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.

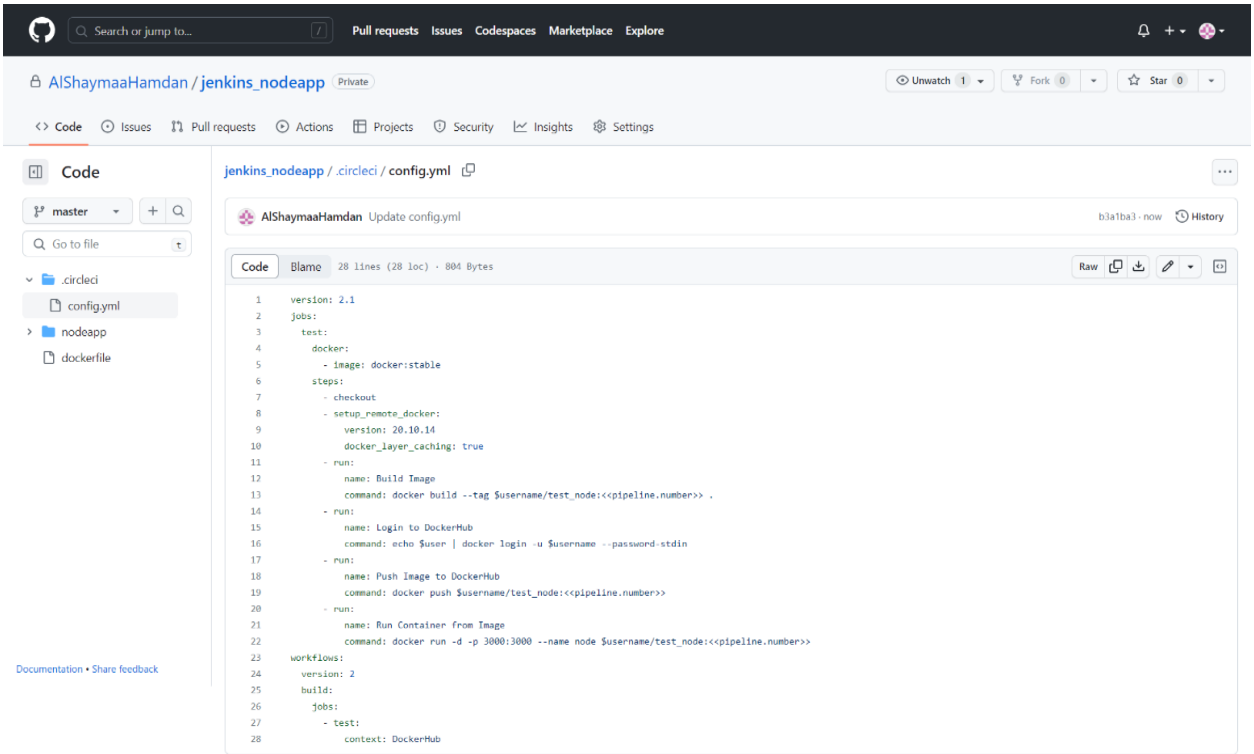
Step 1: Sign Up to CircleCI with GitHub

- 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.



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:

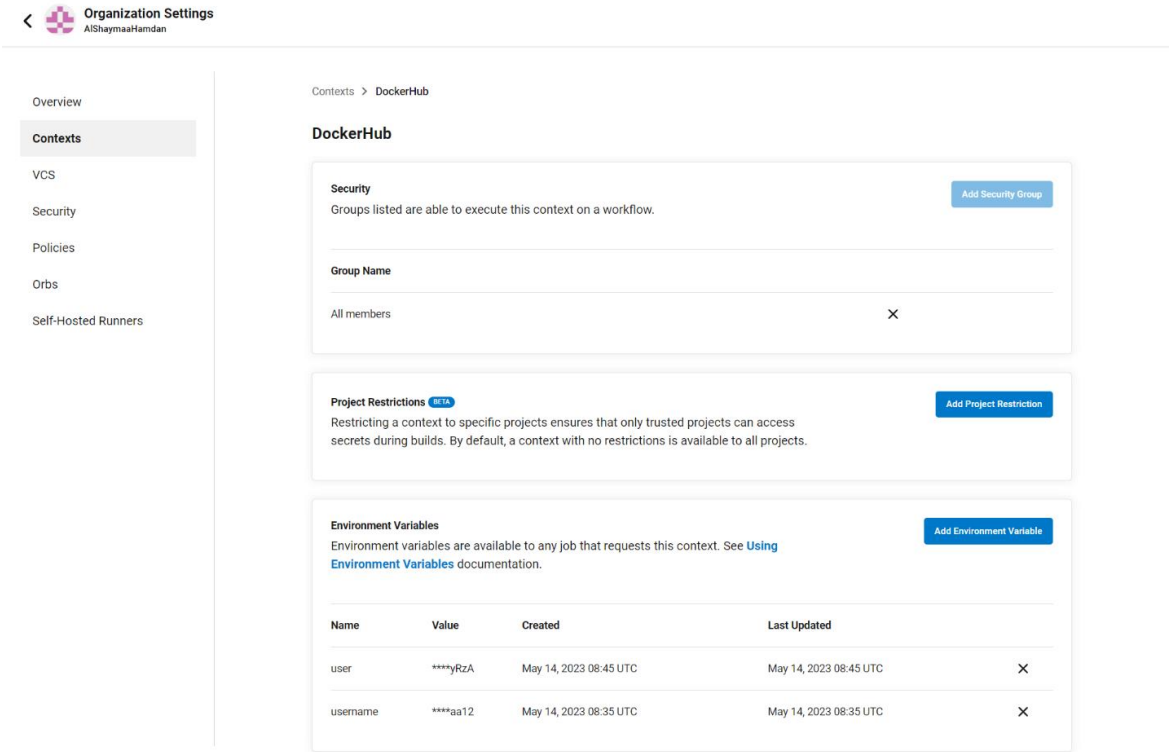
version: 2.1	
jobs: test:	#define jobs #Job name is test
docker: - image: docker:stable auth: username: $username password: $user	#Executor Type #This image specifies the container you want to start for your job #authenticate your DockerHub account. Username and password are stored in env-var in CircleCI 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.

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

Step 4: Create a Context

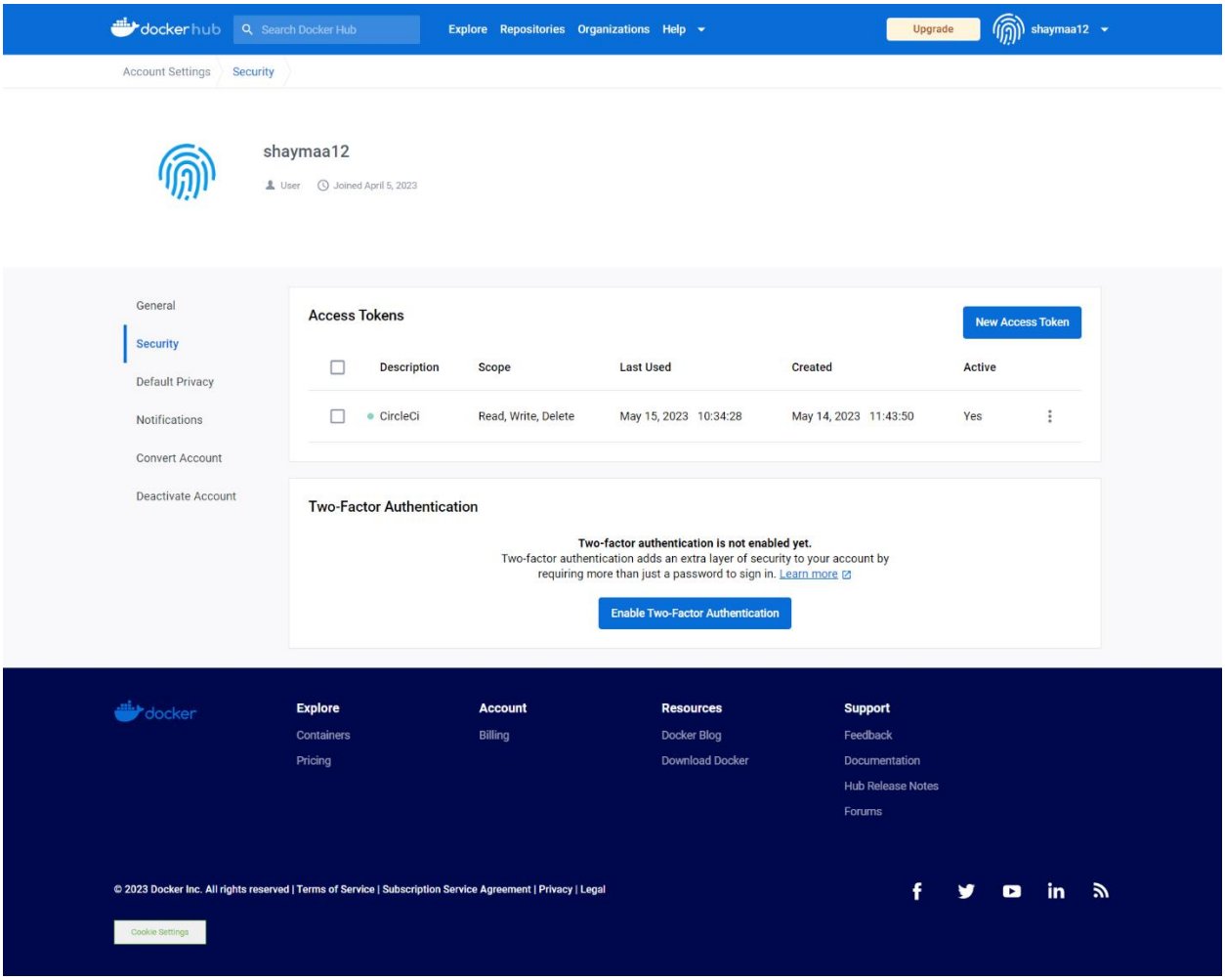
To add your DockerHub account credentials, we use env-vars

- 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.



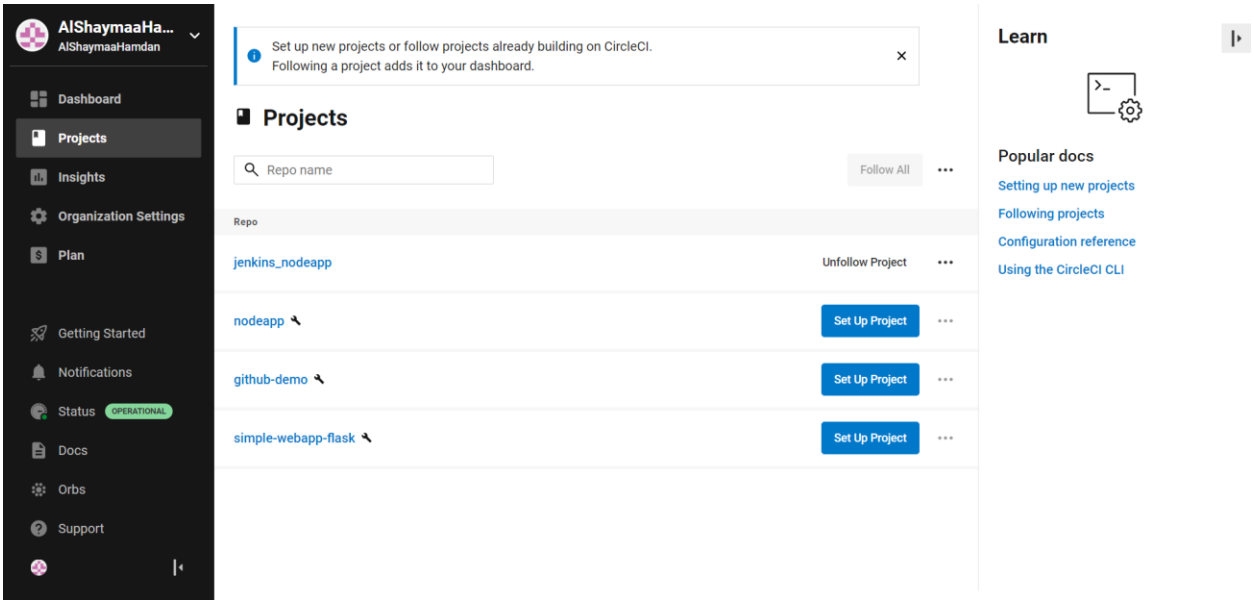
To create access token in DockerHub:

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

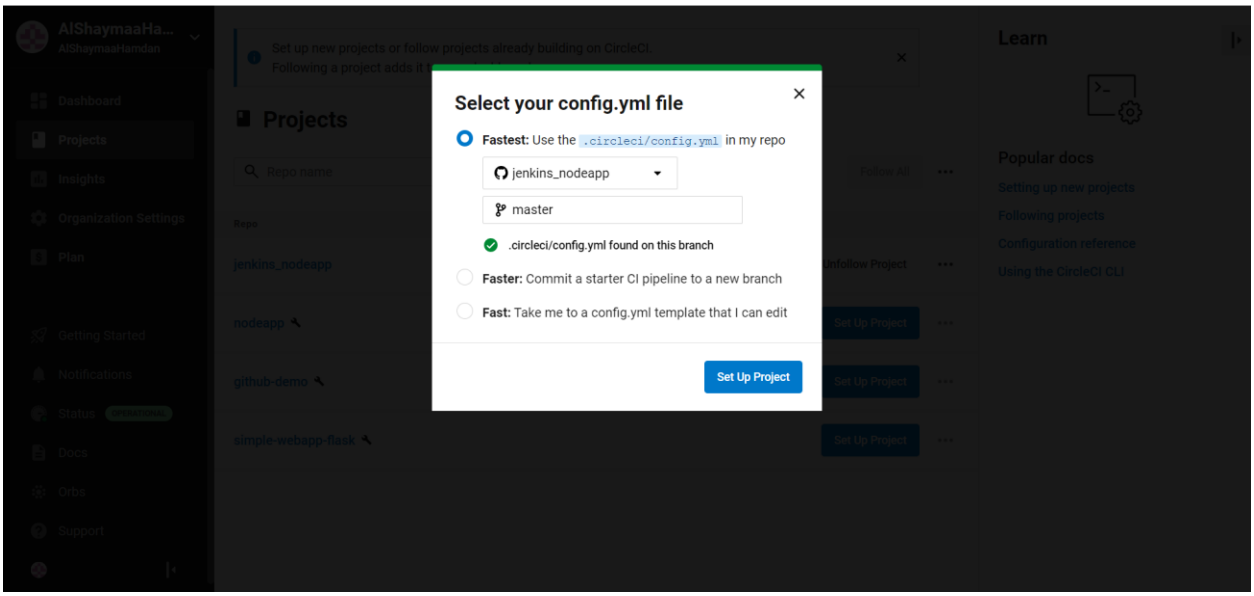


Step 5: Setup and Build your Project

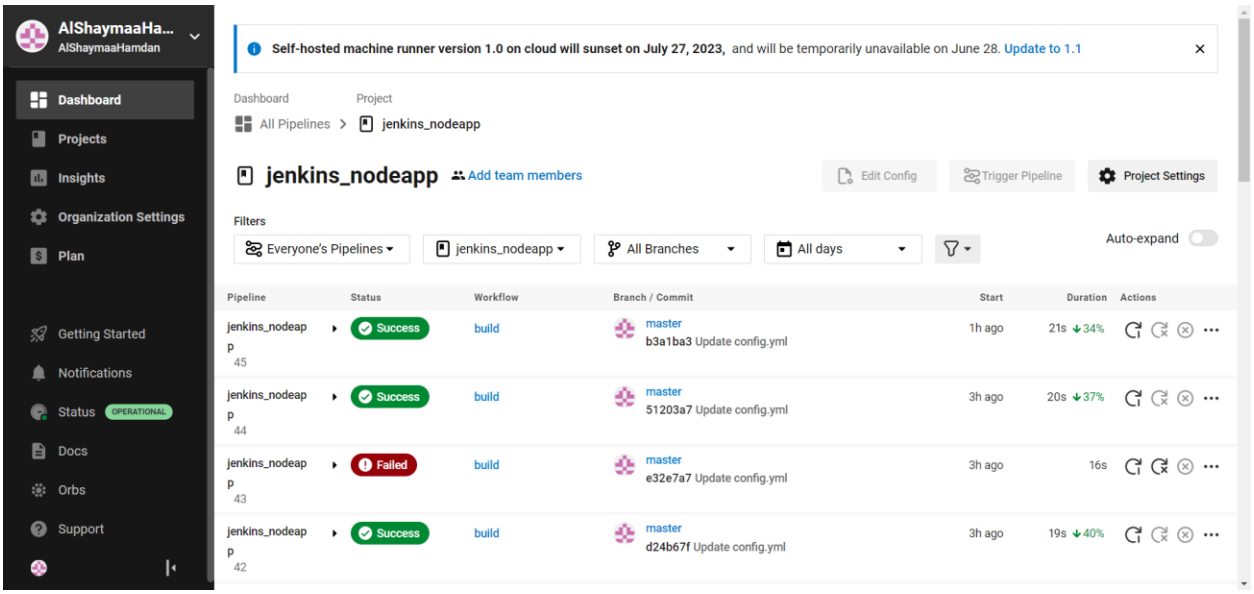
- Choose which GitHub project you want to setup (I chose Jenkins_nodeapp)



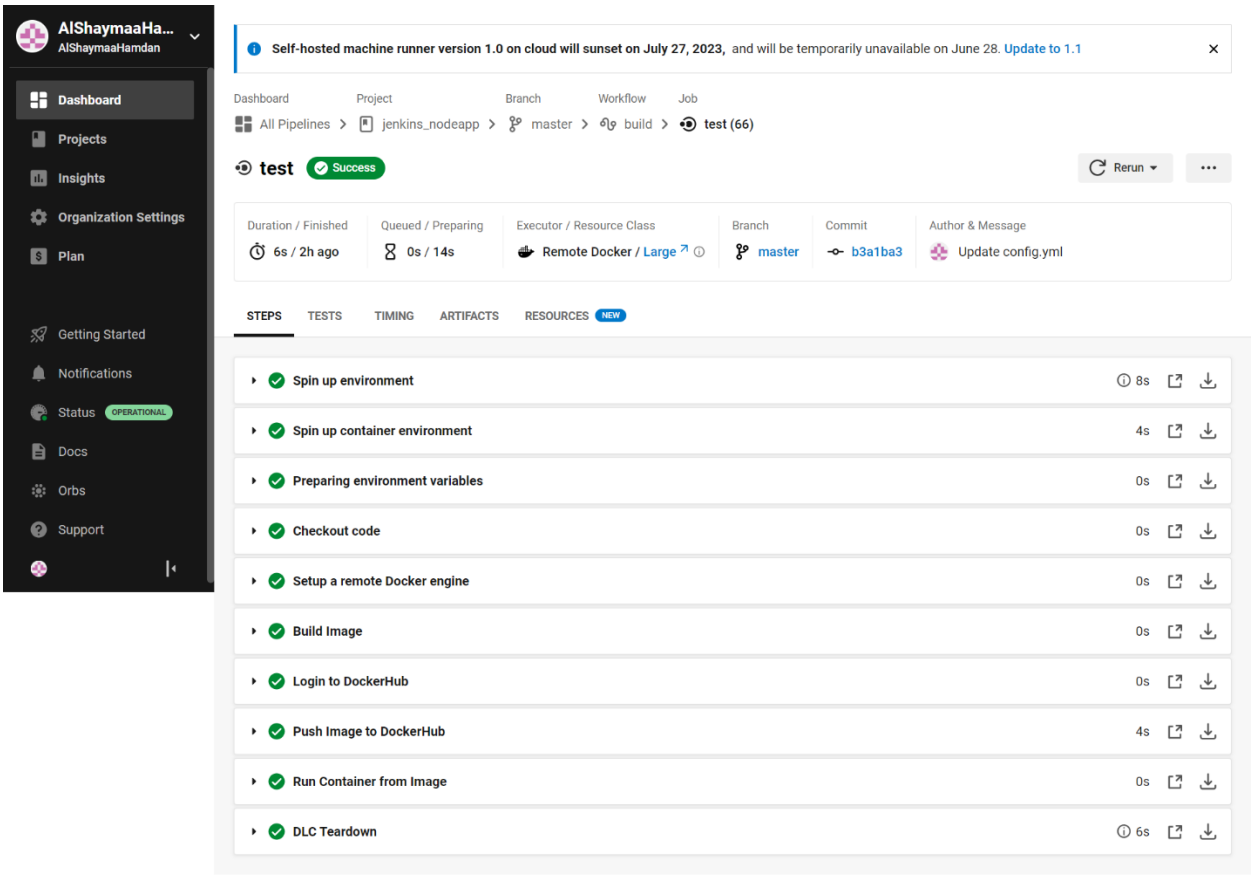
- Write the branch name that contains the config.yml file



- 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.



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



Step 6: Check your DockerHub

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