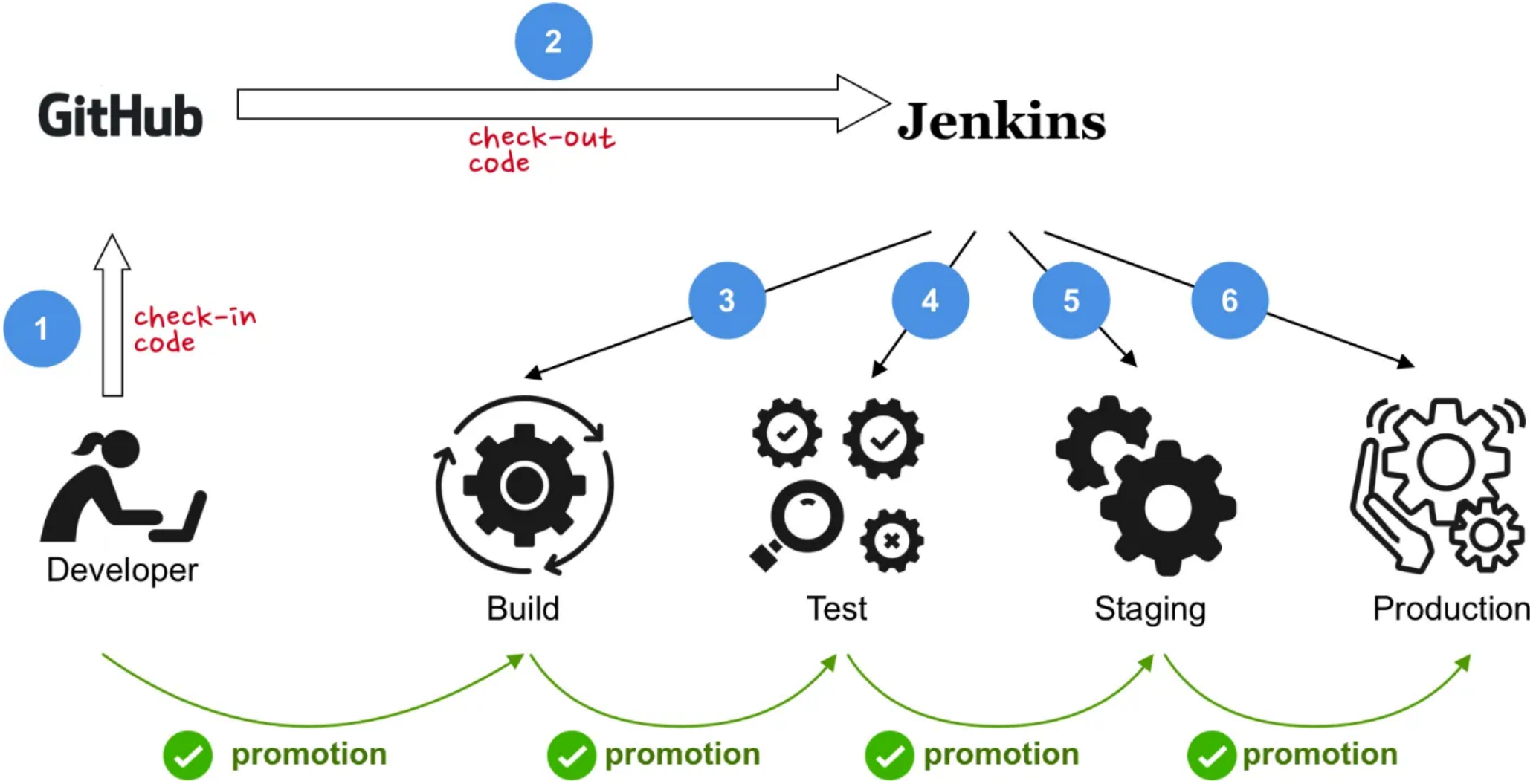
Member-only story

Building CI/CD Pipeline with Jenkins, Kubernetes & GitHub: Part 2

How To Configure Jenkins To Build Your CI CD Pipeline?

@pramodchandrayan · Follow

Published in SysopsMicro · 7 min read · May 24, 2021



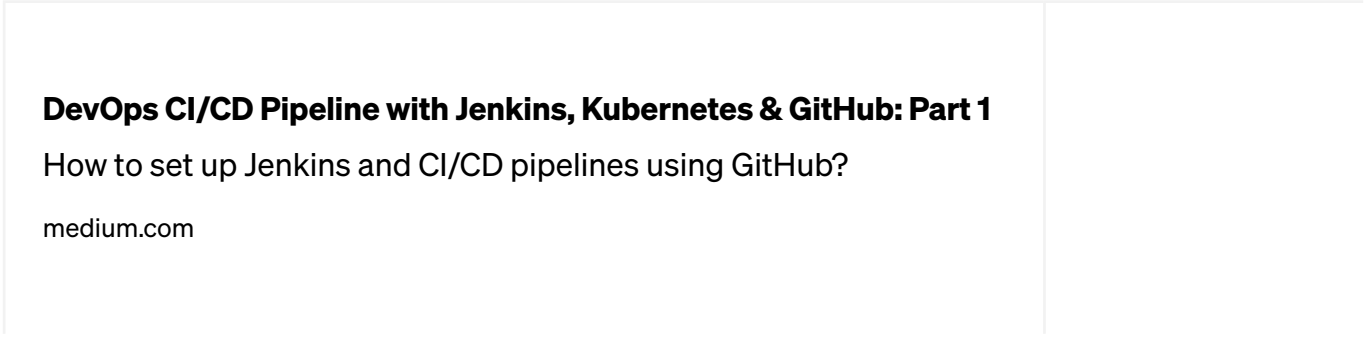
Source

I feel:

> “The world of microservices has lead to the DevOps innovation and this innovation is saving hell lot of time energy and money for the organization adopting it ”

This is in continuation to my first part :

DevOps CI/CD Pipeline with Jenkins, Kubernetes & GitHub



Now we will look deeper into how one can play with Jenkins powerful CI/CD features to automate the continuous integration and continuous deployment of the code repository existent in GitHub under the organization category

We will cover configuring:

- Multibranch Pipeline
- Github Organization pipeline

and see how the CI/CD pipeline is automatically executed with changes in the master branch.

Let's get started with :

How To Configure the GitHub Credentials in Jenkins:

Before we go on to configure the multi-branch pipeline, we need to create some credentials in the Jenkins UI interface, why we need this is because our Jenkinsfile which will be a part of every repo in your organization, required these Credentials to fetch the code form your repo. These credentials

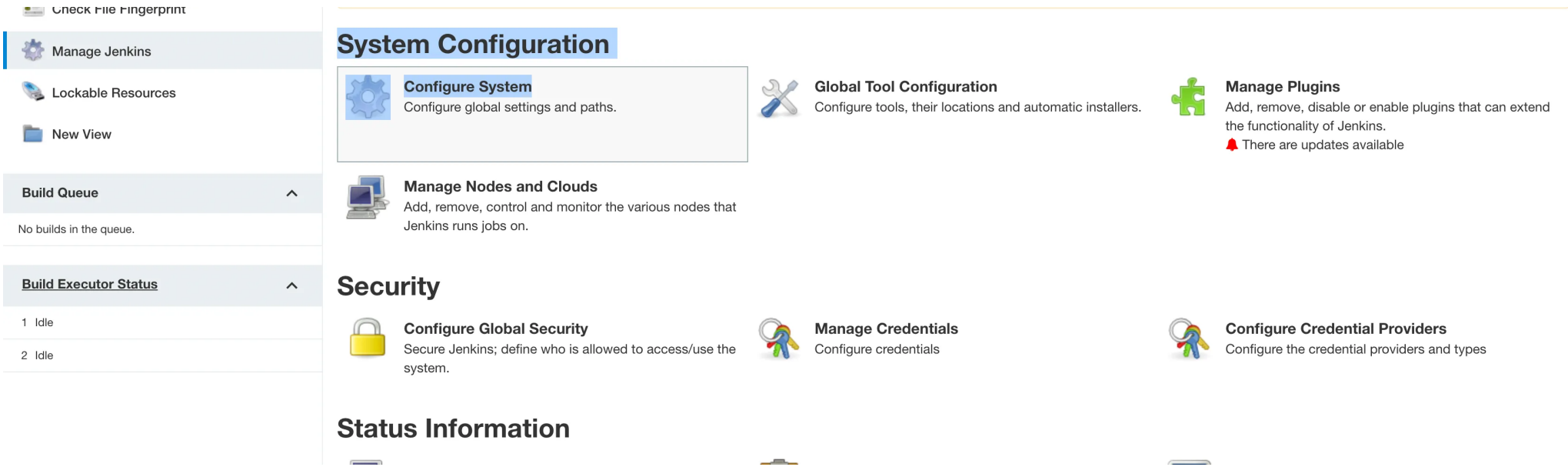
- **ORGANIZATION_NAME** (This should be the name of your Github organization)
- **YOUR_DOCKERHUB_USERNAME**(this can be dummy info), because we are currently not taking the pull from any docker hub, in our sample

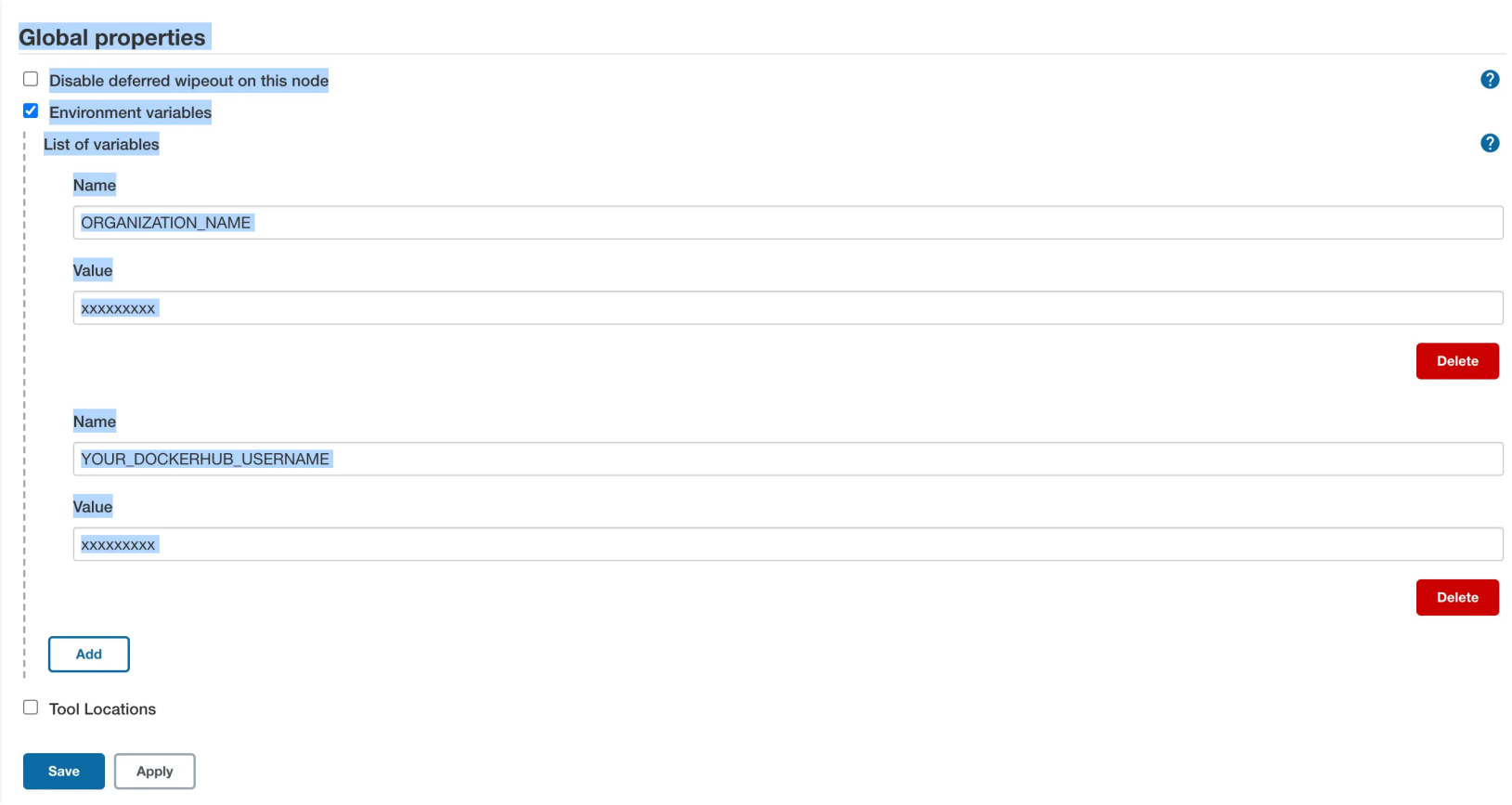
Executable File | 38 lines (33 sloc) | 987 Bytes Raw Blame

```
pipeline {
   agent any

   environment {
     // You must set the following environment variables
     // ORGANIZATION_NAME
     // YOUR_DOCKERHUB_USERNAME (it doesn't matter if you don't have one)

     SERVICE_NAME = "fleetman-webapp"
     REPOSITORY_TAG="${YOUR_DOCKERHUB_USERNAME}/${ORGANIZATION_NAME}-${SERVICE_NAME}:${BUILD_ID}"
   }

   stages {
      stage('Preparation') {
         steps {
            cleanWs()
            git credentialsId: 'GitHub', url: "https://github.com/${ORGANIZATION_NAME}/${SERVICE_NAME}"
         }
      }
      stage('Build') {
         steps {
            sh 'echo No build required for Webapp.'
         }
```

The steps required are as follows:

- Go to the Manage Jenkins menu option
- Click on Configure System


Check File Fingerprint
Manage Jenkins
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
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
Status Information



- Click on Configure System and then go to the **Global properties** section under this, there you will have an option to create **Environment variables**
- Fill ORGANIZATION_NAME & YOUR_DOCKERHUB_USERNAME, as discussed above, then go on to save the credentials

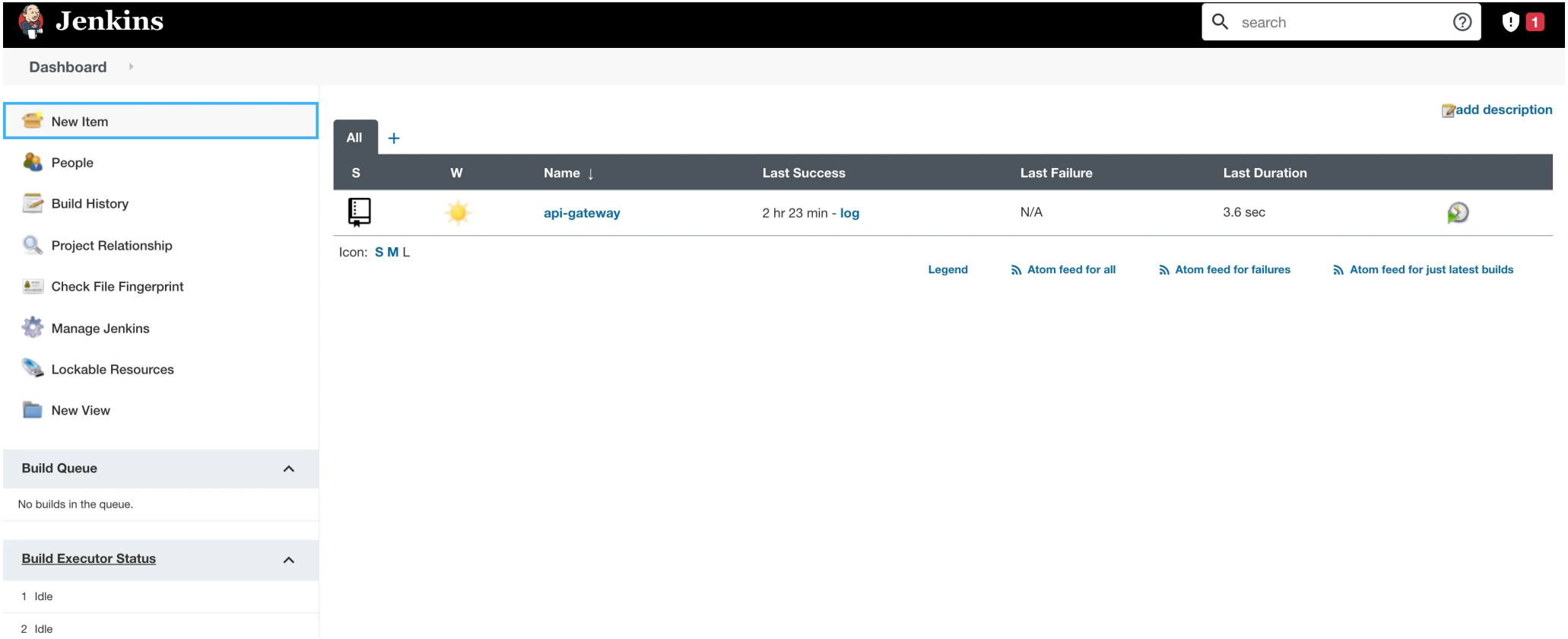


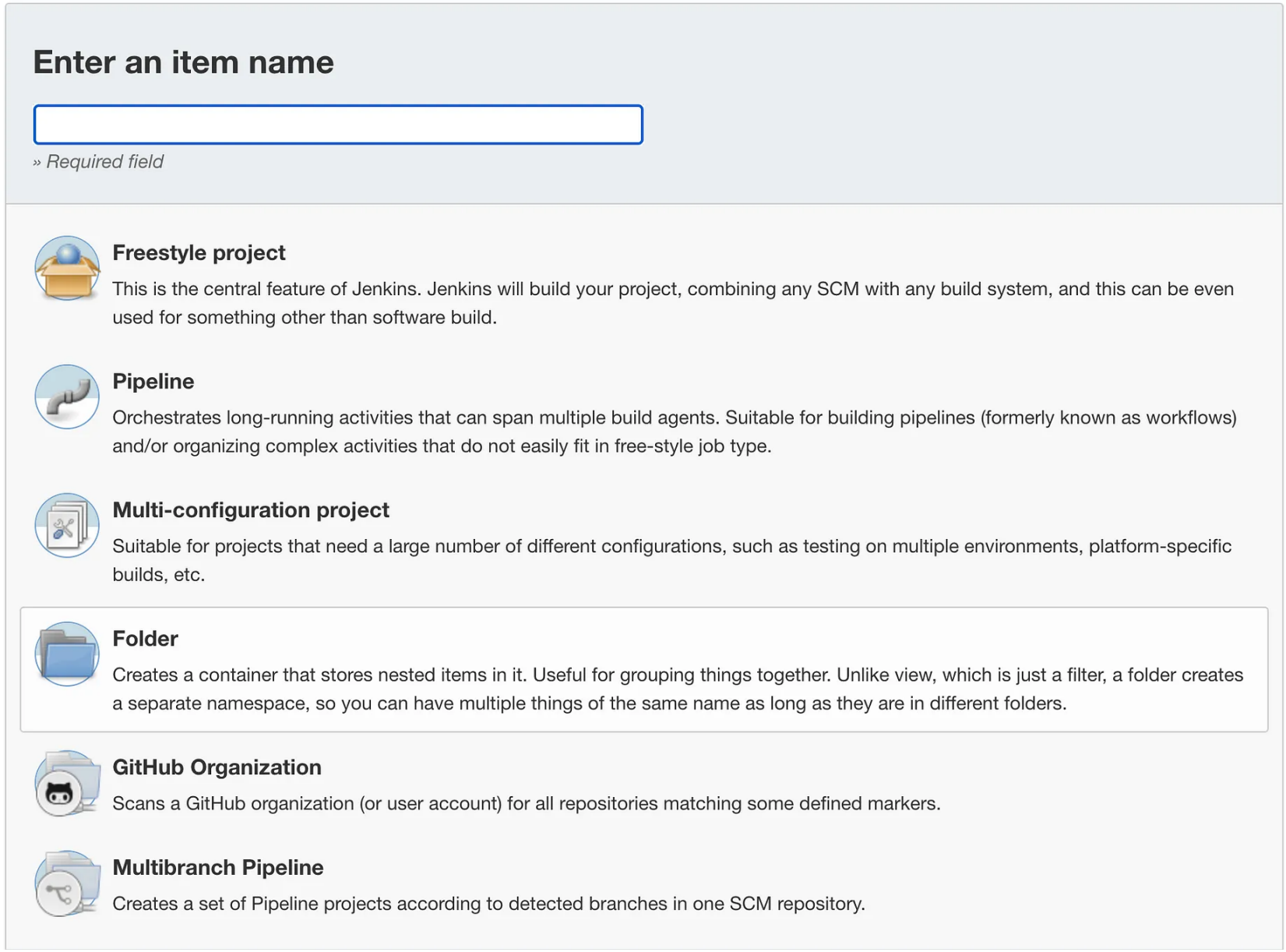
Fig 1.0: Welcome Jenkins page

Multibranch Pipeline Configuration:

In the first part of my Jenkins article, we saw how we can run the Jenkins UI in our web browser, now that you have got the Jenkins UI working as seen in Fig 1.0 above and also got the required environment variable configured, It's time that we start configuring the multi-branch pipeline which will be triggering our CI/CD process for the given repo branch.

Here are the steps for the same:

- Simply click on the “N**ew Item** “ menu tab,


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 even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name 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.

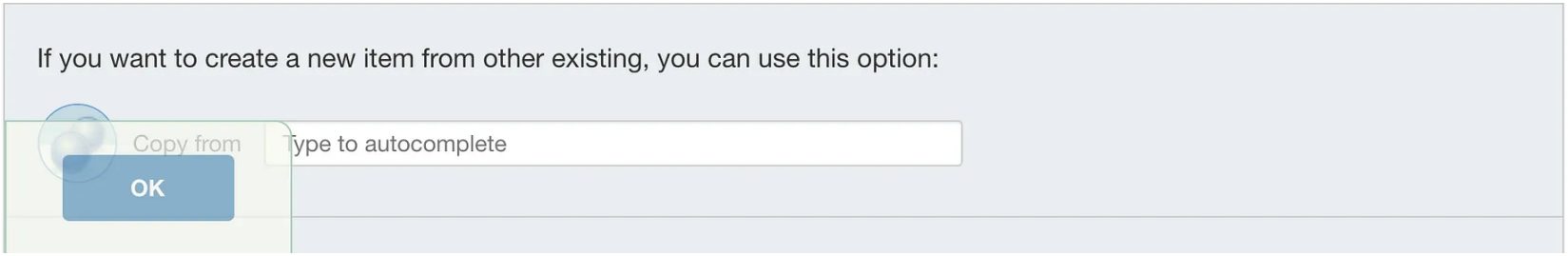


Fig 2.0

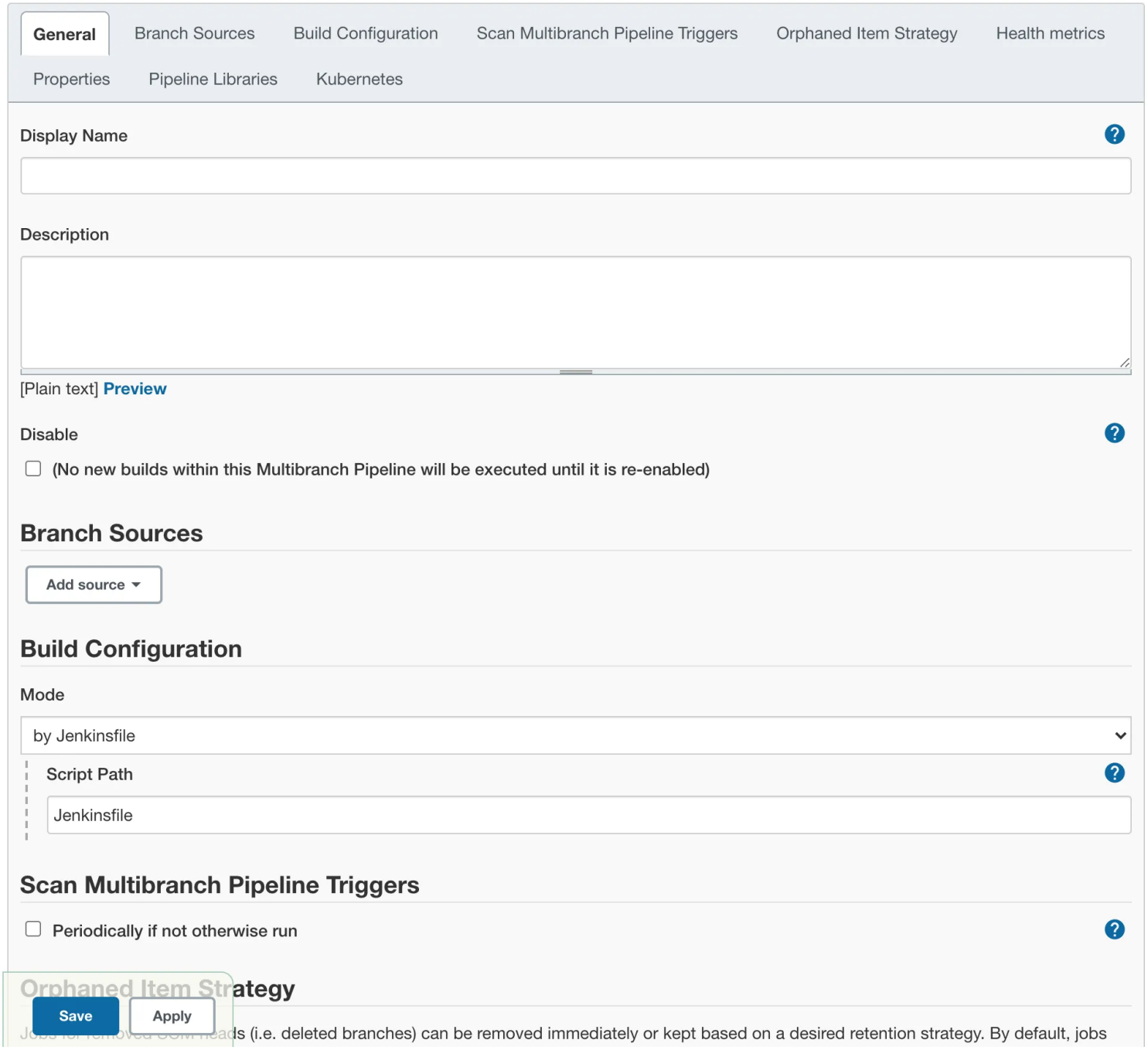
- Enter the repo name as “webapp” in the item name input box shown in the pic above, (you can keep whatever name that suits to your repo), I have kept this because we are going to build CI/CD pipeline to deploy our fleetman-webapp repo as shown below

eazyfin-pramod/fleetman-webapp

This is an experimental project to port the front end to Angular. It’s temporarily on hold, as some very minor problems...

github.com

- Then select Multibranch Pipeline Option and click the save option


General
Branch Sources
Build Configuration
Scan Multibranch Pipeline Triggers
Orphaned Item Strategy
Health metrics
Properties
Pipeline Libraries
Kubernetes
Display Name
Description
[Plain text] Preview
Disable
(No new builds within this Multibranch Pipeline will be executed until it is re-enabled)
Branch Sources
Add source
Build Configuration
Mode
by Jenkinsfile
Script Path
Jenkinsfile
Scan Multibranch Pipeline Triggers
Periodically if not otherwise run
Orphaned Item Strategy
Save
Apply
ds (i.e. deleted branches) can be removed immediately or kept based on a desired retention strategy. By default, jobs

You will then get a screen as shown above, where you need to fill,

- Display name: "**webapp**" (you are free to choose your name), we can skip the description
- Then go to the Branch Configuration section and click on Add source button, once you click the add button, you get a dropdown with some options to choose from, you have to pick Jenkins as we want Jenkins wide global credentials to be applicable across the organization repo.
- Enter your GitHub username and password along with ID name

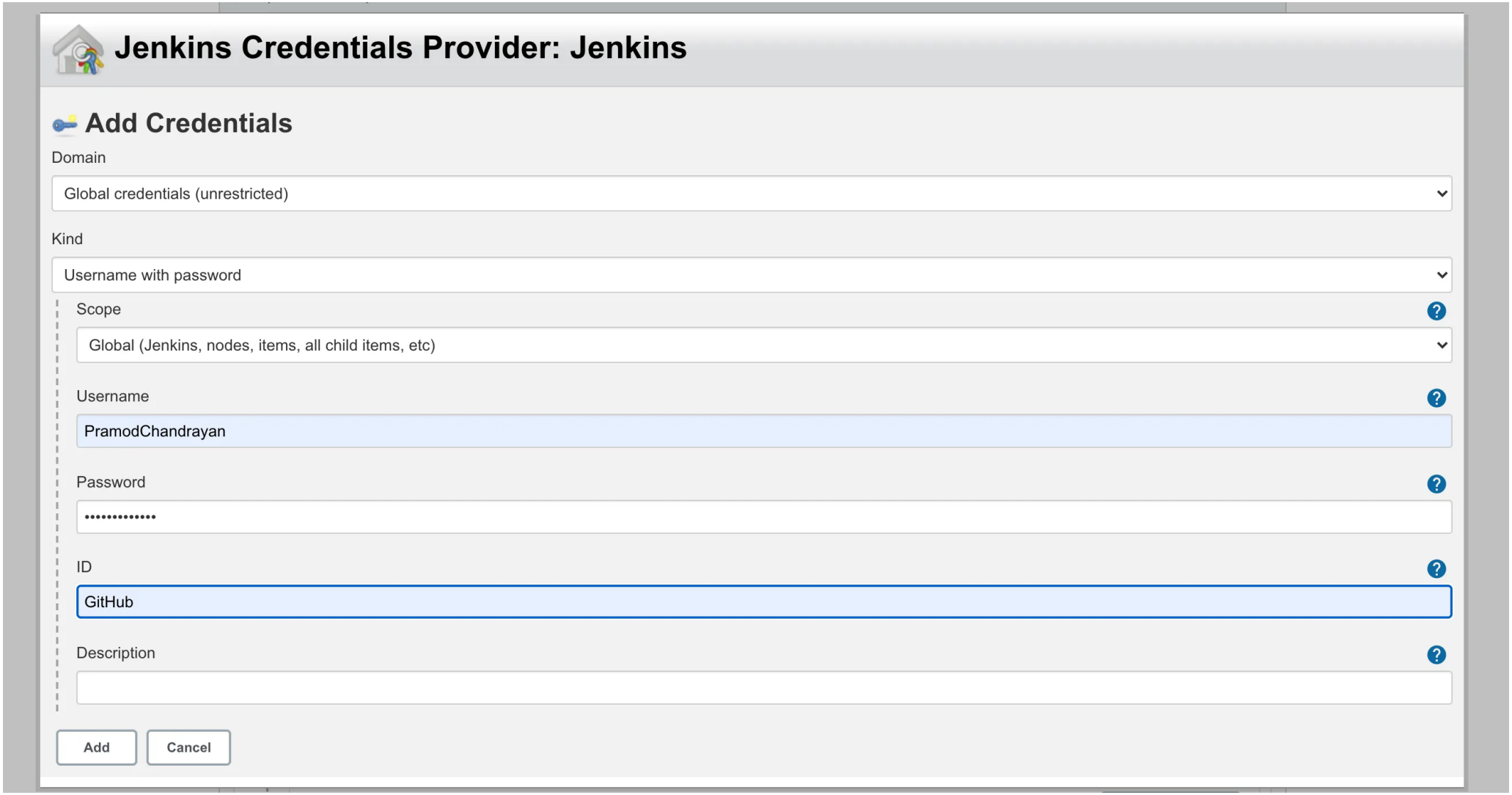


fig: Create Credentials

- Please make sure ID name is "**GitHub**", because as can be seen in our repo Jenkinsfile, the highlighted area below requires your git credentials,

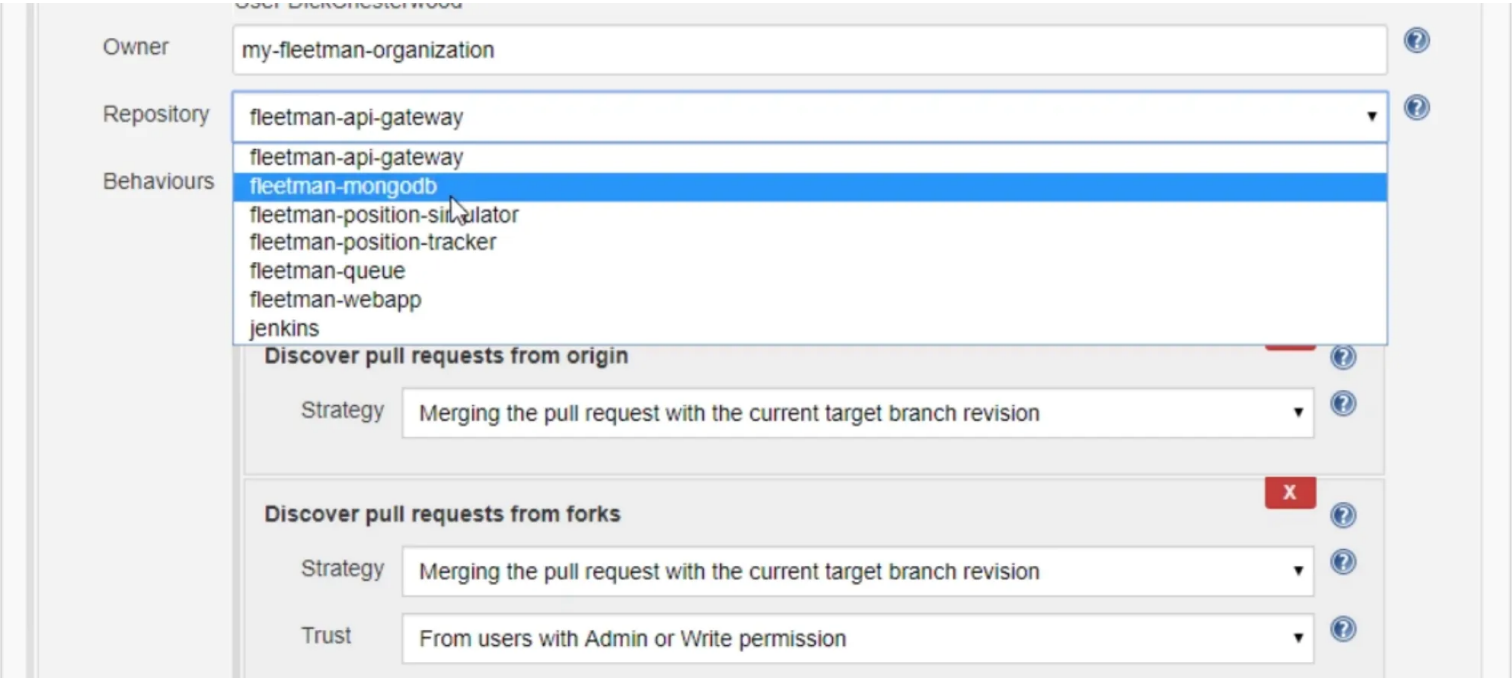
which will look for '**GitHub** ' keyword to fetch the repo. But you are free to choose your name in **Jenkinsfile** and use the same in the Jenkins ' **credentialsId**' config section.

```
pipeline {
   agent any

   environment {
     // You must set the following environment variables
     // ORGANIZATION_NAME
     // YOUR_DOCKERHUB_USERNAME (it doesn't matter if you don't have one)

     SERVICE_NAME = "fleetman-webapp"
     REPOSITORY_TAG="${YOUR_DOCKERHUB_USERNAME}/${ORGANIZATION_NAME}-${SERVICE_NAME}:${BUILD_ID}"
   }

   stages {
      stage('Preparation') {
         steps {
            cleanWs()
            git credentialsId: 'GitHub', url: "https://github.com/${ORGANIZATION_NAME}/${SERVICE_NAME}"
         }
      }
      stage('Build') {
         steps {
            sh 'echo No build required for Webapp.'
         }
      }

      stage('Build and Push Image') {
         steps {
           sh 'docker image build -t ${REPOSITORY_TAG} .'
         }
      }

      stage('Deploy to Cluster') {
          steps {
            sh 'envsubst < ${WORKSPACE}/deploy.yaml | kubectl apply -f -'
          }
```

- Then you can save the credentials by clicking on add button as shown in the '**fig: Create Credentials' image above.**

This will create the credentials against the Tag : **GitHub**, which has to be chosen from add source options as shown below, here you will get the drop-down with your GitHub username flashing, Once you are done selecting, you will be asked to enter,

- **Owner name:** which should be the name of your organization, as soon you enter the Owner name, you will be able to view the list of repo's attached with your Github organization. Select fleetman-web-app, repo, and save your configuration.

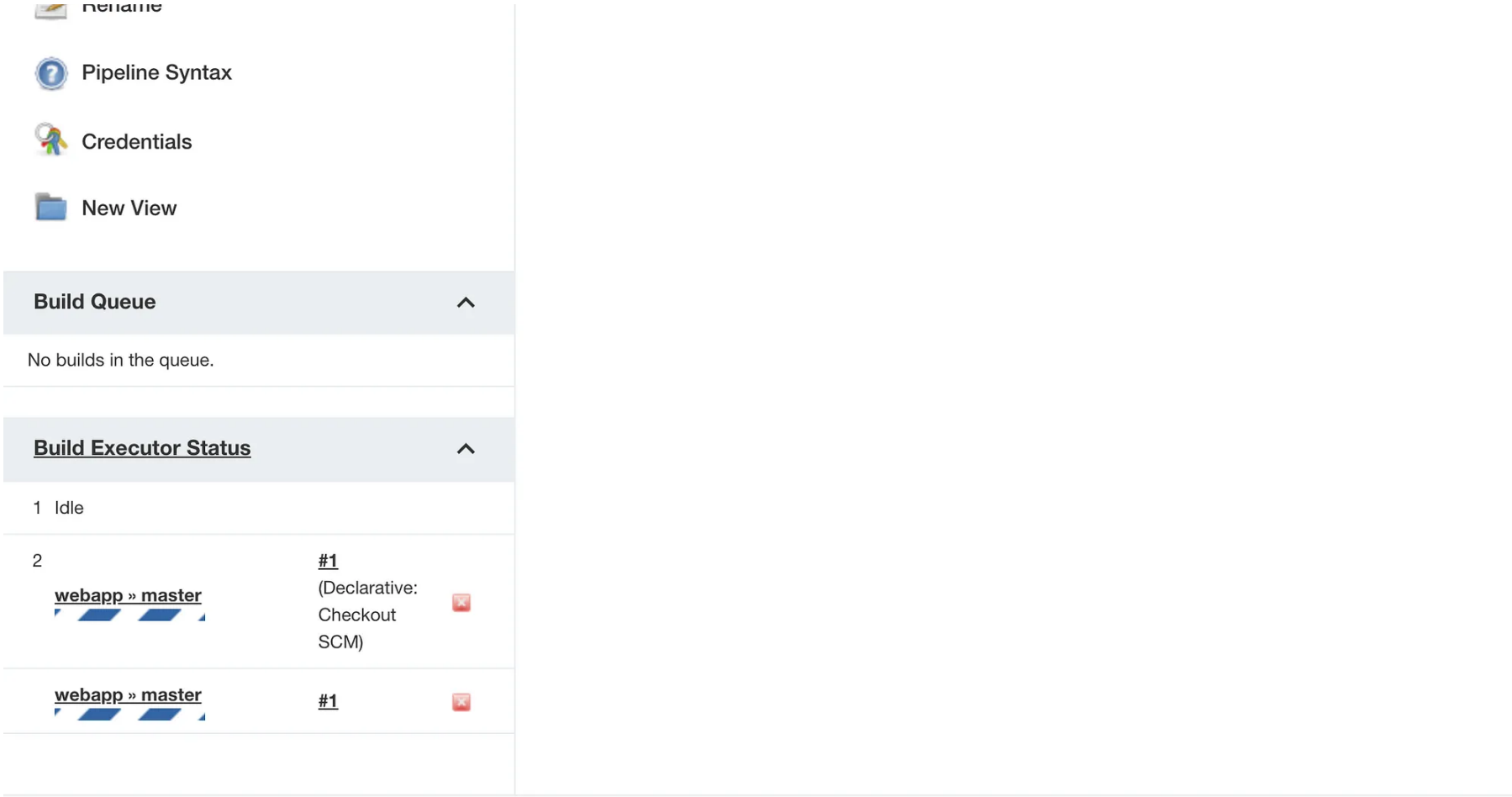


Magic Begins: Your Pipeline Will start building

Once you press the save button, magic will start, your CI/CD pipeline against the web app repo will start building, shown below

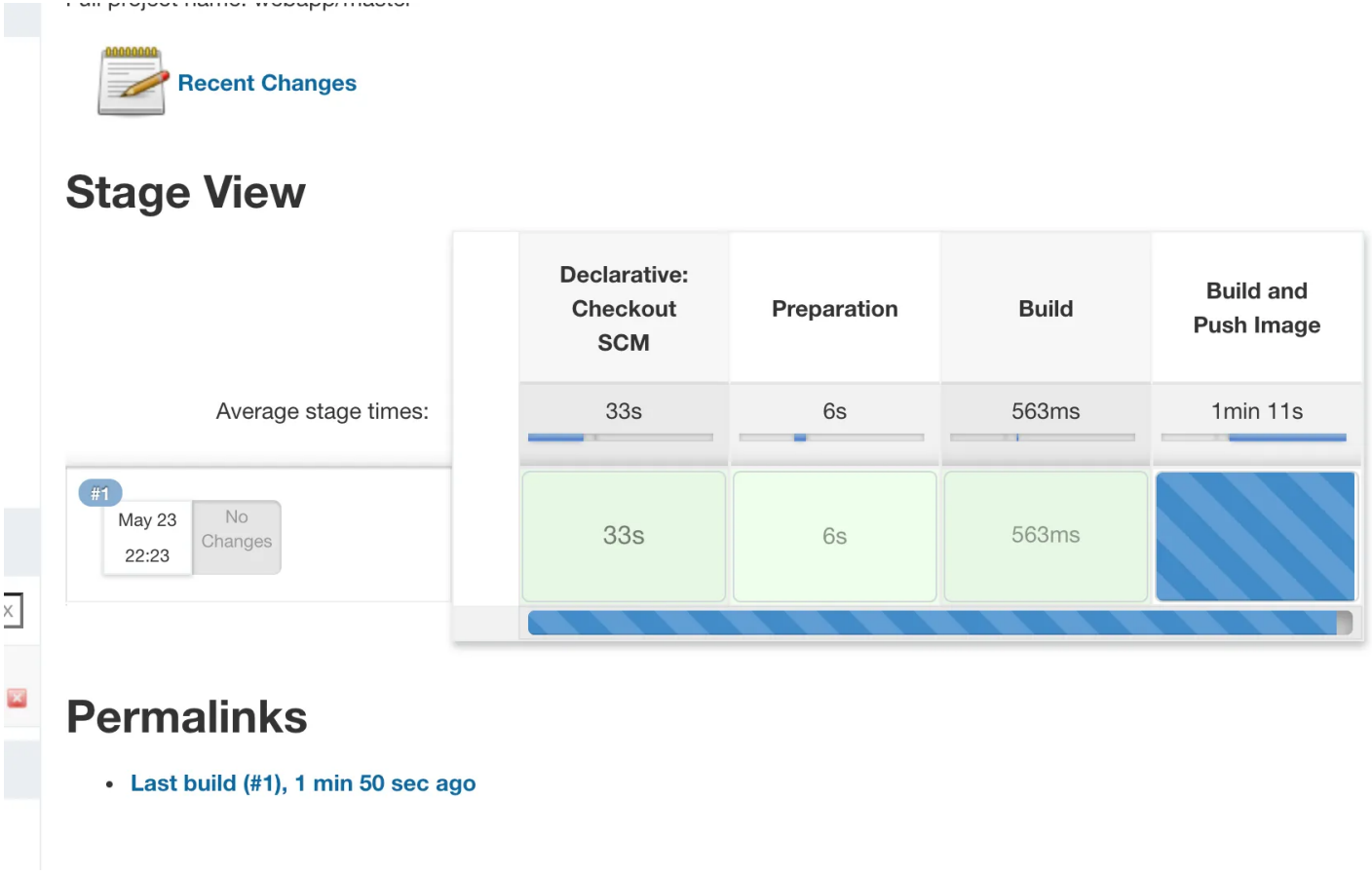
Scan Repository Log

```
Started
[Sun May 23 16:53:29 GMT 2021] Starting branch indexing...
16:53:30 Connecting to https://api.github.com using PramodChandrayan/******
Examining eazyfin-pramod/fleetman-webapp

  Checking branches...

  Getting remote branches...

    Checking branch master

  Getting remote pull requests...
      'Jenkinsfile' found
    Met criteria
Scheduled build for branch: master

  1 branches were processed

  Checking pull-requests...

  0 pull requests were processed

Finished examining eazyfin-pramod/fleetman-webapp

[Sun May 23 16:53:32 GMT 2021] Finished branch indexing. Indexing took 3 sec
Finished: SUCCESS
```



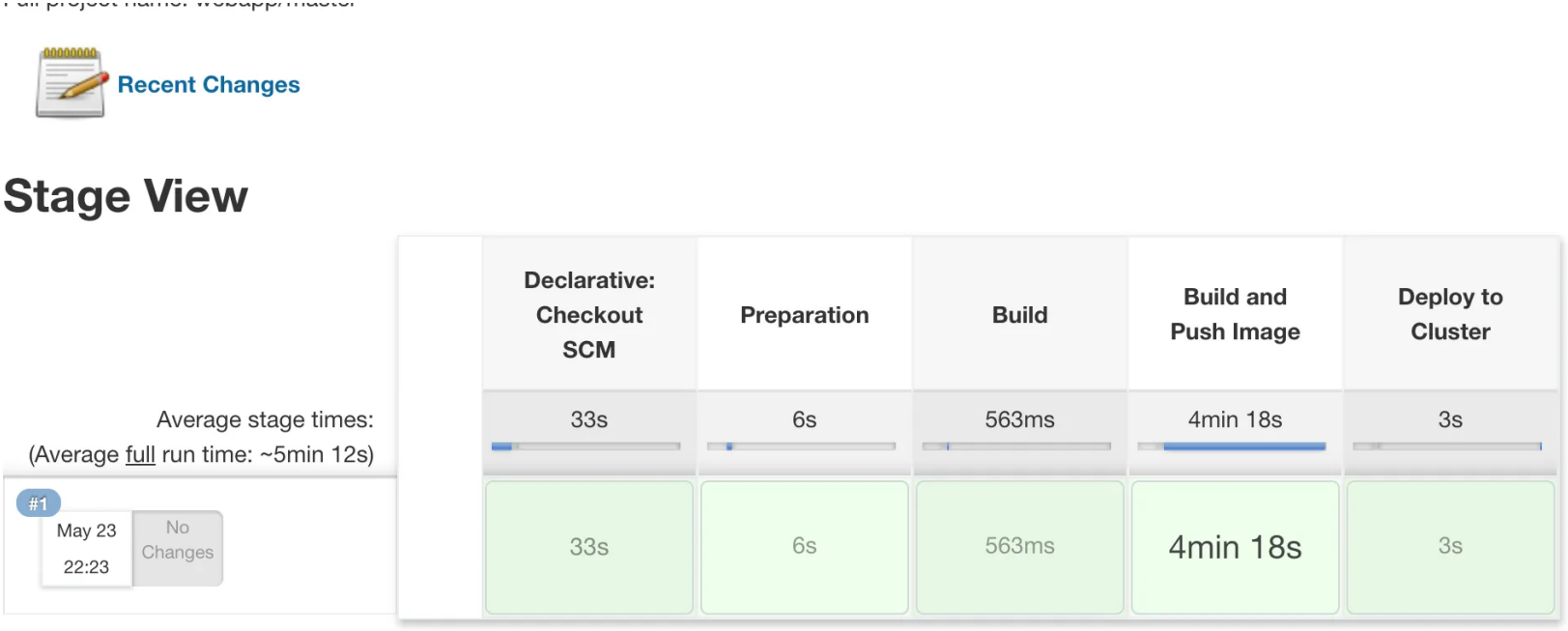
your jenkinsfile configuration.

webapp Pipline Building



Congratulations: You are almost there

Your pipeline building will take some time, so you can relax or have some coffee in between. If your pipeline gets built successfully without any error, you will be able to see that every stage is shown in green



In my case, webapp pipeline has been successfully built and deployed in our Minikube K8s cluster,

Validate If deployment is Successful or Not:

Just check minikube status, to see if minikube is active or not.

```
$ minikube status
```

If not active: Type the below-given command

```
$ minikube start — memory 4028
```

Then type: kubectl get all , command as shown below

```
$ kubectl get all
```

```
zsh: command not found: kubetcl
pramodchandrayan@Pramods-MacBook-Pro jenkins % kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/api-gateway-766545bc6-rx9kf     1/1     Running   0          11h
pod/jenkins-799666d8db-x4kn7        1/1     Running   2          22h
pod/webapp-5978c45565-zjtvr         1/1     Running   0          4m5s

NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                          AGE
service/fleetman-api-gateway      NodePort    10.105.58.32     <none>        8080:30020/TCP                   20h
service/fleetman-webapp           NodePort    10.105.116.110   <none>        80:30080/TCP                     4m5s
service/jenkins                   NodePort    10.108.87.66     <none>        8080:31000/TCP,50000:32251/TCP   22h
service/kubernetes                ClusterIP   10.96.0.1        <none>        443/TCP                          26h

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/api-gateway       1/1     1            1           20h
deployment.apps/jenkins           1/1     1            1           22h
deployment.apps/webapp            1/1     1            1           4m5s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/api-gateway-58677b968c       0         0         0       20h
replicaset.apps/api-gateway-766545bc6        1         1         1       11h
replicaset.apps/jenkins-799666d8db           1         1         1       22h
replicaset.apps/webapp-5978c45565            1         1         1       4m5s
pramodchandrayan@Pramods-MacBook-Pro jenkins %
```

Output:

- You can see from the highlighted area that, our webapp deployment is up and running.
- Our service “**fleetman-webapp**” is also up and running on the NodePort 30080

Running the webapp deployment:

Get your local minikube cluster IP:

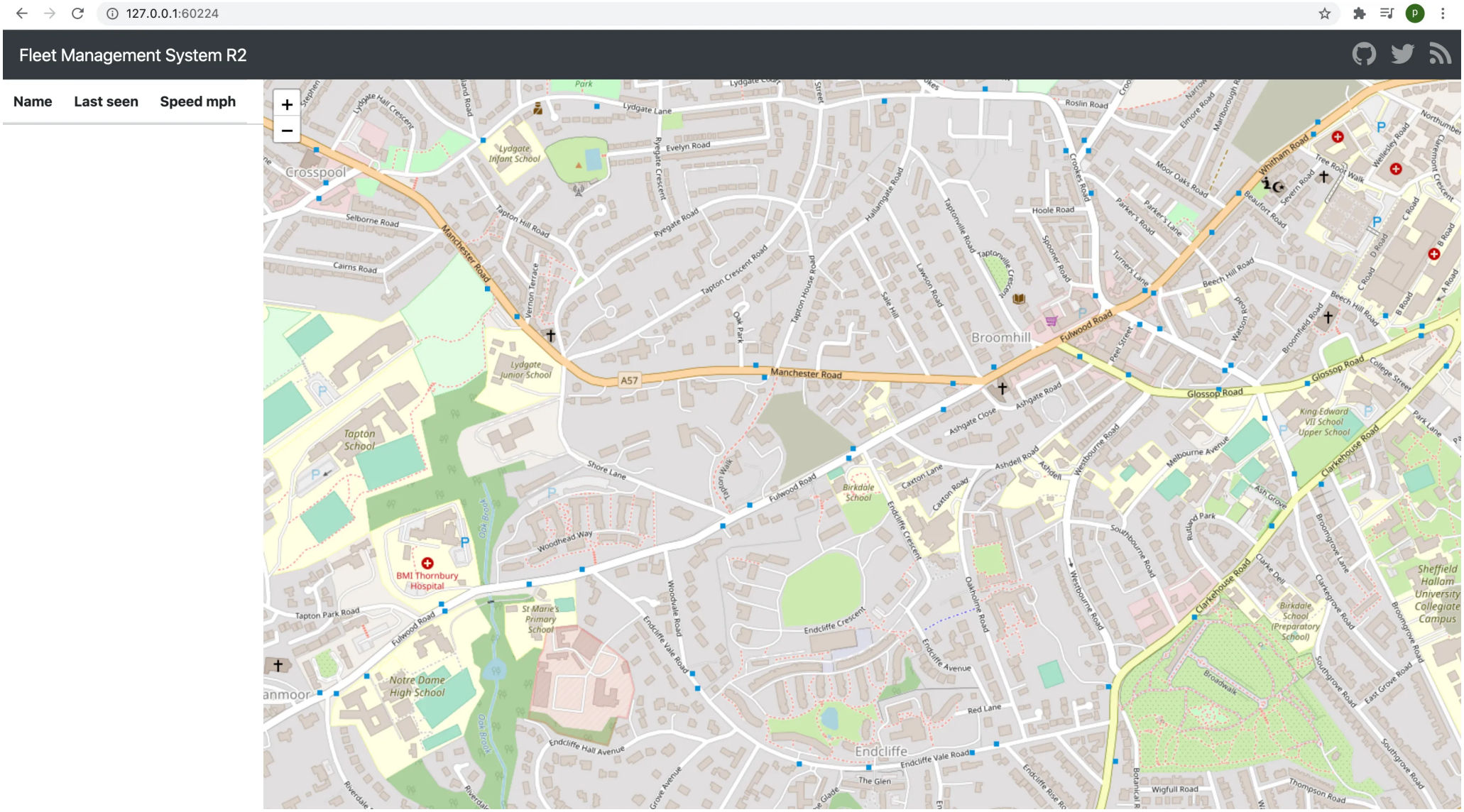
```
$ minikube ip
```

As can be seen, for me my cluster IP is : **192.168.49.2**

Open the web and type this URL given below , along with your fleetman-webapp NodePort number: 30080

“**192.168.49.2:30080**”, on your web browser

If everything goes fine you will be able to view the fleetman-webapp web page as shown below



What If Your URL is not working?

It may happen if you are using the docker driver while starting the minikube

, instead of the virtual box. In this situation minikube IP will not functions and you will be required to start the local tunnel from your terminal using the following commands,

```
$ minikube service jenkins --url
```

```
replicaset.apps/webapp-5978c45565        1         1         1       4m5s
pramodchandrayan@Pramods-MacBook-Pro jenkins % minikube service fleetman-webapp --url
🏃  Starting tunnel for service fleetman-webapp.
|-----------|-----------------|-------------|-----------------------|
| NAMESPACE |      NAME       | TARGET PORT |          URL          |
|-----------|-----------------|-------------|-----------------------|
| default   | fleetman-webapp |             | http://127.0.0.1:60224 |
|-----------|-----------------|-------------|-----------------------|
http://127.0.0.1:60224
❗  Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

As can be seen from the above image, a tunnel service at port no : 60224 has been started, now you just need to go to your favorite browser and type

http://127.0.0.1:60224/ , and you should see the webapp , page opening as shown earlier.

Congratulations on getting your microservice deployed on your local

minikube k8s cluster.

What's Next?

We will look into one more powerful option available with Jenkins called **"Github Organizations"** which will allow you to pick all your master branch repo lying under your organization, in one go and trigger the CI/CD pipeline to build and deploy them in one go.

Meanwhile don't forget to view part 1 of the Jenkins CI/CD auto-deployment series

DevOps CI/CD Pipeline with Jenkins, kubernetes & GitHub: Part 1
How to set up Jenkins and CI/CD pipelines using GitHub?
medium.com

And my other contributions related to Kubernetes:

Kubernetes Fundamentals For Absolute Beginners: Architecture & Components

Learning Kubernetes architecture & components

medium.com

Working With Deployment In Kubernetes

What is Deployment in Kubernetes? How to maintain a rolling update of our application ?

medium.com

and follow me for more informative articles on

- Kubernetes
- Crypto
- DevOps and much more...

Note!

This sample repo has been forked from **chesterwood** and I am highly thankful to him for creating such an awesome CI/CD ready repository.

Thanks a ton for being there with me and inspiring me to write more and learn more...

Microservices DevOps Kubernetes Docker Cloud Computing

Published in SysopsMicro

366 Followers · Last published Nov 25, 2021

Follow

We share our knowledge regarding scalable architectures, clean coding, DevOps, CI/CD to help you learn and grow fast

Written by @pramodchandrayan

1.4K Followers · 27 Following

Follow

Building @krishaq: an Agritech startup committed to revive farming, farmers and our ecology | Writes often about agriculture, climate change & technology

Responses (1)

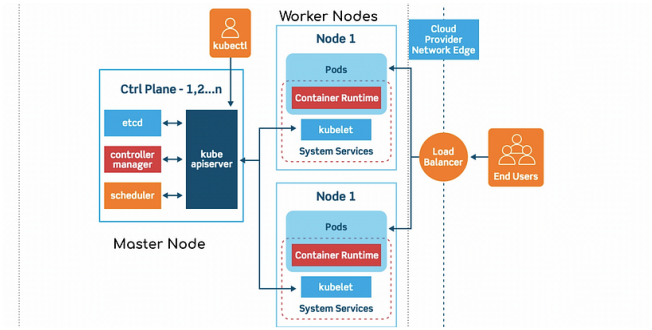
What are your thoughts? Respond

Tian
over 3 years ago

nice

Reply

More from @pramodchandrayan and SysopsMicro

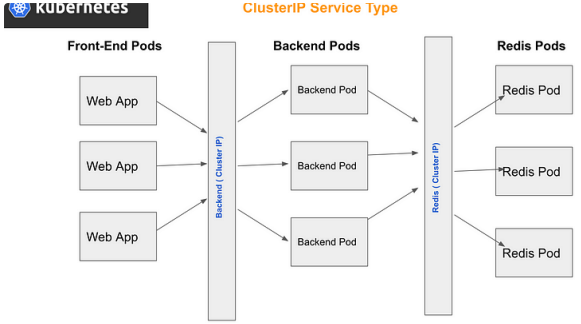


In SysopsMicro by @pramodchandrayan

Kubernetes Fundamentals For Absolute Beginners: Architecture...

Learning Kubernetes architecture & components

Jan 3, 2021 1.3K 9

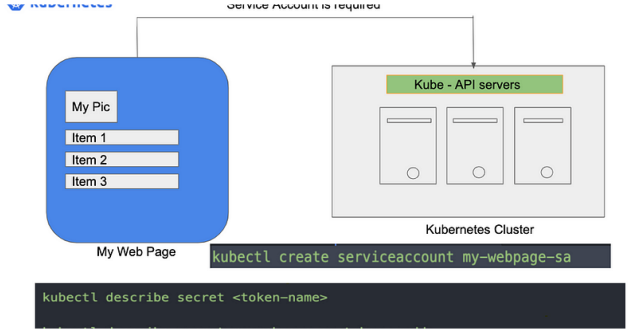


In SysopsMicro by @pramodchandrayan

Working With ClusterIP Service Type In Kubernetes

Working with services in Kubernetes Using ClusterIP

Mar 24, 2021 147 6



In SysopsMicro by @pramodchandrayan

Working with Service Account In Kubernetes

How to configure a service account in Kubernetes and manage it?

Oct 14, 2020 308 2

In SysopsMicro by @pramodchandrayan

AWS EKS Fundamentals: Architecture & Components

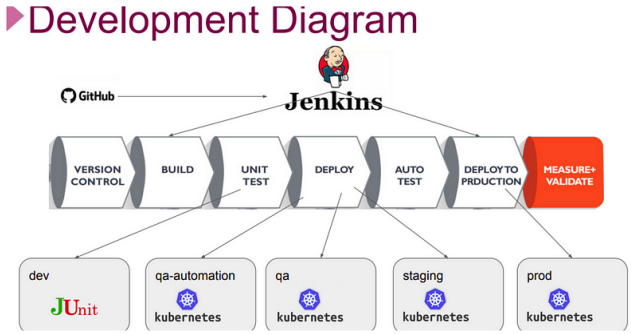
Learn about Managed EKS and Create EKS Cluster using eksctl

Aug 8, 2021 107 2

See all from @pramodchandrayan

See all from SysopsMicro

Recommended from Medium



Adnan Turgay Aydin

Building a Real-Time CI/CD Pipeline: Step-by-Step Guide for...

This DevOps project is designed to enhance your resume and provide hands-on...

Aug 16 22

Evan Selunani

CI/CD With CodePipeline

Learn how to set up a CI/CD pipeline with AWS CodePipeline.

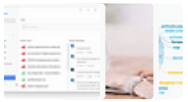
Dec 17 1

Lists

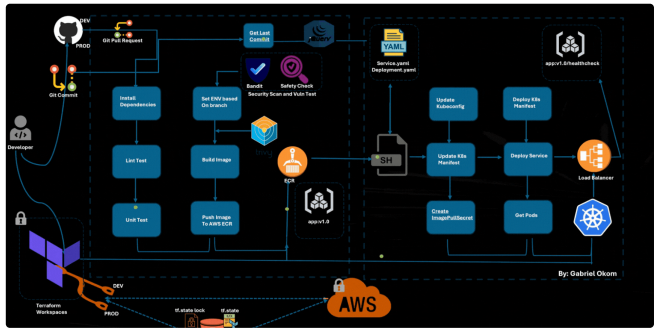
Coding & Development
11 stories · 948 saves

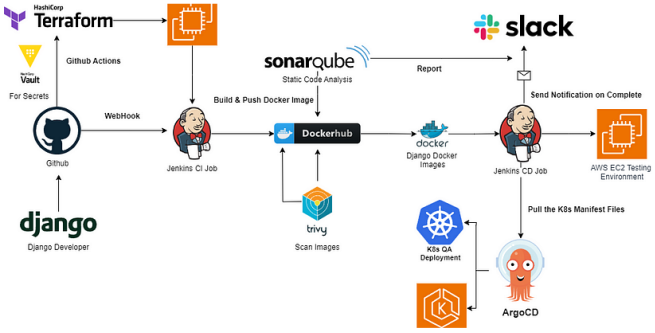
General Coding Knowledge
20 stories · 1833 saves

Natural Language Processing
1879 stories · 1502 saves



Productivity
242 stories · 652 saves

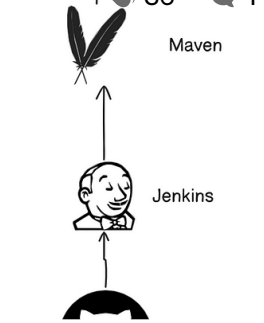




GABRIEL OKOM

CI/CD Pipeline: Deploy Python App with Healthcheck Endpoint on A...

Table of Contents 1. Introduction 2. Prerequisites 3. Tools Used 4. Pipeline...

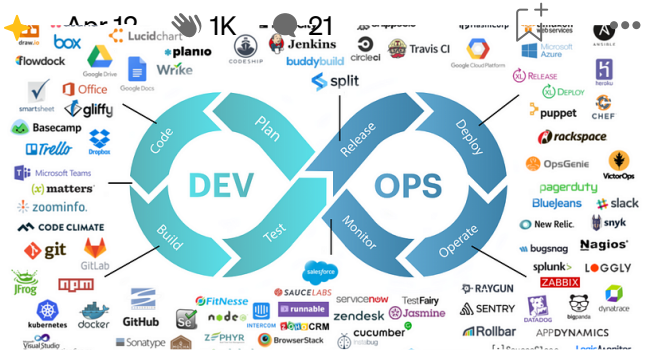


Nov 5 36 1

In Django Unleashed by Joel Wembo

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 19 1K 21

Dacio Bezerra

End-to-End DevOps Project Pipeline

In this article, we will explore an end-to-end DevOps project pipeline using Jenkins that...

Nov 13 36

Harshal Sonawane

50 DevOps Project Ideas: From Beginner to Advanced

DevOps Journey

Nov 15 43 6

See more recommendations