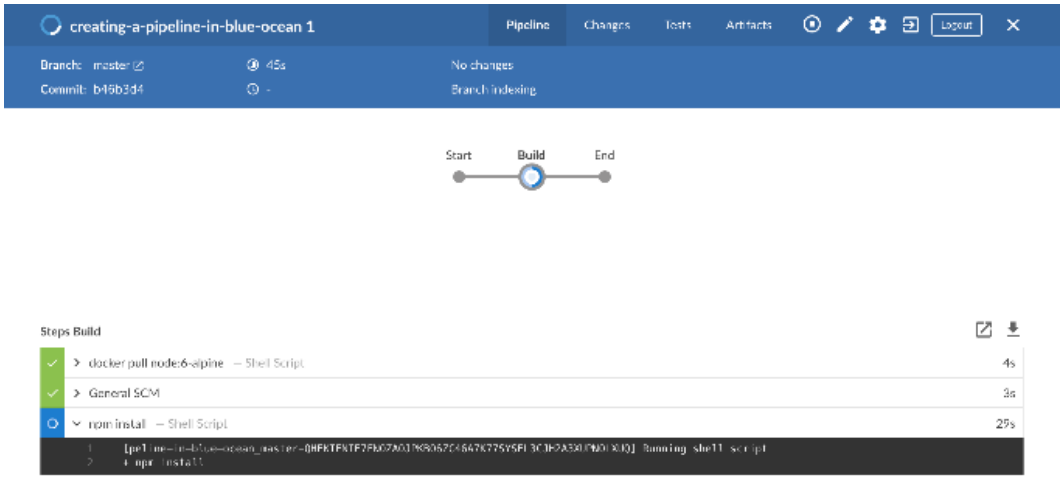


Note: For an explanation of this step, refer to annotation **4** of the Declarative Pipeline in the Create your initial Pipeline… section of the Build a Node.js and React app tutorial.

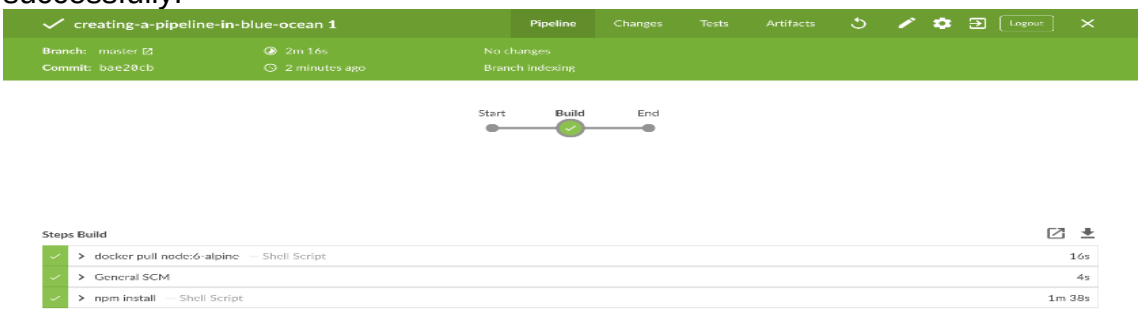
7. (*Optional*) Click the top-left back arrow icon to return to the main Pipeline editor.
8. Click the **Save** button at the top right to begin saving your new Pipeline with its "Build" stage.
9. In the **Save Pipeline** dialog box, specify the commit message in the **Description** field (e.g. Add initial Pipeline (Jenkinsfile)).



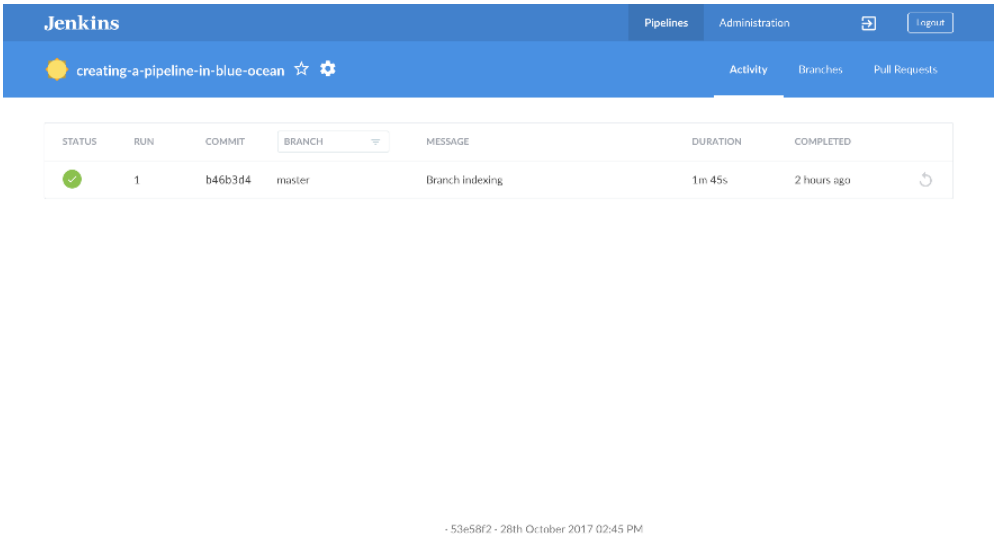
10. Leaving all other options as is, click **Save & run** and Jenkins proceeds to build your Pipeline.
11. When the main Blue Ocean interface appears, click the row to see Jenkins build your Pipeline project.
    Note: You may need to wait several minutes for this first run to complete. During this time, Jenkins does the following:
    . Commits your Pipeline as a Jenkinsfile to the only branch (i.e. master) of your repository.
    a. Initially queues the project to be built on the agent.
    b. Downloads the Node Docker image and runs it in a container on Docker.
    c. Executes the Build stage (defined in the Jenkinsfile) on the Node container. (During this time, npm downloads many dependencies necessary to run your Node.js and React application, which will ultimately be stored in the local node_modules directory within the Jenkins home directory).



12. The Blue Ocean interface turns green if Jenkins built your application successfully.



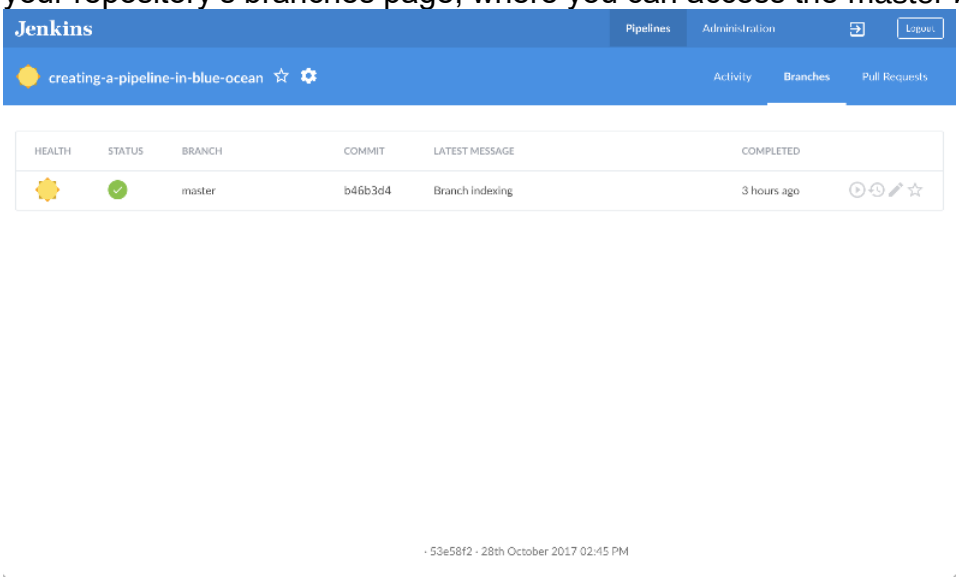
13.
14. Click the **X** at the top-right to return to the main Blue Ocean interface.



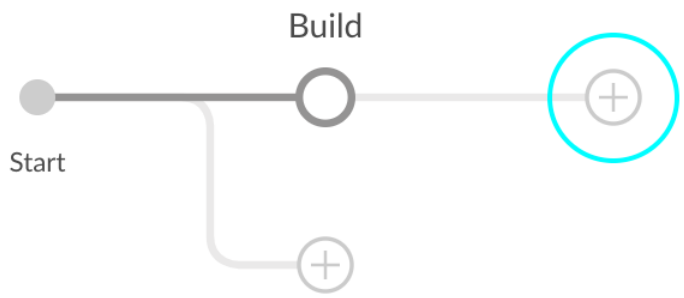
Note: Before continuing on, you can check that Jenkins has created a Jenkinsfile for you at the root of your forked GitHub repository (in the repository's sole master branch).

Add a test stage to your Pipeline

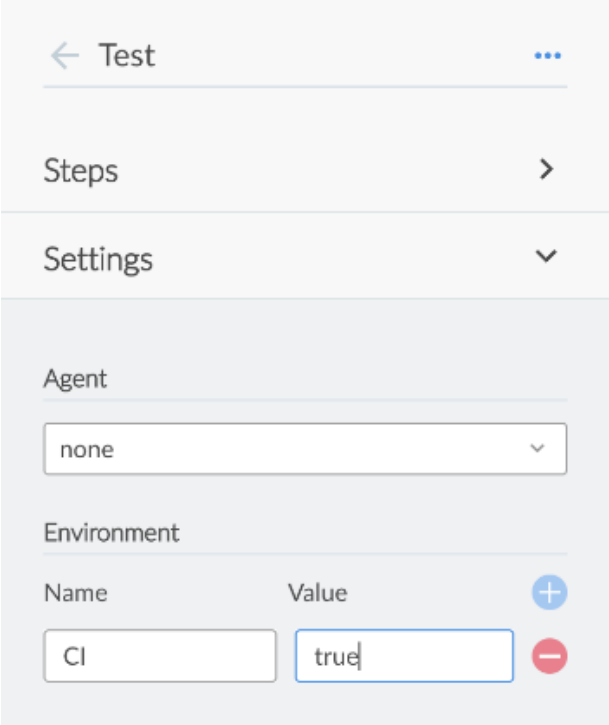
1. From the main Blue Ocean interface, click **Branches** at the top-right to access your repository's branches page, where you can access the master branch.



2. Click the master branch's "Edit Pipeline" icon to open the Pipeline editor for this branch.
3. In the main Pipeline editor, click the **+** icon to the right of the **Build** stage you created <u>above</u> to open the new stage panel on the right.

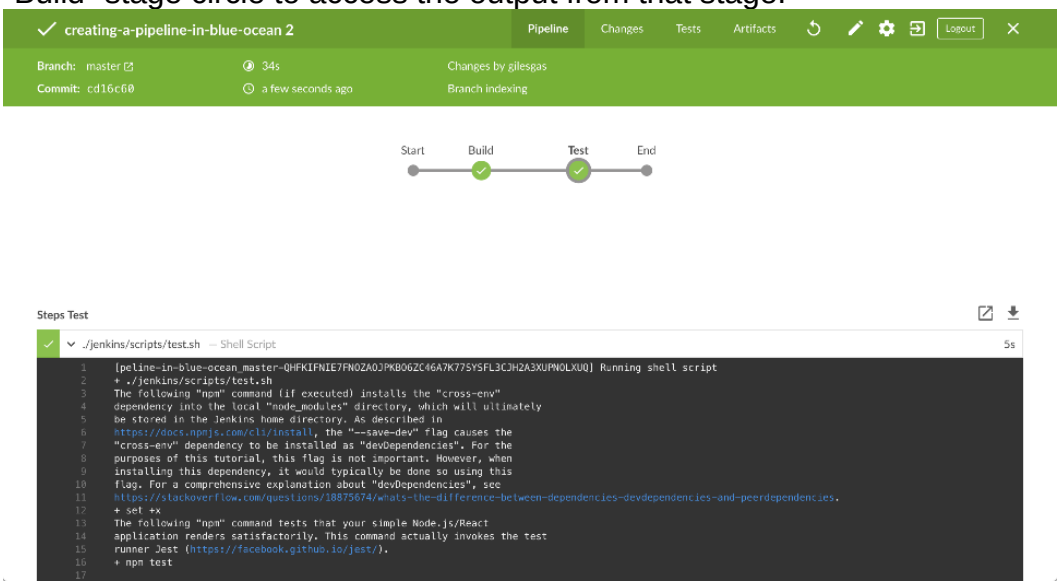


4. In this panel, type Test in the **Name your stage** field and then click the **Add Step** button below to open the **Choose step type** panel.
5. In this panel, click **Shell Script** near the top of the list.
6. In the resulting **Test / Shell Script** panel, specify ./jenkins/scripts/test.sh and then click the top-left back arrow icon to return to the Pipeline stage editor.
7. At the lower-right of the panel, click **Settings** to reveal this section of the panel.
8. Click the **+** icon at the right of the **Environment** heading (for which you'll configure an environment directive).
9. In the **Name** and **Value** fields that appear, specify CI and true, respectively.



Note: For an explanation of this directive and its step, refer to annotations **1** and **3** of the Declarative Pipeline in the Add a test stage… section of the Build a Node.js and React app tutorial.

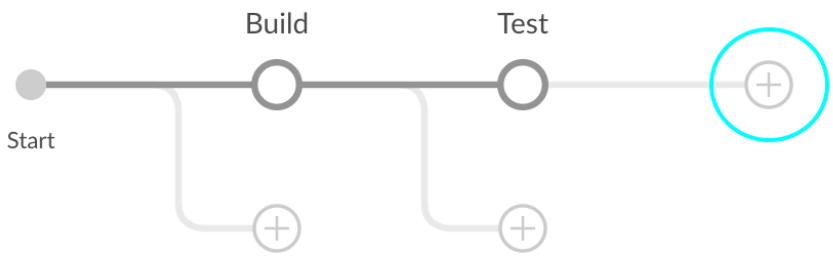
10. (*Optional*) Click the top-left back arrow icon to return to the main Pipeline editor.
11. Click the **Save** button at the top right to begin saving your Pipeline with with its new "Test" stage.
12. In the **Save Pipeline** dialog box, specify the commit message in the **Description** field (e.g. Add 'Test' stage).
13. Leaving all other options as is, click **Save & run** and Jenkins proceeds to build your amended Pipeline.
14. When the main Blue Ocean interface appears, click the *top* row to see Jenkins build your Pipeline project.
    Note: You'll notice from this run that Jenkins no longer needs to download the Node Docker image. Instead, Jenkins only needs to run a new container from the Node image downloaded previously. Therefore, running your Pipeline this subsequent time should be much faster.
    If your amended Pipeline ran successfully, here's what the Blue Ocean interface should look like. Notice the additional "Test" stage. You can click on the previous "Build" stage circle to access the output from that stage.



15. Click the **X** at the top-right to return to the main Blue Ocean interface.

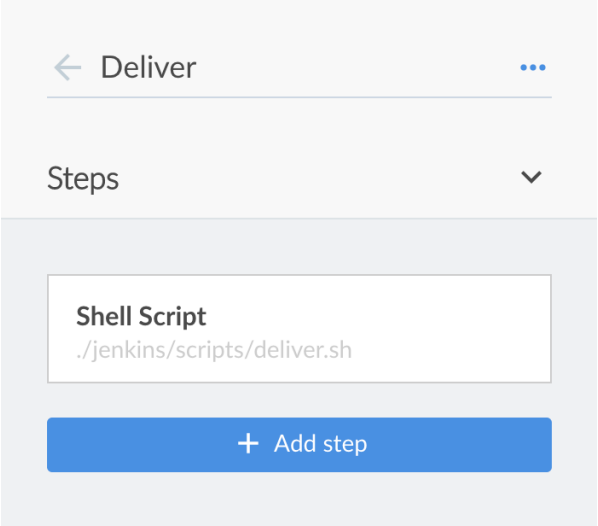
Add a final deliver stage to your Pipeline

1. From the main Blue Ocean interface, click **Branches** at the top-right to access your repository's master branch.
2. Click the master branch's "Edit Pipeline" icon to open the Pipeline editor for this branch.
3. In the main Pipeline editor, click the **+** icon to the right of the **Test** stage you created <u>above</u> to open the new stage panel.



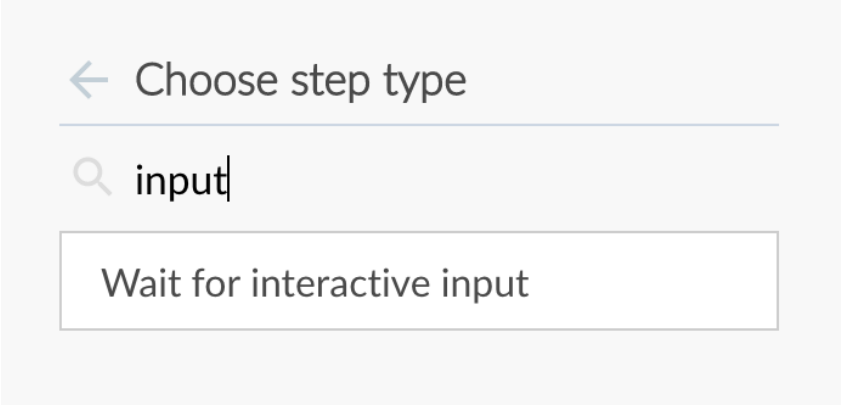
4. In this panel, type Deliver in the **Name your stage** field and then click the **Add Step** button below to open the **Choose step type** panel.
5. In this panel, click **Shell Script** near the top of the list.
6. In the resulting **Deliver / Shell Script** panel,

   specify ./jenkins/scripts/deliver.sh and then click the top-left back arrow icon to return to the Pipeline stage editor.

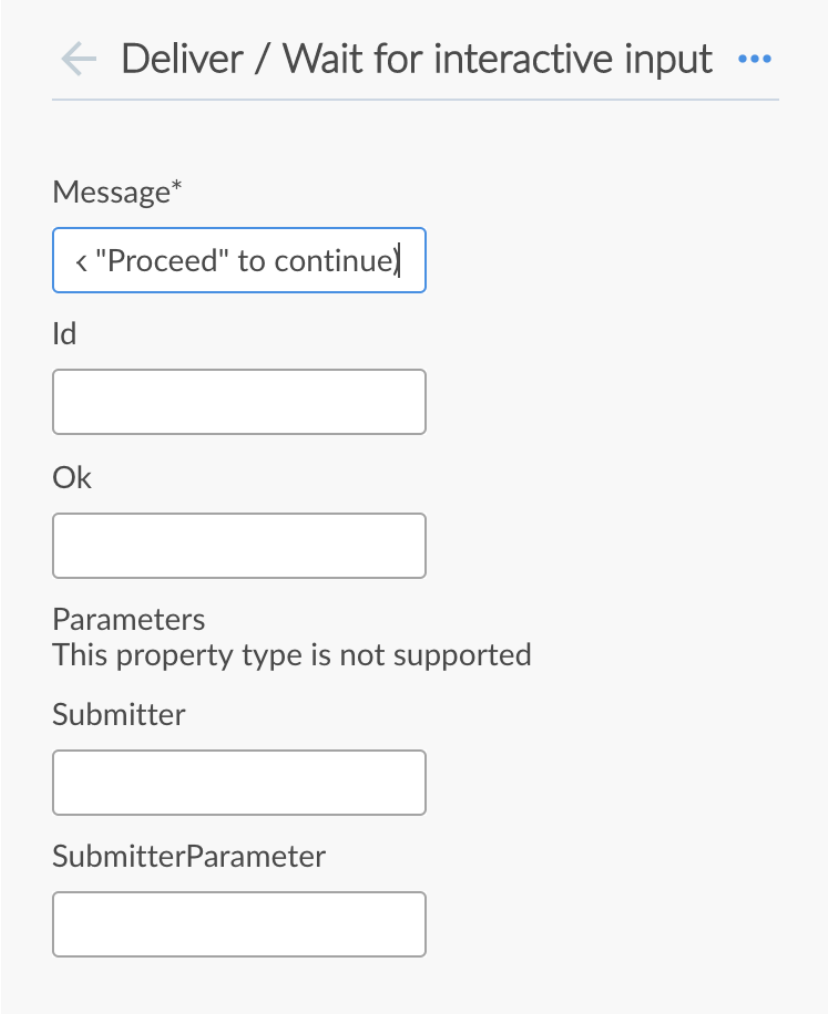


   Note: For an explanation of this step, refer to the deliver.sh file itself located in the jenkins/scripts of your forked repository on GitHub.

7. Click the **Add Step** button again.
8. In the **Choose step type** panel, type input into the **Find steps by name** field.



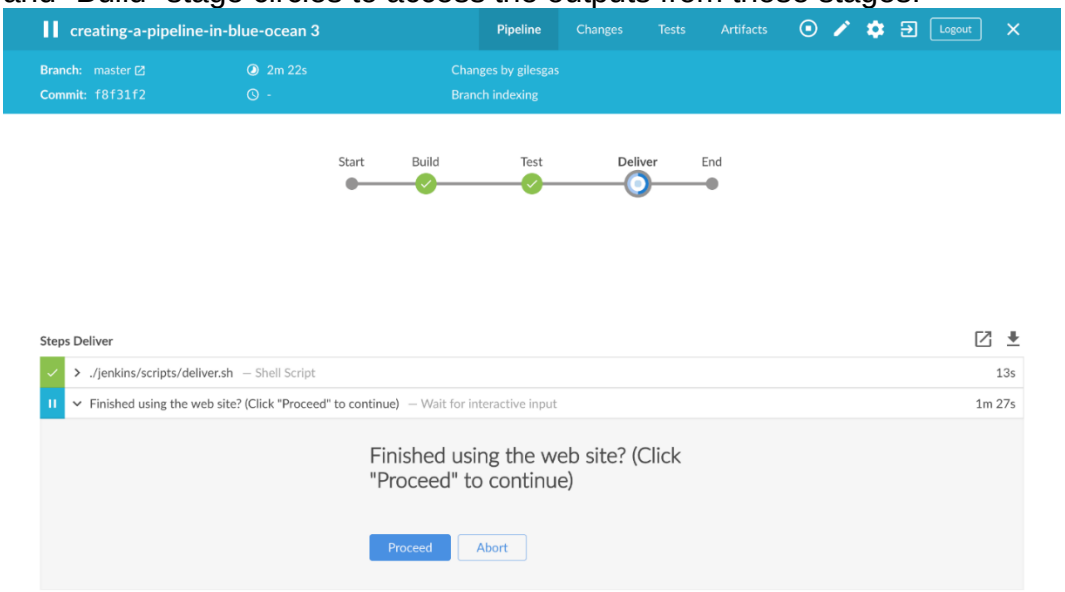
9. Click the filtered **Wait for interactive input** step type.
10. In the resulting **Deliver / Wait for interactive input** panel, specify Finished using the web site? (Click "Proceed" to continue) in the **Message** field and then click the top-left back arrow icon to return to the Pipeline stage editor.



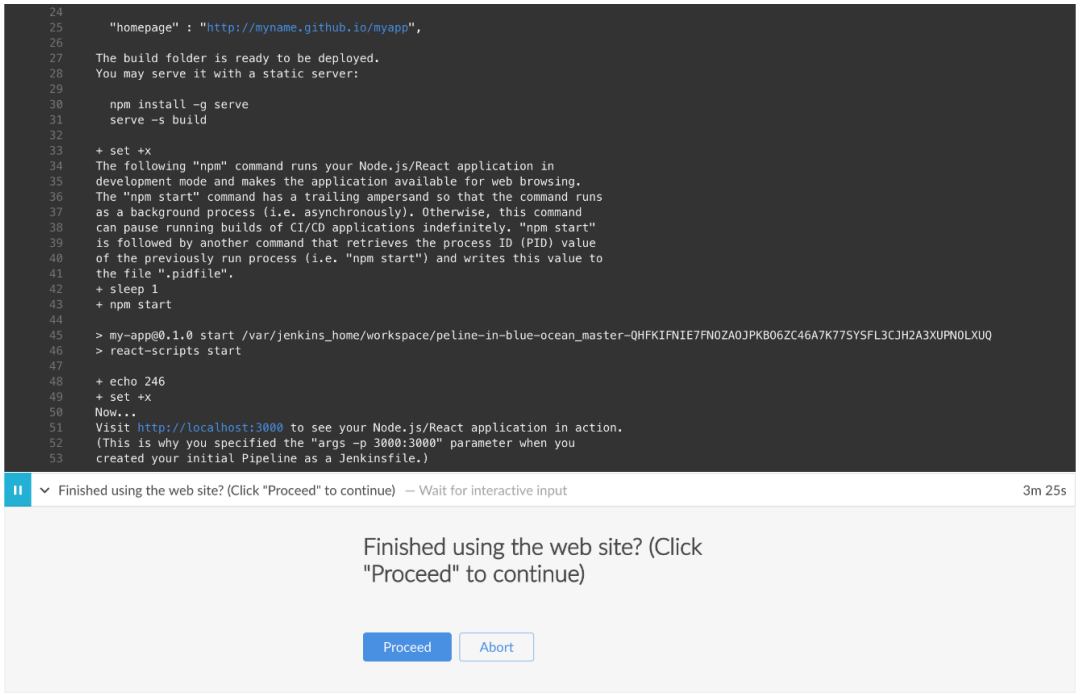
Note: For an explanation of this step, refer to annotation **4** of the Declarative Pipeline in the Add a final deliver stage… section of the Build a Node.js and React app tutorial.

11. Click the **Add Step** button (last time).
12. Click **Shell Script** near the top of the list.
13. In the resulting **Deliver / Shell Script** panel, specify ./jenkins/scripts/kill.sh.
    Note: For an explanation of this step, refer to the kill.sh file itself located in the jenkins/scripts of your forked repository on GitHub.
14. (*Optional*) Click the top-left back arrow icon to return to the main Pipeline editor.
15. Click the **Save** button at the top right to begin saving your Pipeline with with its new "Deliver" stage.

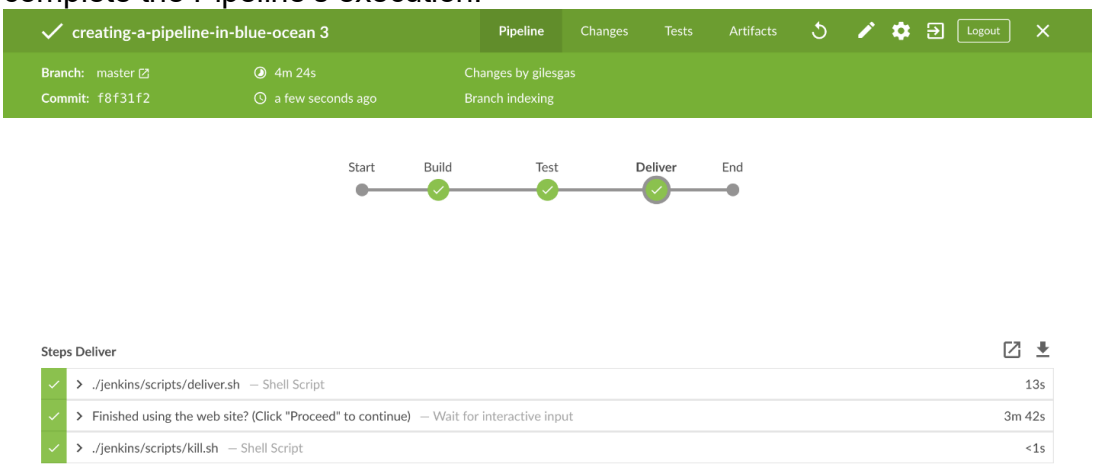
16. In the **Save Pipeline** dialog box, specify the commit message in the **Description** field (e.g. Add 'Deliver' stage).
17. Leaving all other options as is, click **Save & run** and Jenkins proceeds to build your amended Pipeline.
18. When the main Blue Ocean interface appears, click the *top* row to see Jenkins build your Pipeline project.
    If your amended Pipeline ran successfully, here's what the Blue Ocean interface should look like. Notice the additional "Deliver" stage. Click on the previous "Test" and "Build" stage circles to access the outputs from those stages.



19. Ensure you are viewing the "Deliver" stage (click it if necessary), then click the green **./jenkins/scripts/deliver.sh** step to expand its content and scroll down until you see the http://localhost:3000 link.



20. Click the http://localhost:3000 link to view your Node.js and React application running (in development mode) in a new web browser tab. You should see a page/site with the title **Welcome to React** on it.
21. When you are finished viewing the page/site, click the **Proceed** button to complete the Pipeline's execution.



22. Click the **X** at the top-right to return to the main Blue Ocean interface, which lists your previous Pipeline runs in reverse chronological order.

Jenkins

Pipelines | Administration | Logout

creating-a-pipeline-in-blue-ocean

Activity | Branches | Pull Requests

STATUS	RUN	COMMIT	BRANCH	MESSAGE	DURATION	COMPLETED
✓	3	f8f31f2	master	Add 'Deliver' stage	4m 24s	a few seconds...
✓	2	849f4ba	master	Add 'Test' stage	28s	8 minutes ago
✓	1	cb83102	master	Branch indexing	2m 5s	10 minutes ago

· 53e58f2 · 28th October 2017 02:45 PM