

# 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.

#### Plugin Manager

[Updates](#)[Available](#)[Installed](#)[Advanced](#)

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<b>Blue Ocean</b> 1.25.5 External Site/Tool Integrations User Interface BlueOcean Aggregator	1 day 11 hr ago

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 represents 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**.





### Where do you store your code?

 Bitbucket Cloud

 Bitbucket Server

 GitHub

 GitHub Enterprise

 Git

### Connect to Github

Jenkins needs an access key to authorize itself with Github.  
[Create an access key here.](#)

.....|

Connect

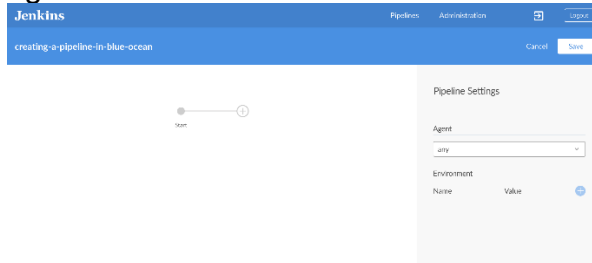
### Complete

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

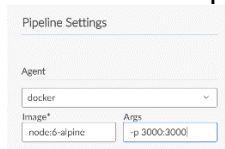
9. In **Which organization does the repository belong to?**, click your GitHub account (where you forked the repository above).
10. In **Choose a repository**, click your forked repository **creating-a-pipeline-in-blue-ocean**.
11. Click **Create Pipeline**.  
Blue Ocean detects that there is no Jenkinsfile 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.

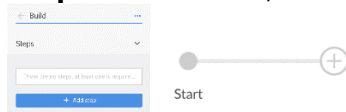


2. In the **Image** and **Args** fields that appear, specify node:6-alpine and -p 3000:3000 respectively.

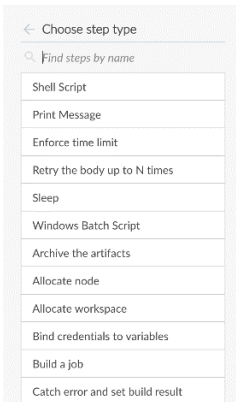


**Note:** For an explanation of these values, refer to annotations **1** and **2** of the Declarative Pipeline in the [`Create your initial Pipeline...`](#) section of the [Build a Node.js and React app](#) tutorial.

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.

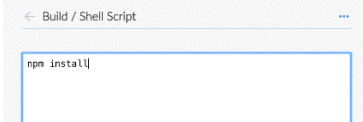


5. In this panel, click **Shell Script** near the top of the list (to choose that step type), which opens the **Build / Shell Script** panel, where you can enter this step's values.




**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)).

Save Pipeline

Saving the pipeline will commit a Jenkinsfile to the repository.

Description

Add initial Pipeline (Jenkinsfile)

☒ Commit to master  
☐ Commit to new branch

my-new-branch

Save & runCancel

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).





Jenkins						
<div> <div> Pipelines Administration </div> <div> creating-a-pipeline-in-blue-ocean ☆ ⚙️ </div> <div> Activity Branches Pull Requests </div> </div>						
STATUS	RUN	COMMIT	BRANCH	MESSAGE	DURATION	COMPLETED
✓	1	b46b3d4	master	Branch indexing	1m 45s	2 hours ago

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


**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).

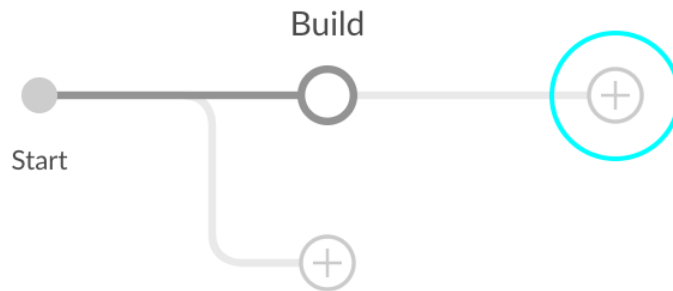
Add a test stage to your Pipeline


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.

Jenkins						
<div> <div> Pipelines Administration </div> <div> creating-a-pipeline-in-blue-ocean ☆ ⚙️ </div> <div> Activity Branches Pull Requests </div> </div>						
HEALTH	STATUS	BRANCH	COMMIT	LATEST MESSAGE	COMPLETED	
⚙️	✓	master	b46b3d4	Branch indexing	3 hours ago	🕒 ⚙️ ☆

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

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 above 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.

← Test
...

Steps
>

Settings
▼


Agent

none

Environment

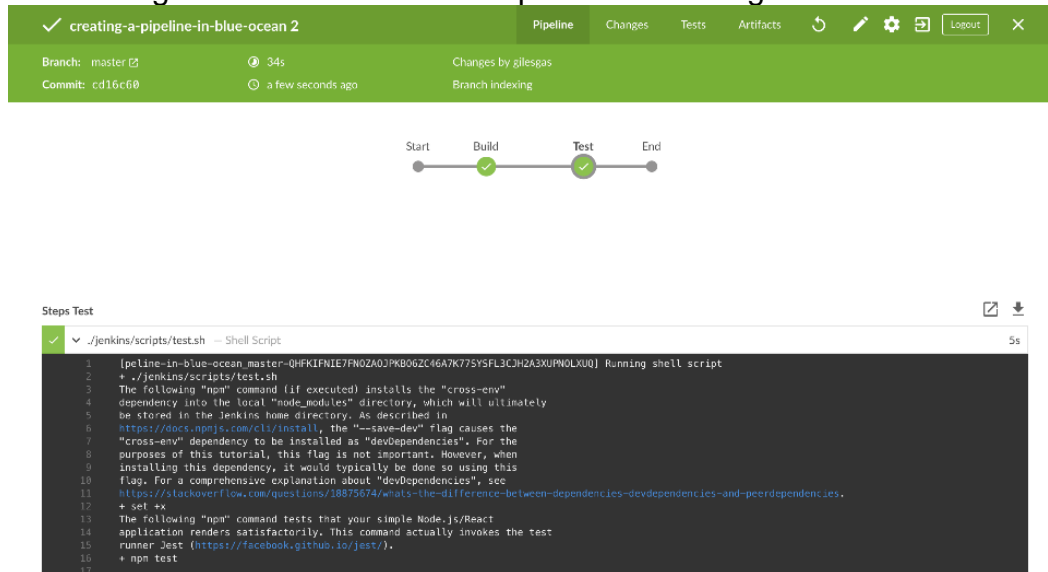
Name	Value	
CI	true	<div>+</div> <div>-</div>

**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 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.




The screenshot shows the Jenkins Blue Ocean interface for a pipeline named "creating-a-pipeline-in-blue-ocean 2". The pipeline status is "Success" (green checkmark). The pipeline details show it was built on the "master" branch, committed by "cd16c60", and took "34s" to complete. The pipeline graph shows two stages: "Build" and "Test", both of which are completed (green checkmarks). Below the graph, the "Steps Test" section shows the output of the "Test" stage, which includes a shell script that installs dependencies and runs tests.

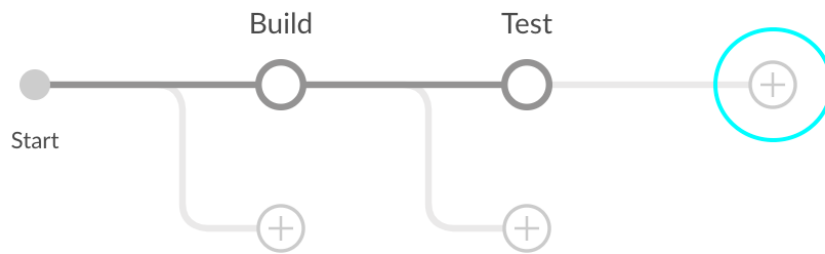
```

Steps Test
✓ ./jenkins/scripts/test.sh -- Shell Script
1 |pipeline-in-blue-ocean_master-QHFKIFNIE7FNOZA0JPKB06ZC46A7K77SY5FL3CJH2A3XUPNOLXUQ| Running shell script
2 + ./jenkins/scripts/test.sh
3 The following "npm" command (if executed) installs the "cross-env"
4 dependency into the local "node_modules" directory, which will ultimately
5 be stored in the Jenkins home directory. As described in
6 https://docs.npmjs.com/cli/install, the "--save-dev" flag causes the
7 "cross-env" dependency to be installed as "devDependencies". For the
8 purposes of this tutorial, this flag is not important. However, when
9 installing this dependency, it would typically be done so using this
10 flag. For a comprehensive explanation about "devDependencies", see
11 https://stackoverflow.com/questions/18875674/whats-the-difference-between-dependencies-devdependencies-and-peerdependencies.
12 + set +x
13 The following "npm" command tests that your simple Node.js/React
14 application renders satisfactorily. This command actually invokes the test
15 runner jest (https://facebook.github.io/jest/).
16 + npm test
17

```

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 above 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.



← Deliver

Steps

Shell Script
   
 ./jenkins/scripts/deliver.sh


+ Add step

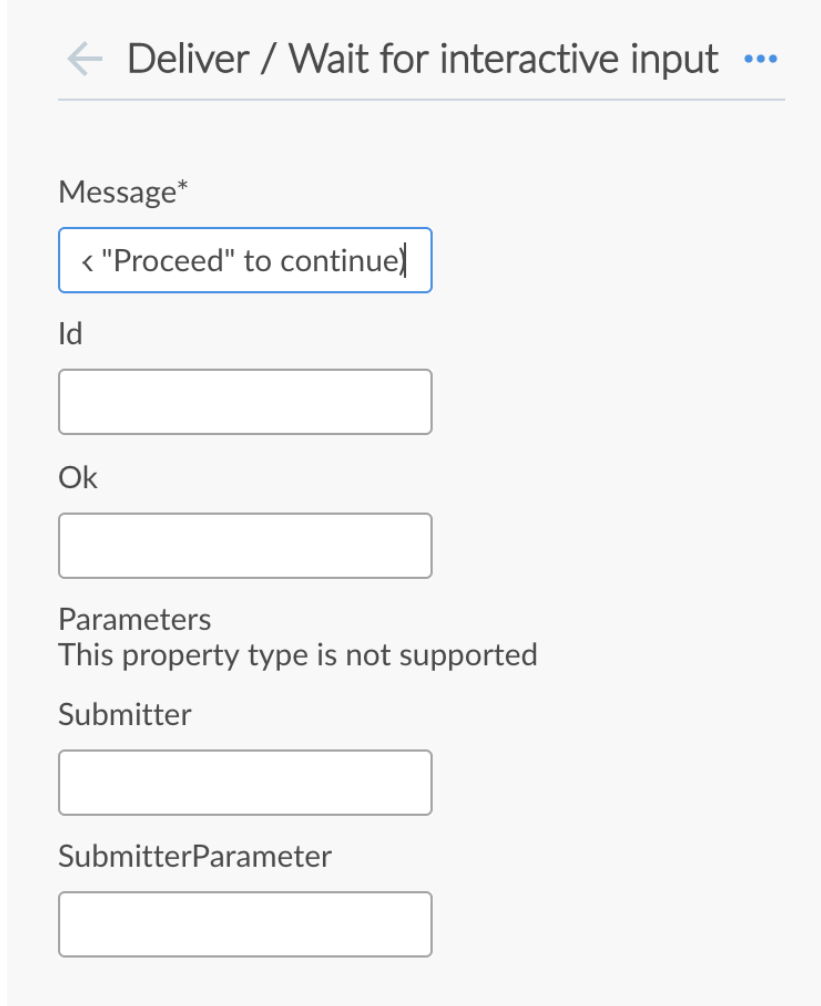
**Note:** For an explanation of this step, refer to the deliver.sh file itself located in the jenkins/scripts of your forked repository on GitHub.

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

← Choose step type

Wait for interactive input

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.



← Deliver / Wait for interactive input ...

Message\*

< "Proceed" to continue

Id

Ok


Parameters

This property type is not supported

Submitter

SubmitterParameter

**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 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.

The screenshot displays the Jenkins Blue Ocean interface for a pipeline named "creating-a-pipeline-in-blue-ocean 3". The top navigation bar includes tabs for Pipeline, Changes, Tests, and Artifacts, along with icons for play, edit, settings, and a Logout button. Below the navigation bar, a status bar shows the branch as "master" and the commit as "f8f31f2". The main area features a pipeline graph with stages: Start, Build, Test, Deliver, and End. The Build and Test stages are marked with green checkmarks, indicating successful completion. The Deliver stage is currently active, highlighted with a blue circle. Below the graph, the "Steps Deliver" section is expanded, showing two steps: a shell script step ".jenkins/scripts/deliver.sh" which completed in 13s, and a wait step "Finished using the web site? (Click 'Proceed' to continue)" which is currently running and has a duration of 1m 27s. The wait step's content area displays the text "Finished using the web site? (Click 'Proceed' to continue)" and two buttons: "Proceed" and "Abort".

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.

```
24 "homepage" : "http://myname.github.io/myapp",
25
26
27 The build folder is ready to be deployed.
28 You may serve it with a static server:
29
30 npm install -g serve
31 serve -s build
32
33 + set +x
34 The following "npm" command runs your Node.js/React application in
35 development mode and makes the application available for web browsing.
36 The "npm start" command has a trailing ampersand so that the command runs
37 as a background process (i.e. asynchronously). Otherwise, this command
38 can pause running builds of CI/CD applications indefinitely. "npm start"
39 is followed by another command that retrieves the process ID (PID) value
40 of the previously run process (i.e. "npm start") and writes this value to
41 the file ".pidfile".
42 + sleep 1
43 + npm start
44
45 > my-app@0.1.0 start /var/jenkins_home/workspace/peline-in-blue-ocean_master-QHFKIFNIE7FN0ZA0JPKB06ZC46A7K77SYSFL3CJH2A3XUPN0LXUQ
46 > react-scripts start
47
48 + echo 246
49 + set +x
50 Now...
51 Visit http://localhost:3000 to see your Node.js/React application in action.
52 (This is why you specified the "args -p 3000:3000" parameter when you
53 created your initial Pipeline as a Jenkinsfile.)
```

Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input 3m 25s

Finished using the web site? (Click "Proceed" to continue)

Proceed Abort

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.

✓ creating-a-pipeline-in-blue-ocean 3

Pipeline Changes Tests Artifacts ↺ ⚙️ 📄 Logout ✕

Branch: master [🔗](#)

🕒 4m 24s

Changes by gilesgas

Commit: f8f31f2

🕒 a few seconds ago

Branch indexing

Start

Build ✓

Test ✓

Deliver ✓

End

Steps Deliver [🔗](#) [📄](#)

✓ > ./jenkins/scripts/deliver.sh — Shell Script 13s

✓ > Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input 3m 42s

✓ > ./jenkins/scripts/kill.sh — Shell Script <1s

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.

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	↺