

# Jenkins Pipeline to Create Docker Image and Push to Dockerhub



Brandon Jones · Follow

Published in The Startup · 5 min read · Nov 4, 2020



200





Recently, I have been spending some time learning Jenkins and automating tasks. I wanted to figure out a way to create a pipeline which pulled from a Github repo, created a docker image, and pushed the image to Dockerhub. I was unable to find a guide that walked users through this simple task and ended up piecing together several posts to complete the task. As a Jenkins newbie this can be frustrating so I wanted to create a blog post which walked

users through the process. All the code for this can be found at the repo [here](#).

The project I turned into an image was just a simple Angular application. I literally just ran the `ng new <new-project-name>` command and used that for my test pipeline project. This Jenkinsfile should work for any project that you have a Dockerfile for and able to create an image. I've used this method for express projects too, just using different Dockerfiles.

First just create the Angular project...

```
ng new <new-project-name>
```

Medium

 Search

 Write

Sign up

Sign in



within the root directory of the project. The Dockerfile is pretty straightforward, but I can walk you through how it works. The FROM command is using node as the base for the image which the entire application is built. The RUN `mkdir -p /app` creates an app directory and WORKDIR indicates this is where the application will be created. COPY

copies the package.json files into the working directory and RUN npm install installs all the dependencies located in the package.json file. The remaining files are copied over using the COPY command. Since it's an angular application, port 4200 is exposed and the CMD npm run start command is run to start the application.

To test the Dockerfile, move into the root directory of the project and run the command “docker build .” and don't forget the ‘.’ at the end. This indicates there is a Dockerfile found in the current directory. The output should be like mine below. If there are any errors something is most likely wrong with the Dockerfile.

```
FROM node:latest as node
RUN mkdir -p /app
WORKDIR /app
COPY package*.json /app/
RUN npm install
COPY . /app/
EXPOSE 4200
CMD ["npm", "run", "start"]
```

```
Sending build context to Docker daemon 283.8MB
Step 1/8 : FROM node:latest as node
---> 5377c9a2fb1f
Step 2/8 : RUN mkdir -p /app
---> Using cache
---> dd2b639050ca
Step 3/8 : WORKDIR /app
---> Using cache
---> 43815b4deda5
Step 4/8 : COPY package*.json /app/
---> Using cache
---> 5cb9c82f03ed
Step 5/8 : RUN npm install
---> Using cache
---> 6be8961b71c2
Step 6/8 : COPY . /app/
---> 8509d721a7f2
Step 7/8 : EXPOSE 4200
---> Running in 64a92cde1453
Removing intermediate container 64a92cde1453
---> d5d92b908a88
Step 8/8 : CMD ["npm", "run", "start"]
---> Running in ed3b24ce7d96
Removing intermediate container ed3b24ce7d96
---> 4fdfe06af10f
Successfully built 4fdfe06af10f
```

Output from docker build . command

The Jenkins pipeline depends on a Jenkinsfile and you can find mine [here](#). Jenkins files can be pretty complex, but I kept mine very simple for learning purposes. The Jenkinsfile is divided into 4 stages, a clone, build, test, and push stage. The clone stage checks out the repo from github. The build stage builds the image and stores it in a variable named 'app'. Be sure to change 'brandonjones085' to which Dockerhub repo you'd like to push the image to. I left the Test stage in the file as a placeholder for future unit tests. At this


point, the logs will just echo 'Tests'. Finally, the image is pushed to Dockerhub with the 'latest' tag and using the stored 'git' credentials. Create this file in the same root directory as the Dockerfile that was previously created.


Once you have the Jenkinsfile created, create a Github repo and push the entire project to the repo.


```
node {  
    def app  
    stage('Clone repository') {  
  
        checkout scm  
    }  
  
    stage('Build image') {  
  
        app = docker.build("brandonjones085/test")  
    }  
  
    stage('Test image') {  
        app.inside {  
  
            sh 'echo "Tests passed"'  
        }  
    }  
  
    stage('Push image') {  
  
        docker.withRegistry('https://registry.hub.docker.com', 'git') {
```


```
    app.push("${env.BUILD_NUMBER}")  
    app.push("latest")  
  }  
}
```

Now we can begin working in Jenkins and creating the project. I'm assuming you already have a Jenkins server installed and running. First, let's add the Dockerhub credentials in Jenkins. These credentials will be used to log into Dockerhub. Click Manage Jenkins, then Manage Credentials.


Jenkins


 ?


 brandon jones


 log out


Dashboard >


 New Item


 People


 Build History


 Project Relationship

 Check File Fingerprint

 Manage Jenkins

 My Views

 Lockable Resources

 New View

Build Queue ^

No builds in the queue.

Build Executor Status ^


1 Idle

2 Idle


Manage Jenkins

New version of Jenkins (2.264) is available for [download](#) ([changelog](#)).


System Configuration




**Configure System**  
Configure global settings and paths.



**Global Tool Configuration**  
Configure tools, their locations and automatic installers.




**Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.  
⚠ There are updates available




**Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.


Security




**Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.



**Manage Credentials**  
Configure credentials



**Configure Credential Providers**  
Configure the credential providers and types



**Manage Users**  
Create/delete/modify users that can log in to this Jenkins

Manage Credentials



Dashboard > Credentials

New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Credentials

T	P	Store	Domain	ID	Name
		Jenkins	(global)	github_creds	brandonjones085/*****
		Jenkins	(global)	brandonjones085	dockerhub/***** (dockerhub account)
		Jenkins	(global)	git	brandonjones085/*****

Icon: S M L


Stores scoped to Jenkins

P	Store	Domains
	Jenkins	(global)


Global

Click global, then Add Credentials to add a new credential with a Global Scope.


Scope


Global (Jenkins, nodes, items, all child items, etc) 

Username


brandonjones085 

Password


 Concealed

Change Password 

ID

git 

Description



Save

Enter Credentials

Add your Dockerhub username and password. The ID is was is used in the Jenkinsfile and your credentials are stored and you can see this used in the Jenkinsfile.

## Enter an item name

» Required field



### Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be



### Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflow type).



### Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform



### Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder as long as they are in different folders.



### GitHub Organization

Scans a GitHub organization (or user account) for all repositories matching some defined markers.



### Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

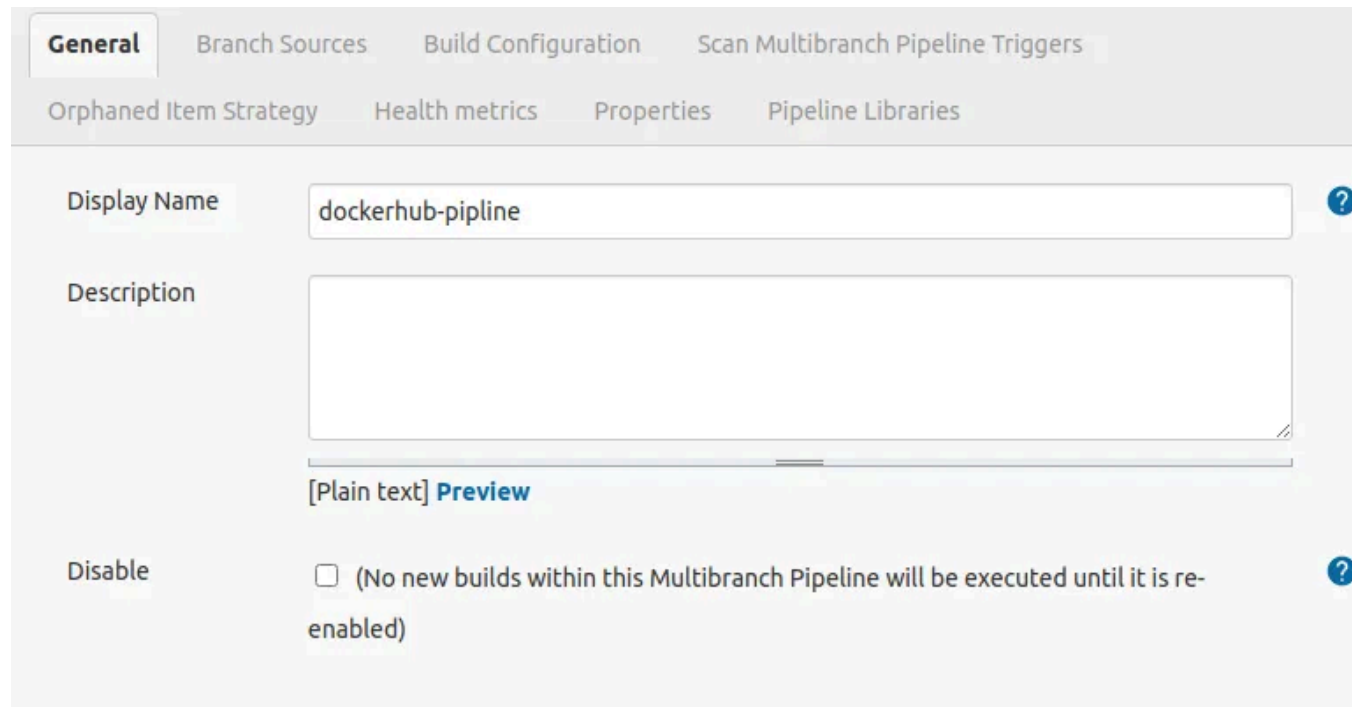
OK

Copy from Type to autocomplete

New Multibranch Pipeline Project

Now the pipeline is ready to be created. Go back to the dashboard and select new Item. Enter a name for the new item, select Multibranch Pipeline and click OK.

Enter a Display Name for the pipeline.



The screenshot shows the Jenkins configuration page for a Multibranch Pipeline. The 'General' tab is selected, and the 'Display Name' field contains 'dockerhub-pipeline'. The 'Description' field is empty. Below the description, there is a '[Plain text] Preview' button. The 'Disable' checkbox is unchecked, and the text '(No new builds within this Multibranch Pipeline will be executed until it is re-enabled)' is displayed. There are help icons (question marks) next to the 'Display Name' and 'Disable' sections.

General	Branch Sources	Build Configuration	Scan Multibranch Pipeline Triggers
Orphaned Item Strategy	Health metrics	Properties	Pipeline Libraries

Display Name:

Description:

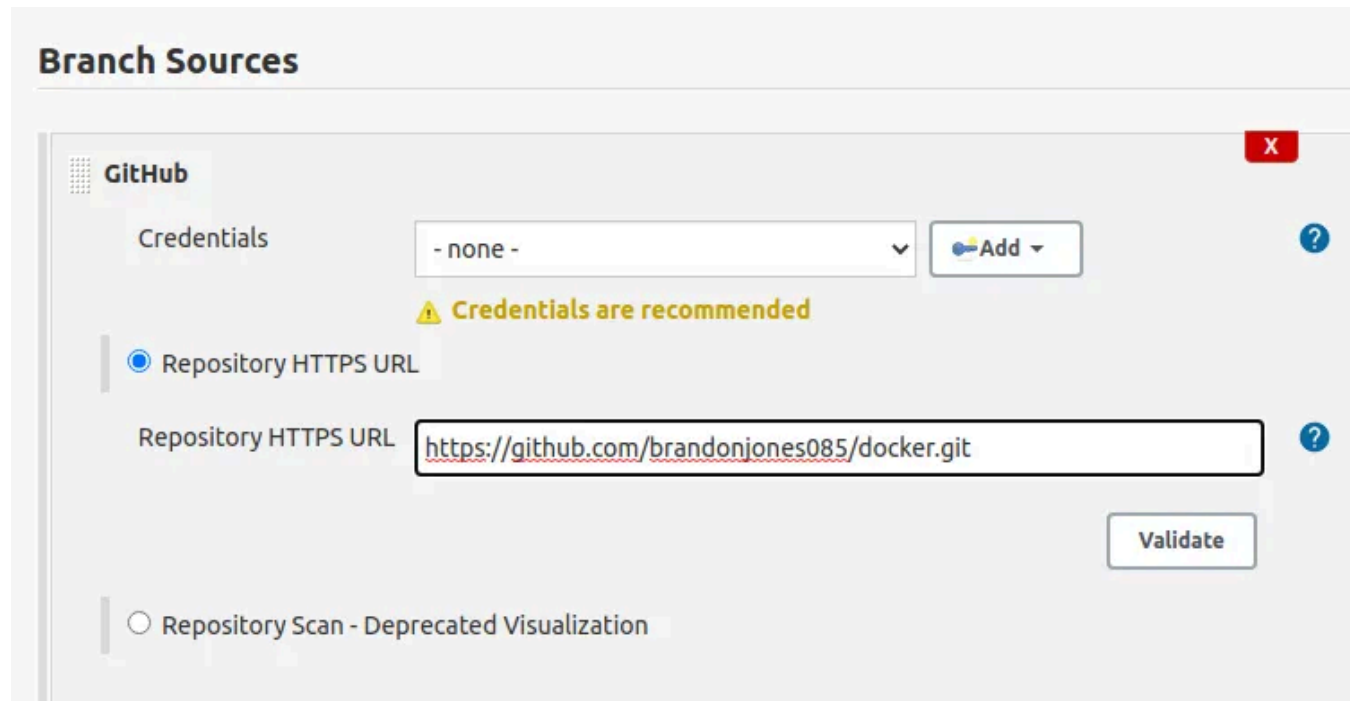
[Plain text] [Preview](#)

Disable: ☐ (No new builds within this Multibranch Pipeline will be executed until it is re-enabled)

General Tab

Under Branch Sources, enter the Github repo URL and click Validate. Since this is a public repo, you won't need to add any credentials, but if you're

using a private repo, you will need the credentials.



The screenshot shows the 'Branch Sources' configuration page in Jenkins. The 'GitHub' provider is selected. The 'Credentials' dropdown is set to '- none -', with an 'Add' button and a warning message 'Credentials are recommended'. The 'Repository HTTPS URL' is set to 'https://github.com/brandonjones085/docker.git'. A 'Validate' button is present. The 'Repository Scan - Deprecated Visualization' option is also visible.

**Branch Sources**

**GitHub**

Credentials: - none - [Add] ?

⚠ Credentials are recommended

☒ Repository HTTPS URL

Repository HTTPS URL: https://github.com/brandonjones085/docker.git ?

[Validate]

☐ Repository Scan - Deprecated Visualization

Branch Sources Tab

Under Build Configuration leave the default Jenkinsfile because this will look for the Jenkinsfile in the cloned repo.

### Build Configuration

Mode

Script Path  [?](#)

Build Configuration

That's it, click save and the project should begin running immediately.

The project will take a few minutes to run, but the initial output should look similar to mine.



## Scan Repository Log

Started

[Tue Nov 03 21:40:09 EST 2020] Starting branch indexing...

21:40:09 Connecting to <https://api.github.com> with no credentials, anonymous access

21:40:09 Jenkins-Imposed API Limiter: Current quota for Github API usage has 59 remaining (9 under budget). Next quota of 60 in 54 min

Examining [brandonjones085/docker](#)

Checking branches...

21:40:10 Jenkins-Imposed API Limiter: Current quota for Github API usage has 58 remaining (8 under budget). Next quota of 60 in 54 min

Getting remote branches...

Checking branch [master](#)

21:40:10 Jenkins-Imposed API Limiter: Current quota for Github API usage has 58 remaining (8 under budget). Next quota of 60 in 54 min

Getting remote pull requests...

'Jenkinsfile' found

Met criteria

Scheduled build for branch: master

21:40:10 Jenkins-Imposed API Limiter: Current quota for Github API usage has 58 remaining (8 under budget). Next quota of 60 in 54 min

Checking branch [new-feature](#)

'Jenkinsfile' found

Met criteria

Scheduled build for branch: new-feature

21:40:11 Jenkins-Imposed API Limiter: Current quota for Github API usage has 56 remaining (6 under budget). Next quota of 60 in 54 min

2 branches were processed

Checking pull-requests...

0 pull requests were processed

Finished examining [brandonjones085/docker](#)

[Tue Nov 03 21:40:11 EST 2020] Finished branch indexing. Indexing took 1.6 sec

Finished: SUCCESS

Scan Repository Log

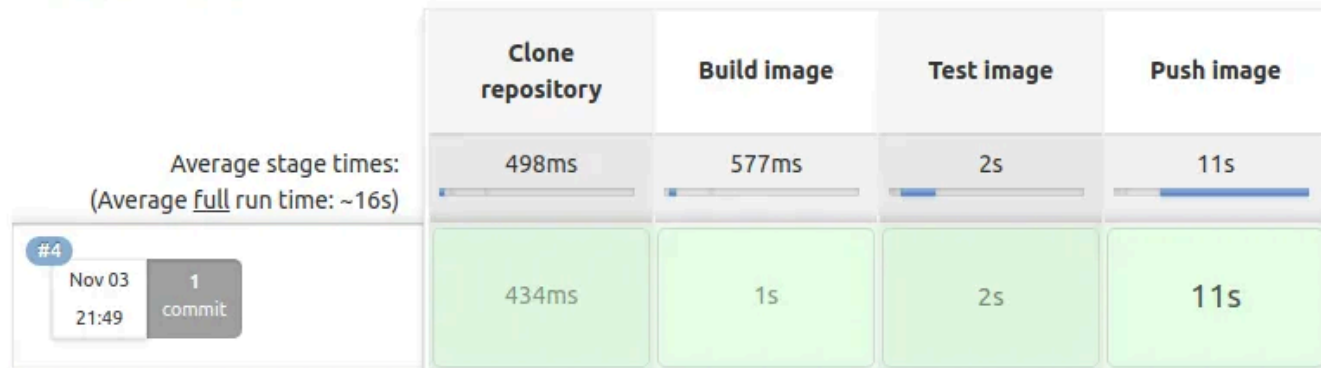
# Branch master

Full project name: docker-pipeline/master



Recent Changes

## Stage View



You can find the entire finished project [here](#). It really is very simple to implement after you've finished the project once to see how everything works together. Everything here should be working, but we all know how that goes. If you run into any issues, feel free to reach out and I'll try to help you work through the problem.

Automate the planet!

-Brandon



@brandon\_jones08

Jenkins

Docker

Dockerhub

Jenkins Pipeline

Jenkinsfile



## Written by Brandon Jones

43 Followers · Writer for The Startup

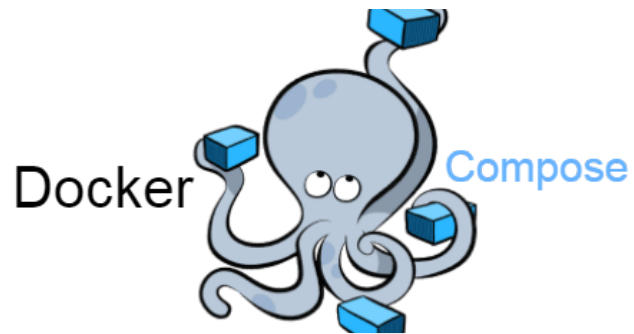
SysAdmin, Devops, Containers, Networks, Automation, Fiddle, Banjo, Pups

Follow



---

More from Brandon Jones and The Startup

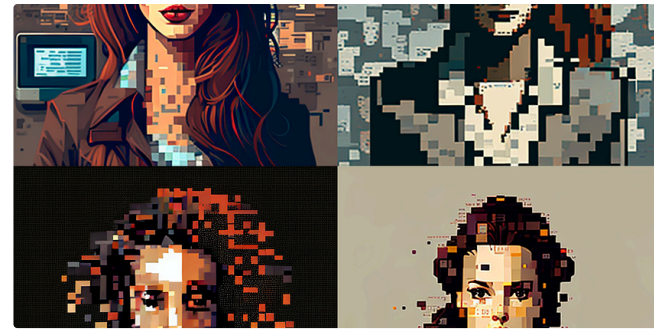



 Brandon Jones

## MEAN Application and Docker Compose

Docker compose can be a really powerful tool. Typically when I'm getting ready to...

Nov 12, 2020



 Zulie Rane in The Startup

## If You Want to Be a Creator, Delete All (But Two) Social Media Platforms

In October 2022, during the whole Elon Musk debacle, I finally deleted Twitter from my...



Apr 19, 2023

 51K


 1228



 Jano le Roux in The Startup

## How to Get a Secret Verified Email Badge (That Almost No One Has)



 Greyson Ferguson in The Startup

## Countries You Can Move To Indefinitely Without a Visa

Stop landing in that damn SPAM folder.

★ Aug 21 🖱 1.1K 💬 26



Why deal with all of that visa paperwork when you could just move without filling out a...

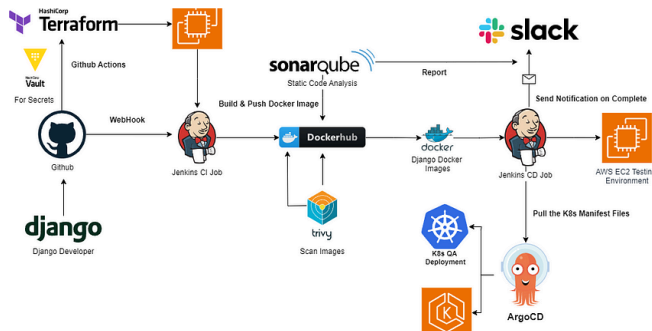
★ Jul 4 🖱 7.1K 💬 86



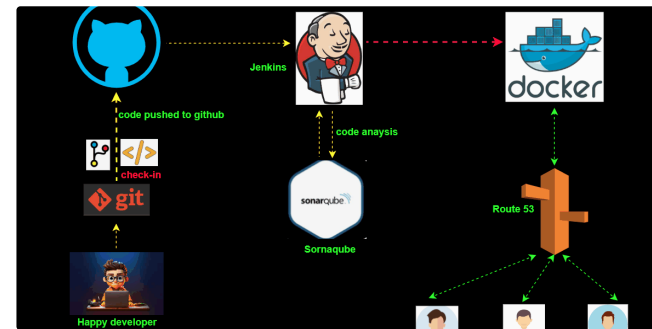
See all from Brandon Jones

See all from The Startup

## Recommended from Medium



Joel Wembo in Django Unleashed



Victor wasonga onyango

## Technical Guide: End-to-End CI/CD DevOps with Jenkins, Docker,...

Building an end-to-end CI/CD pipeline for Django applications using Jenkins, Docker,...



Apr 12



642



12



## Building a Robust CI/CD Pipeline with Jenkins, SonarQube, Docker,...

Introduction



Jul 27



1



### Lists



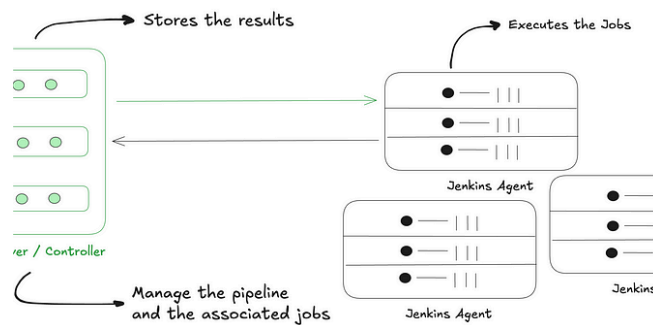
#### Coding & Development

11 stories · 765 saves



#### Natural Language Processing

1670 stories · 1250 saves



Mahedi Hasan Jisan

## Jenkins Architecture and Pipelines



Sapna Yadav

## JFROG Artifactory : Integration with Jenkins .

In software development, Jenkins is an open-source automation server for continuous...



5d ago



2



1



+



Bernardin Houessou

## DevSecOps dotnet pipeline : (Bitbucket, Jenkins, SonarQube,...

DevSecOps : .Net , Jenkins, Sonarqube,  
Checkmarx, Owasp, Docker, K8s, Trivy



Apr 26



3



+

We will be creating Jfrog artifactory in cloud ,  
alternatively we can also download package...

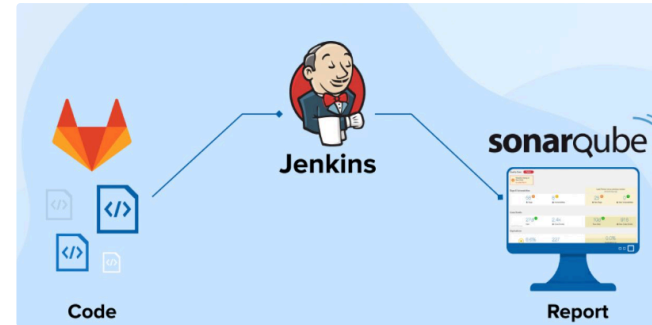
Mar 23



6



+



Liliane Konissi

## Integrating SonarQube with Jenkins

Introduction:

Mar 18



4



+

See more recommendations

