

Lab Documentation

Integrating NODE.JS project with CircleCI, to build, test your project.

CircleCI: CI & CD platform that helps the development teams to release code rapidly and automate the build, test and deploy.

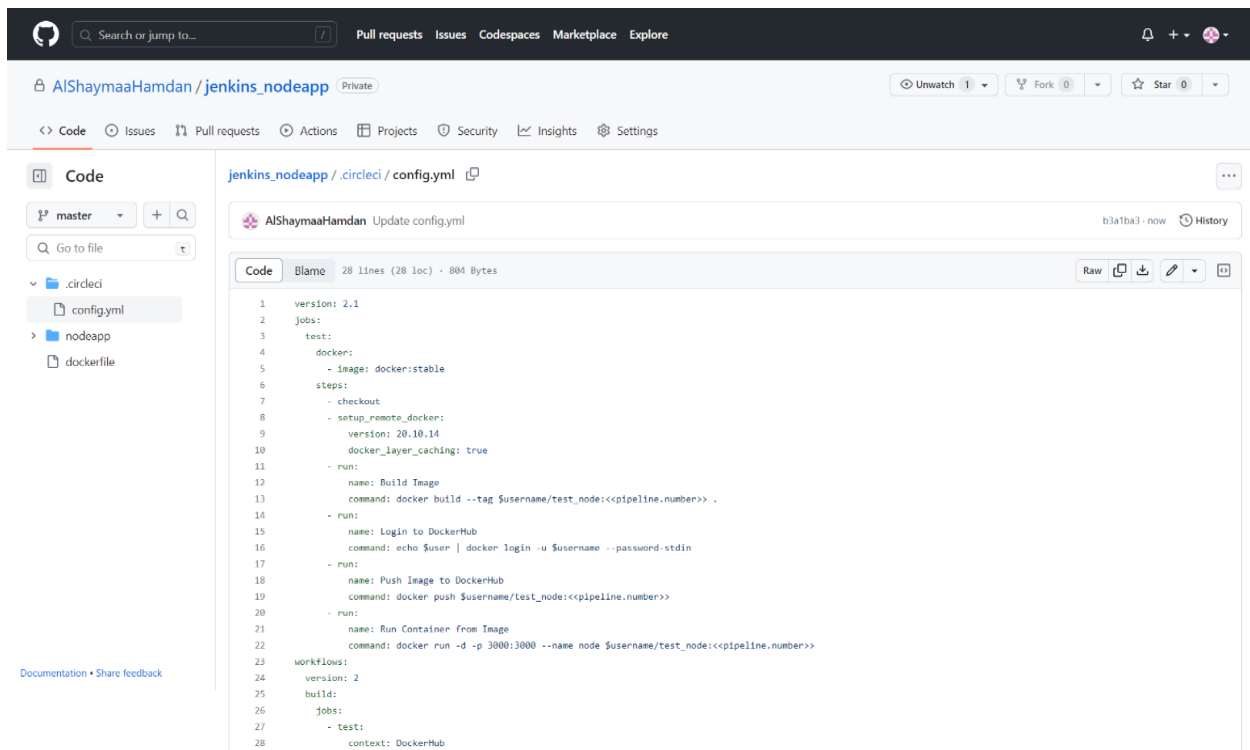
Step 1: Sign Up to CircleCI with GitHub

- Click on Sign Up with GitHub to start authentication process and allow CircleCI to access your code.
- Add your GitHub username and password to sign in.
- After CircleCI is opened, Go to Projects Page.

Step 2: Write CircleCI Configuration File

To connect your project with CircleCI, we use a YAML configuration file where the entire delivery process from build to deploy is orchestrated.

- On your GitHub private repository add `config.yml` file on `.circleci` directory, file path: `(.circleci/config.yml)`.
- Update the contents of `config.yml` file as below.



The screenshot shows a GitHub repository for 'AlShaymaaHamdan / jenkins_nodeapp'. The file '.circleci/config.yml' is selected, showing its contents. The file is a CircleCI configuration in YAML format, defining a test job that uses Docker to build and run a Node.js application.

```

1  version: 2.1
2  jobs:
3    test:
4      docker:
5        - image: docker:stable
6      steps:
7        - checkout
8        - setup_remote_docker:
9          version: 20.10.14
10         docker_layer_caching: true
11        - run:
12          name: Build Image
13          command: docker build --tag $username/test_node:<<pipeline.number>> .
14        - run:
15          name: Login to DockerHub
16          command: echo $user | docker login -u $username --password-stdin
17        - run:
18          name: Push Image to DockerHub
19          command: docker push $username/test_node:<<pipeline.number>>
20        - run:
21          name: Run Container from Image
22          command: docker run -d -p 3000:3000 --name node $username/test_node:<<pipeline.number>>
23  workflows:
24    version: 2
25    build:
26      jobs:
27        - test:
28          context: DockerHub
  
```

CircleCI config file consist of two parts:

- Jobs: is an atomic unit of work or task that you want to accomplish within your overall process.

Jobs consist of:

- executor: the execution environment for each job.

Like, Docker executor that run jobs in Docker containers and machine executor in virtual machines with Linux, macOS or Windows images.

CircleCI provides convenience images for a variety of use-cases:

<https://circleci.com/developer/images>

- steps: the actions and commands
- Workflow: One or more jobs that form a full process.

Sets dependencies between jobs

config.yml content:

version: 2.1	
jobs: test:	#define jobs #Job name is test
docker: - image: docker:stable auth: username: \$username password: \$user	#Executor Type #This image specifies the container you want to start for your job #authenticate your DockerHub account. Username and password are stored in env-var in CircleCI Context called "DockerHub"
steps: - checkout - setup_remote_docker: version: 20.10.14 docker_layer_caching: true	#Commands run in the container #To checkout your code (Cloning git repository). #To build Docker images. any Docker commands you run in your job will be executed locally on the virtual machine used to spin up your primary Docker container. # DLC saves Docker image layers created within your jobs, and caches them to be reused during future builds.
- run: name: Build Image command: docker build --tag \$username/test_node:<<pipeline.number>> . - run: name: Login to DockerHub command: echo \$user docker login -u \$username --password-stdin	Commands: # Build Image using Dockerfile and tag it with your DockerHub account name and pipeline number.

<pre> - run: name: Push Image to DockerHub command: docker push \$username/test_node:<<pipeline.number>> - run: name: Run Container from Image command: docker run -d -p 3000:3000 --name node \$username/test_node:<<pipeline.number>> </pre>	<p>Pipeline number is a “Pipeline Value” that are available to all pipeline configurations and can be used without previous declaration.</p> <p># Login to your DockerHub account using username and password env-var in CircleCI Context.</p> <p># Push the image to DockerHub with the same tag</p> <p># Run a container from the image you’ve pushed to DockerHub</p>
<pre> workflows: version: 2 build: jobs: - test: context: DockerHub </pre>	<p># Used for orchestrating all jobs</p> <p># Workflow name is build</p> <p># Specify jobs for the workflow</p> <p># The context used in “test” job is DockerHub, contains two env-var:</p> <p>Username: DockerHub username</p> <p>User: DockerHub password</p>

Step 4: Create a Context

To add your DockerHub account credentials, we use env-vars

- Click on “Organization Settings” on CircleCI, then choose “Contexts”
- Click on “Create Context” then write a unique name.
- Click on “Add Environment Variable”
- Name your Env-var Username and give it a value of your DockerHub username
- The same to DockerHub password.
The password could be an access token or your password.

Organization Settings
AlshaimaHamdan

Overview
Contexts
VCS
Security
Policies
Orbs
Self-Hosted Runners

Contexts > DockerHub

DockerHub

Security
Add Security Group

Groups listed are able to execute this context on a workflow.

Group Name

All members

Project Restrictions
Add Project Restriction

Restricting a context to specific projects ensures that only trusted projects can access secrets during builds. By default, a context with no restrictions is available to all projects.

Environment Variables
Add Environment Variable

Environment variables are available to any job that requests this context. See [Using Environment Variables](#) documentation.

Name	Value	Created	Last Updated	
user	****RzA	May 14, 2023 08:45 UTC	May 14, 2023 08:45 UTC	×
username	****aa12	May 14, 2023 08:35 UTC	May 14, 2023 08:35 UTC	×

To create access token in DockerHub:

- Go to “Account Settings”
- Click on “Security”, then “New Access Token”

The screenshot shows the Docker Hub account settings page for user 'shaymaa12'. The user's profile is at the top, showing a fingerprint icon, the username 'shaymaa12', and the status 'User' with a join date of 'April 5, 2023'. The 'Security' tab is selected in the left sidebar, which also includes 'General', 'Default Privacy', 'Notifications', 'Convert Account', and 'Deactivate Account'. The main content area is divided into two sections: 'Access Tokens' and 'Two-Factor Authentication'. The 'Access Tokens' section has a 'New Access Token' button and a table with one token for 'CircleCI'. The 'Two-Factor Authentication' section states it is not enabled yet and provides an 'Enable Two-Factor Authentication' button. The footer contains navigation links for 'Explore', 'Account', 'Resources', and 'Support', along with copyright information and social media icons.

Access Tokens

<input type="checkbox"/>	Description	Scope	Last Used	Created	Active
<input type="checkbox"/>	CircleCI	Read, Write, Delete	May 15, 2023 10:34:28	May 14, 2023 11:43:50	Yes

Two-Factor Authentication

Two-factor authentication is not enabled yet. Two-factor authentication adds an extra layer of security to your account by requiring more than just a password to sign in. [Learn more](#)

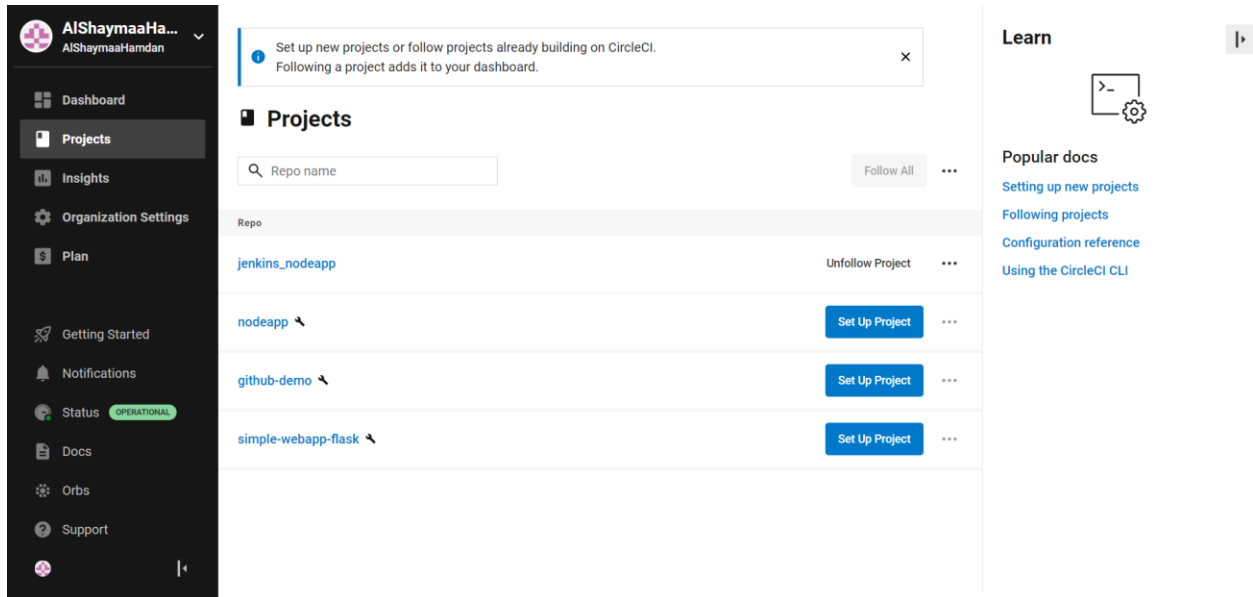
[Enable Two-Factor Authentication](#)

© 2023 Docker Inc. All rights reserved | [Terms of Service](#) | [Subscription Service Agreement](#) | [Privacy](#) | [Legal](#)

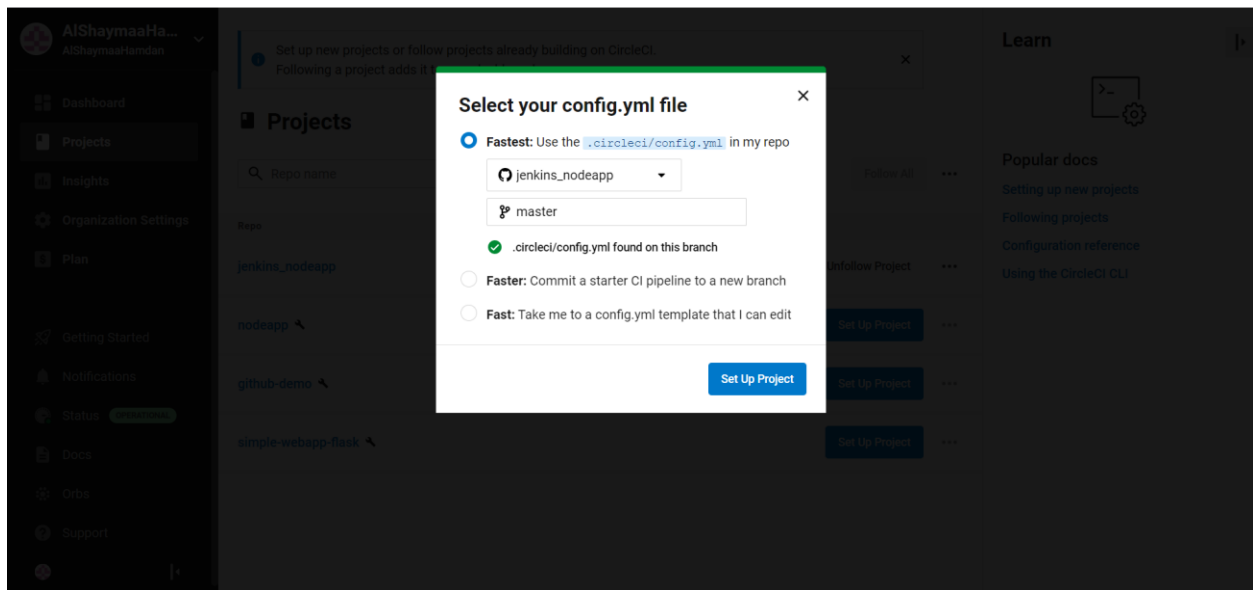
[Cookie Settings](#)

Step 5: Setup and Build your Project

- Choose which GitHub project you want to setup (I chose Jenkins_nodeapp)



- Write the branch name that contains the config.yml file



- The first “Pipeline” will start automatically.
“Dashboard” page contains all “Pipelines” for all of your projects.
Pipeline: is composed of workflows, which are composed of jobs. By navigating from a pipeline to a specific job, you can access your job output, test results, and artifacts through several tabs.
Pipelines trigger when a change is pushed to a project that has a CircleCI configuration file included, and can also be scheduled, triggered manually.

AlShaymaaHamdan

- Dashboard
- Projects
- Insights
- Organization Settings
- Plan
- Getting Started
- Notifications
- Status **OPERATIONAL**
- Docs
- Orbs
- Support

jenkins_nodeapp [Add team members](#) [Edit Config](#) [Trigger Pipeline](#) [Project Settings](#)

Filters: [Everyone's Pipelines](#) [jenkins_nodeapp](#) [All Branches](#) [All days](#) [Auto-expand](#)

Pipeline	Status	Workflow	Branch / Commit	Start	Duration	Actions
jenkins_nodeapp p45	Success	build	master b3a1ba3 Update config.yml	1h ago	21s ↓ 34%	Refresh Cancel More
jenkins_nodeapp p44	Success	build	master 51203a7 Update config.yml	3h ago	20s ↓ 37%	Refresh Cancel More
jenkins_nodeapp p43	Failed	build	master e32e7a7 Update config.yml	3h ago	16s	Refresh Cancel More
jenkins_nodeapp p42	Success	build	master d24b67f Update config.yml	3h ago	19s ↓ 40%	Refresh Cancel More

- Click on “Success” to view the “Pipeline Steps”

AlShaymaaHamdan

- Dashboard
- Projects
- Insights
- Organization Settings
- Plan
- Getting Started
- Notifications
- Status **OPERATIONAL**
- Docs
- Orbs
- Support

test [Success](#) [Rerun](#) [More](#)

Duration / Finished: 6s / 2h ago | Queued / Preparing: 0s / 14s | Executor / Resource Class: Remote Docker / Large [?](#) | Branch: master | Commit: b3a1ba3 | Author & Message: Update config.yml

STEPS TESTS TIMING ARTIFACTS RESOURCES **NEW**

Spin up environment	8s	Copy Download
Spin up container environment	4s	Copy Download
Preparing environment variables	0s	Copy Download
Checkout code	0s	Copy Download
Setup a remote Docker engine	0s	Copy Download
Build Image	0s	Copy Download
Login to DockerHub	0s	Copy Download
Push Image to DockerHub	4s	Copy Download
Run Container from Image	0s	Copy Download
DLC Teardown	6s	Copy Download

Step 6: Check your DockerHub

The image will appear on your account with tags of the pipeline number.

The screenshot shows the Docker Hub interface for a repository named `shaymaa12/test_node`. The page includes a header with the Docker Hub logo, a search bar, and navigation links. The repository page itself has a blue header with the repository name and a 'General' tab selected. Below the header, there is a section for adding a short description, a 'Docker commands' section with a 'Public View' button, and a 'Tags' section showing a table of tags. The 'Tags' table has columns for Tag, OS, Type, Pulled, and Pushed. The tags are 45, 44, and Its, all of type Image, pushed 2, 4, and 4 hours ago respectively. There is also an 'Automated Builds' section and a 'README' section at the bottom.

Repository: shaymaa12/test_node

Description: This repository does not have a description. [Add a short description](#)

Docker commands: To push a new tag to this repository, `docker push shaymaa12/test_node:tagname`

Tags: This repository contains 3 tag(s).

Tag	OS	Type	Pulled	Pushed
45	linux	Image	---	2 hours ago
44	linux	Image	---	4 hours ago
Its	linux	Image	---	4 hours ago

[See all](#) [Go to Advanced Image Management](#)

Automated Builds: Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating. [Read more about automated builds](#)

README: Repository description is empty. [Click here](#) to edit.