

Create a Pipeline in Blue Ocean

Table of Contents

- [Prerequisites](#)
- [Run Jenkins in Docker](#)
 - [On macOS and Linux](#)
 - [On Windows](#)
 - [Accessing the Jenkins/Blue Ocean Docker container](#)
 - [Setup wizard](#)
 - [Stopping and restarting Jenkins](#)
- [Fork the sample repository on GitHub](#)
- [Create your Pipeline project in Blue Ocean](#)
- [Create your initial Pipeline](#)
- [Add a test stage to your Pipeline](#)
- [Add a final deliver stage to your Pipeline](#)
- [Follow up \(optional\)](#)
- [Wrapping up](#)

This tutorial shows you how to use the [Blue Ocean](#) feature of Jenkins to create a Pipeline that will orchestrate building a simple application.

Before starting this tutorial, it is recommended that you run through at least one of the initial set of tutorials from the [Tutorials overview](#) page first to familiarize yourself with CI/CD concepts (relevant to a technology stack you're most familiar with) and how these concepts are implemented in Jenkins.

This tutorial uses the same application that the [Build a Node.js and React app with npm](#) tutorial is based on. Therefore, you'll be building the same application although this time, completely through Blue Ocean. Since Blue Ocean provides a simplified Git-handling experience, you'll be interacting directly with the repository on GitHub (as opposed to a local clone of this repository).

Duration: This tutorial takes 20-40 minutes to complete (assuming you've already met the [prerequisites](#) below). The exact duration will depend on the speed of your machine and whether or not you've already [run Jenkins in Docker](#) from [another tutorial](#).

You can stop this tutorial at any point in time and continue from where you left off.

If you've already run through [another tutorial](#), you can skip the [Prerequisites](#) and [Run Jenkins in Docker](#) sections below and proceed on to [forking the sample repository](#). If you need to restart Jenkins, simply follow the restart instructions in [Stopping and restarting Jenkins](#) and then proceed on.

Prerequisites

For this tutorial, you will require:

- A macOS, Linux or Windows machine with:

- 256 MB of RAM, although more than 512MB is recommended.
 - 10 GB of drive space for Jenkins and your Docker images and containers.
 - The following software installed:
 - [Docker](#) - Read more about installing Docker in the [Installing Docker](#) section of the [Installing Jenkins](#) page.
- Note:** If you use Linux, this tutorial assumes that you are not running Docker commands as the root user, but instead with a single user account that also has access to the other tools used throughout this tutorial.

Run Jenkins in Docker

In this tutorial, you'll be running Jenkins as a Docker container from the [jenkinsci/blueocean](#) Docker image.

To run Jenkins in Docker, follow the relevant instructions below for either [macOS and Linux](#) or [Windows](#).

You can read more about Docker container and image concepts in the [Docker](#) and [Downloading and running Jenkins in Docker](#) sections of the [Installing Jenkins](#) page.

On macOS and Linux

1. Open up a terminal window.
2. Run the jenkinsci/blueocean image as a container in Docker using the following [docker run](#) command (bearing in mind that this command automatically downloads the image if this hasn't been done):

```
docker run \
  --rm \
  -u root \
  -p 8080:8080 \
  -v jenkins-data:/var/jenkins_home \ (1)
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v "$HOME"/:/home \ (2)
  jenkinsci/blueocean
```

Maps the /var/jenkins_home directory in the container to the Docker [volume](#) with the **1** name jenkins-data. If this volume does not exist, then this docker run command will automatically create the volume for you.

2 Maps the \$HOME directory on the host (i.e. your local) machine (usually the /Users/<your-username> directory) to the /home directory in the container.

Note: If copying and pasting the command snippet above doesn't work, try copying and pasting this annotation-free version here:

```
docker run \
  --rm \
  -u root \
  -p 8080:8080 \
  -v jenkins-data:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
```

```
-v "$HOME":/home \
jenkinsci/blueocean
```

3. Proceed to the [Setup wizard](#).

On Windows

1. Open up a command prompt window.
2. Run the jenkinsci/blueocean image as a container in Docker using the following [docker run](#) command (bearing in mind that this command automatically downloads the image if this hasn't been done):

```
docker run ^
  --rm ^
  -u root ^
  -p 8080:8080 ^
  -v jenkins-data:/var/jenkins_home ^
  -v /var/run/docker.sock:/var/run/docker.sock ^
  -v "%HOMEPATH%":"/home ^
  jenkinsci/blueocean
```

For an explanation of these options, refer to the [macOS and Linux](#) instructions above.

3. Proceed to the [Setup wizard](#).

Accessing the Jenkins/Blue Ocean Docker container

If you have some experience with Docker and you wish or need to access the Jenkins/Blue Ocean Docker container through a terminal/command prompt using the [docker exec](#) command, you can add an option like `--name jenkins-tutorials` (with the [docker run](#) above), which would give the Jenkins/Blue Ocean Docker container the name "jenkins-tutorials".

This means you could access the Jenkins/Blue Ocean container (through a separate terminal/command prompt window) with a `docker exec` command like:

```
docker exec -it jenkins-tutorials bash
```

Setup wizard

Before you can access Jenkins, there are a few quick "one-off" steps you'll need to perform.

Unlocking Jenkins

When you first access a new Jenkins instance, you are asked to unlock it using an automatically-generated password.

1. After the 2 sets of asterisks appear in the terminal/command prompt window, browse to `http://localhost:8080` and wait until the **Unlock Jenkins** page appears.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

2. From your terminal/command prompt window again, copy the automatically-generated alphanumeric password (between the 2 sets of asterisks).

```
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@24cf7404: defining beans [filter,legacy]; root of factory hierarchy
Sep 30, 2017 7:18:39 AM jenkins.install.SetupWizard init
INFO:

*****
*****
*****

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

3f064d3663814887964b662940572567

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****

--> setting agent port for jnlp
--> setting agent port for jnlp... done
Sep 30, 2017 7:18:51 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Sep 30, 2017 7:18:52 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Sep 30, 2017 7:18:52 AM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
Sep 30, 2017 7:18:52 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Sep 30, 2017 7:18:58 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Sep 30, 2017 7:18:59 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata. 25,543 ms
```

3. On the **Unlock Jenkins** page, paste this password into the **Administrator password** field and click **Continue**.

Customizing Jenkins with plugins

After [unlocking Jenkins](#), the **Customize Jenkins** page appears.

On this page, click **Install suggested plugins**.

The setup wizard shows the progression of Jenkins being configured and the suggested plugins being installed. This process may take a few minutes.

Creating the first administrator user

Finally, Jenkins asks you to create your first administrator user.

1. When the **Create First Admin User** page appears, specify your details in the respective fields and click **Save and Finish**.
2. When the **Jenkins is ready** page appears, click **Start using Jenkins**.

Notes:

- This page may indicate **Jenkins is almost ready!** instead and if so, click **Restart**.
 - If the page doesn't automatically refresh after a minute, use your web browser to refresh the page manually.
3. If required, log in to Jenkins with the credentials of the user you just created and you're ready to start using Jenkins!

Stopping and restarting Jenkins

Throughout the remainder of this tutorial, you can stop the Jenkins/Blue Ocean Docker container by typing `Ctrl-C` in the terminal/command prompt window from which you ran the `docker run ...` command [above](#).

To restart the Jenkins/Blue Ocean Docker container:

1. Run the same `docker run ...` command you ran for [macOS](#), [Linux](#) or [Windows](#) above.
Note: This process also updates the `jenkinsci/blueocean` Docker image, if an updated one is available.
2. Browse to `http://localhost:8080`.
3. Wait until the log in page appears and log in.

Fork the sample repository on GitHub

Fork the simple "Welcome to React" Node.js and React application on GitHub into your own GitHub account.

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.
Note: This is a different repository to the one used in the [Build a Node.js and React app with npm](#) tutorial. Although these repositories contain the same application code, ensure you fork and use the correct one before continuing on.

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://localhost:8080/blue` and are logged in or
 - have browsed to `http://localhost: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

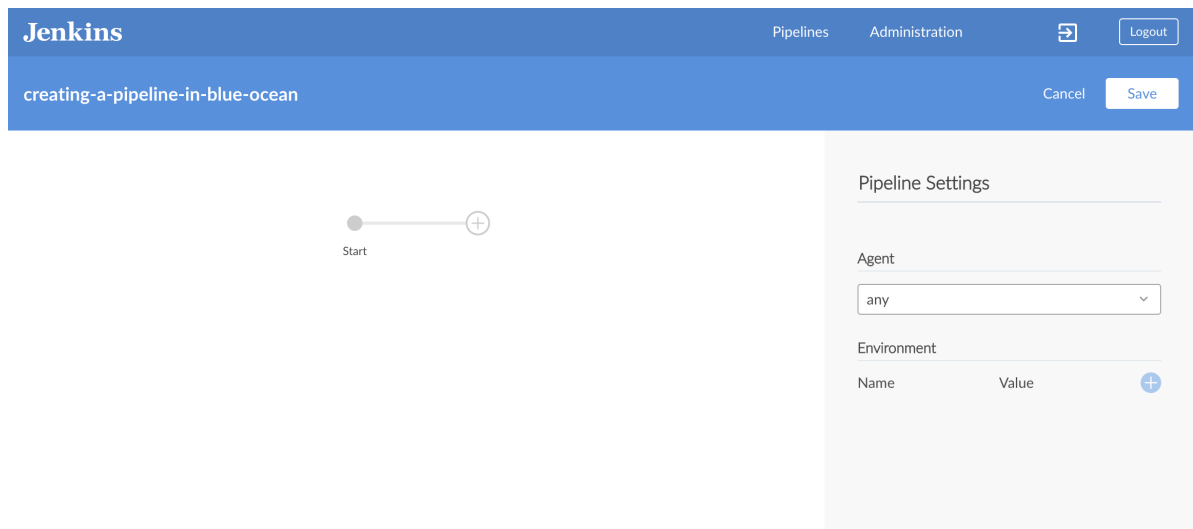
Complete

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.



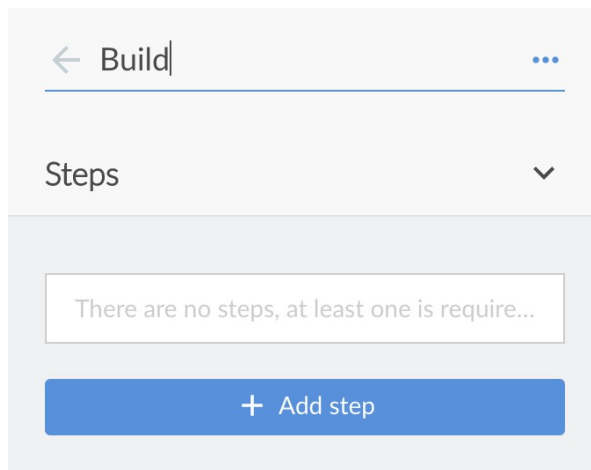
2. In the **Image** and **Args** fields that appear, specify `node:6-alpine` and `-p 3000:3000` respectively.

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](#).

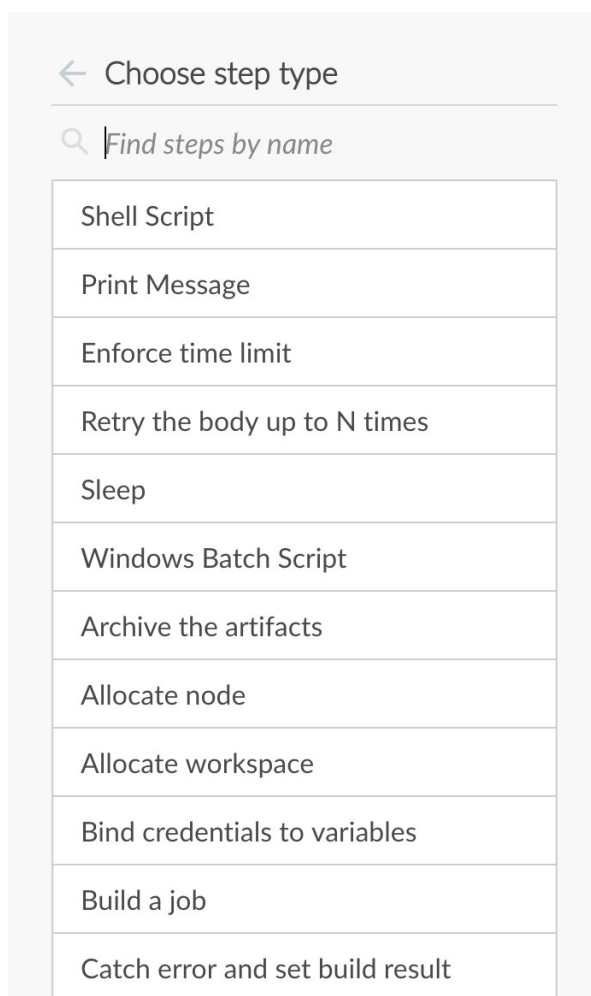
3. Back in the main Pipeline editor, click the + icon, which opens the new stage panel on the right.



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.

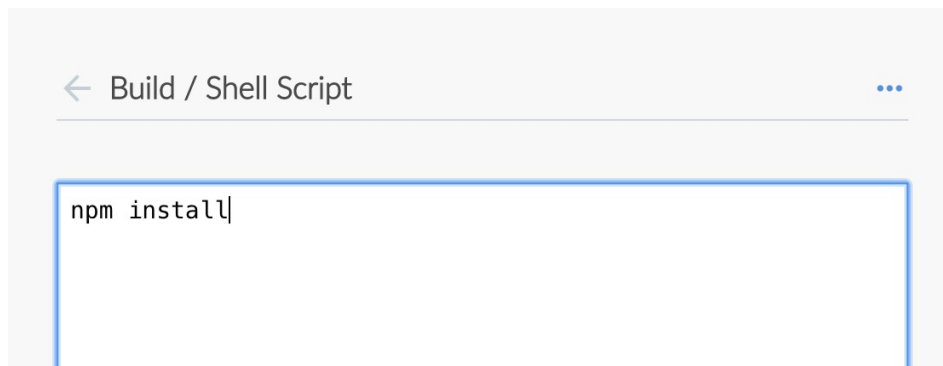


5. In this panel, click **Shell Script** near the top of the list (to choose that step type), which opens the **Build / Shell Script** panel, where you can enter this step's values.




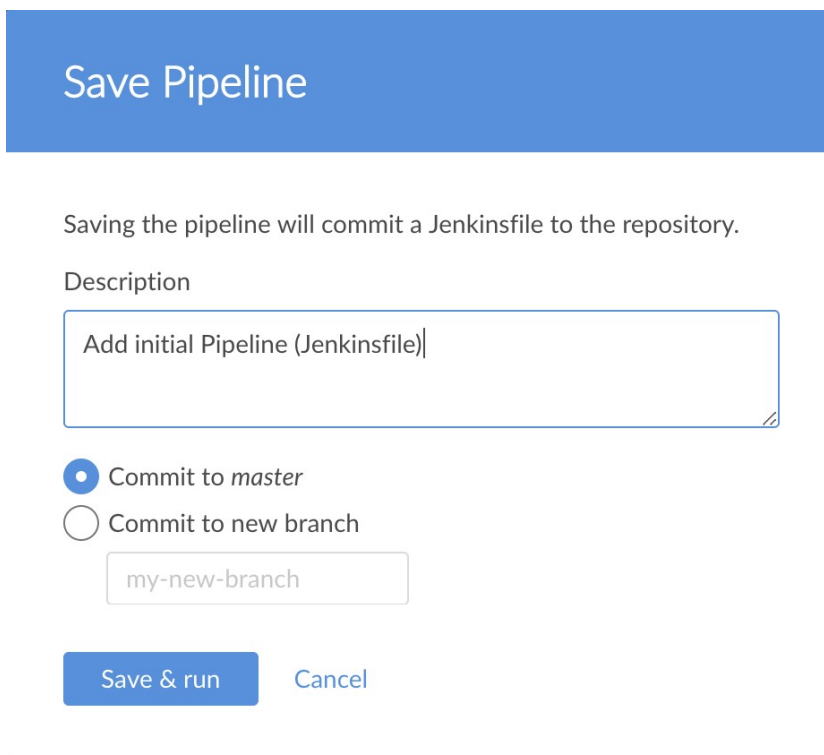
Tip: The most commonly used step types appear closest to the top of this list. To find other steps further down this list, you can filter this list using the **Find steps by name** option.

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)).

A screenshot of the 'Save Pipeline' dialog box. It has a blue header with the text 'Save Pipeline'. Below the header, it says 'Saving the pipeline will commit a Jenkinsfile to the repository.' There is a 'Description' label above a text input field containing 'Add initial Pipeline (Jenkinsfile)'. Below the input field are two radio button options: 'Commit to master' (selected) and 'Commit to new branch'. Under the 'Commit to new branch' option is a text input field containing 'my-new-branch'. At the bottom are two buttons: 'Save & run' and 'Cancel'.

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:
 - a. Commits your Pipeline as a Jenkinsfile to the only branch (i.e. master) of your repository.

- b. Initially queues the project to be built on the agent.
- c. Downloads the Node Docker image and runs it in a container on Docker.
- d. 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).

The screenshot shows the Jenkins Blue Ocean interface for a pipeline named "creating-a-pipeline-in-blue-ocean 1". The interface has a blue header with tabs for Pipeline, Changes, Tests, and Artifacts. Below the header, it shows the current branch as "master" and the commit as "b46b3d4". The build status is "No changes" and "Branch indexing". The build progress bar shows "Start", "Build" (in progress), and "End". Below the progress bar, the "Steps Build" section lists three steps: "docker pull node:6-alpine" (4s), "General SCM" (3s), and "npm install" (29s). The "npm install" step is expanded, showing a terminal output with a red error message: "[pipeline-in-blue-ocean_master-QHFK1FNIE7FN0ZA0JPKB06ZC46A7K77SYSFL3CJH2A3XUPNOLXUQ] Running shell script + npm install".

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

The screenshot shows the Jenkins Blue Ocean interface for the same pipeline "creating-a-pipeline-in-blue-ocean 1". The interface now has a green header. The build status is "No changes" and "Branch indexing". The build progress bar shows "Start", "Build" (completed with a green checkmark), and "End". Below the progress bar, the "Steps Build" section lists three steps: "docker pull node:6-alpine" (16s), "General SCM" (4s), and "npm install" (1m 38s). All steps are marked with green checkmarks, indicating a successful build.

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

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 respository’s branches page, where you can access the master branch.

Jenkins


Pipelines
Administration

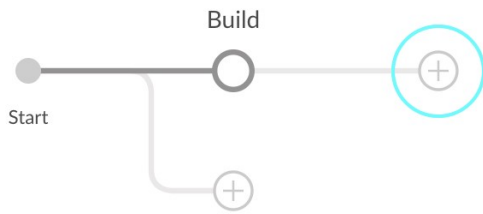
Logout


creating-a-pipeline-in-blue-ocean ☆ ⚙️

Activity
Branches
Pull Requests

· 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	+ -

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).

- Leaving all other options as is, click **Save & run** and Jenkins proceeds to build your amended Pipeline.
- 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.

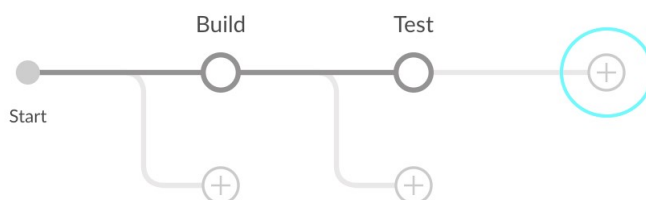
If your amended Pipeline ran successfully, here's what the Blue Ocean interface should look like. Notice the additional "Test" stage. You can click on the previous "Build" stage circle to access the output from that stage.


The screenshot displays the Jenkins Blue Ocean interface for a pipeline named "creating-a-pipeline-in-blue-ocean 2". The top navigation bar includes tabs for Pipeline, Changes, Tests, Artifacts, and a Logout button. Below the navigation bar, the pipeline's status is shown as "Branch: master" with a commit hash "cd16c60" and a duration of "34s". The pipeline's progress is visualized as a horizontal line with four stages: Start, Build, Test, and End. The Build and Test stages are marked with green checkmarks, indicating successful completion. Below the progress bar, the "Steps Test" section is expanded, showing the execution of a shell script. The script's output is displayed in a dark-themed terminal window, showing the installation of the "cross-env" dependency and the successful execution of the "npm test" command.

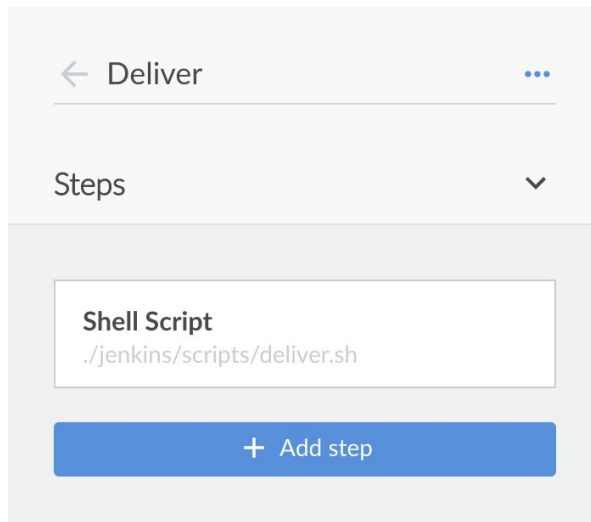
- Click the **X** at the top-right to return to the main Blue Ocean interface.

Add a final deliver stage to your Pipeline

- From the main Blue Ocean interface, click **Branches** at the top-right to access your repository's master branch.
- Click the master branch's "Edit Pipeline" icon  to open the Pipeline editor for this branch.
- In the main Pipeline editor, click the + icon to the right of the **Test** stage you created [above](#) to open the new stage panel.

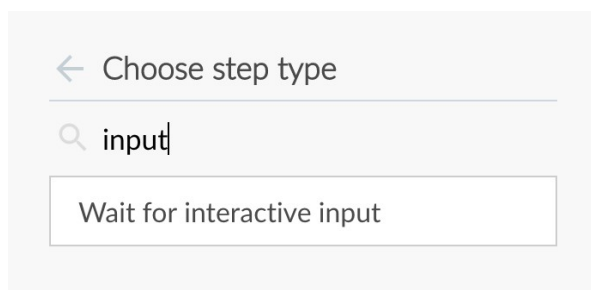



4. In this panel, type **Deliver** in the **Name your stage** field and then click the **Add Step** button below to open the **Choose step type** panel.
5. In this panel, click **Shell Script** near the top of the list.
6. In the resulting **Deliver / Shell Script** panel, specify `./jenkins/scripts/deliver.sh` and then click the top-left back arrow icon  to return to the Pipeline stage editor.



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.

← 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 shows the Jenkins Pipeline interface for a pipeline named "creating-a-pipeline-in-blue-ocean 3". The top bar includes tabs for Pipeline, Changes, Tests, and Artifacts, along with a Logout button. Below the top bar, the current branch is "master" (commit f8f31f2) and the pipeline was last changed by "gilesgas" 2m 22s ago. The pipeline status is "Branch indexing".

The pipeline graph shows five stages: Start, Build, Test, Deliver, and End. The Build and Test stages are marked with green checkmarks, indicating they have completed successfully. The Deliver stage is currently active, indicated by a blue circle with a white dot.

Below the pipeline graph, the "Steps Deliver" section is expanded, showing two steps:

- Step 1: `./jenkins/scripts/deliver.sh` — Shell Script (13s)
- Step 2: Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input (1m 27s)

The second step is currently active, displaying a message: "Finished using the web site? (Click 'Proceed' to continue)". Below the message are 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.

The screenshot shows the Jenkins Pipeline interface for the "Deliver" stage. The "Steps Deliver" section is expanded, showing two steps:

- Step 1: `./jenkins/scripts/deliver.sh` — Shell Script (13s)
- Step 2: Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input (3m 25s)

The second step is currently active, displaying a terminal view of the `./jenkins/scripts/deliver.sh` script. The script content is as follows:

```
24 "homepage" : "http://myname.github.io/myapp",
25
26 The build folder is ready to be deployed.
27 You may serve it with a static server:
28
29 npm install -g serve
30 serve -s build
31
32 + set +x
33 The following "npm" command runs your Node.js/React application in
34 development mode and makes the application available for web browsing.
35 The "npm start" command has a trailing ampersand so that the command runs
36 as a background process (i.e. asynchronously). Otherwise, this command
37 can pause running builds of CI/CD applications indefinitely. "npm start"
38 is followed by another command that retrieves the process ID (PID) value
39 of the previously run process (i.e. "npm start") and writes this value to
40 the file ".pidfile".
41 + sleep 1
42 + npm start
43
44 > my-app@0.1.0 start /var/jenkins_home/workspace/peline-in-blue-ocean_master-QHFKIFNIE7FN0ZA0JPKB06ZC46A7K77SY5FL3CJH2A3XUPNOLXUQ
45 > react-scripts start
46
47 + echo 246
48 + set +x
49 Now...
50 Visit http://localhost:3000 to see your Node.js/React application in action.
51 (This is why you specified the "args -p 3000:3000" parameter when you
52 created your initial Pipeline as a Jenkinsfile.)
53
```

Below the terminal view, the second step is currently active, displaying a message: "Finished using the web site? (Click 'Proceed' to continue)". Below the message are two buttons: "Proceed" and "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

PipelineChangesTestsArtifactsLogout

Branch: master

Commit: f8f31f2

4m 24s

a few seconds ago

Changes by gilesgas

Branch indexing

Start

Build

Test

Deliver

End

Steps Deliver

✓> ./jenkins/scripts/deliver.sh — Shell Script13s

✓> Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input3m 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.

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

Follow up (optional)

If you check the contents of the Jenkinsfile that Blue Ocean created at the root of your forked `creating-a-pipeline-in-blue-ocean` repository, notice the location of the `environment` directive. This directive's location within the "Test" stage means that the environment variable `CI` (with its value of `true`) is only available within the scope of this "Test" stage.

You can set this directive in Blue Ocean so that its environment variable is available globally throughout Pipeline (as is the case in the [Build a Node.js and React app with npm](#) tutorial). To do this:

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 **Test** stage you created [above](#) to begin editing it.
4. In the stage panel on the right, click **Settings** to reveal this section of the panel.
5. Click the minus (-) icon at the right of the CI environment directive (you created earlier) to delete it.
6. Click the top-left back arrow icon  to return to the main Pipeline editor.
7. In the **Pipeline Settings** panel, click the + icon at the right of the **Environment** heading (for which you'll configure a *global* environment directive).
8. In the **Name** and **Value** fields that appear, specify CI and true, respectively.
9. Click the **Save** button at the top right to begin saving your Pipeline with its relocated environment directive.
10. In the **Save Pipeline** dialog box, specify the commit message in the **Description** field (e.g. Make environment directive global).
11. Leaving all other options as is, click **Save & run** and Jenkins proceeds to build your amended Pipeline.
12. When the main Blue Ocean interface appears, click the *top* row to see Jenkins build your Pipeline project.
You should see the same build process you saw when you completed adding the final deliver stage ([above](#)). However, when you inspect the Jenkinsfile again, you'll notice that the environment directive is now a sibling of the agent section.

Wrapping up

Well done! You've just used the Blue Ocean feature of Jenkins to build a simple Node.js and React application with npm!

The "Build", "Test" and "Deliver" stages you created above are the basis for building other applications in Jenkins with any technology stack, including more complex applications and ones that combine multiple technology stacks together.

Because Jenkins is extremely extensible, it can be modified and configured to handle practically any aspect of build orchestration and automation.

To learn more about what Jenkins can do, check out:

- The [Tutorials overview](#) page for other introductory tutorials.

- The [User Handbook](#) for more detailed information about using Jenkins, such as [Pipelines](#) (in particular [Pipeline syntax](#)) and the [Blue Ocean](#) interface.
 - The [Jenkins blog](#) for the latest events, other tutorials and updates.
-

[Was this page helpful?](#)

Please submit your feedback about this page through this [quick form](#).

Alternatively, if you don't wish to complete the quick form, you can simply indicate if you found this page helpful?

☐ Yes ☐ No

Type the answer to 9 plus 4 before clicking "Submit" below.

Submit

See existing feedback [here](#).

[Improve this page](#) | [Page history](#)



The content driving this site is licensed under the Creative Commons Attribution-ShareAlike 4.0 license.

Resources

- [Events](#)
- [Documentation](#)
- [Blog](#)

Solutions

- [Android](#)
- [C/C++](#)
- [Docker](#)
- [Embedded](#)
- [GitHub](#)
- [Java](#)
- [PHP](#)
- [Continuous Delivery](#)
- [Python](#)
- [Ruby](#)

Project

- [Issue tracker](#)
- [Wiki](#)

- [GitHub](#)
- [Jenkins on Jenkins](#)

Community

- [Users mailing list](#)
- [Developers mailing list](#)
- [Special Interest Groups](#)
- [Twitter](#)
- [Reddit](#)
- [Merchandise](#)
- [Awards](#)