D — DEVOPS

Jenkins Multibranch Pipeline Tutorial For Beginners

by **Bibin Wilson** · August 6, 2020

Jenkins multi-branch pipeline is the best way to automate CI/CD workflows as it is entirely a git-based pipeline as code (Jenkinsfile)

If you are looking for a well-automated Pull Request based or branch-based Jenkins **Continuous Integration** & **Delivery** (CI/CD) pipeline, this guide will help you get the overall picture of how to achieve it using the Jenkins multibranch pipeline.

Jenkins's multi-branch pipeline is one of the best ways to design CI/CD workflows as it is entirely a git-based (source control) pipeline as code. This guide will talk about all the key concepts involved in a Jenkins multi-branch pipeline setup.

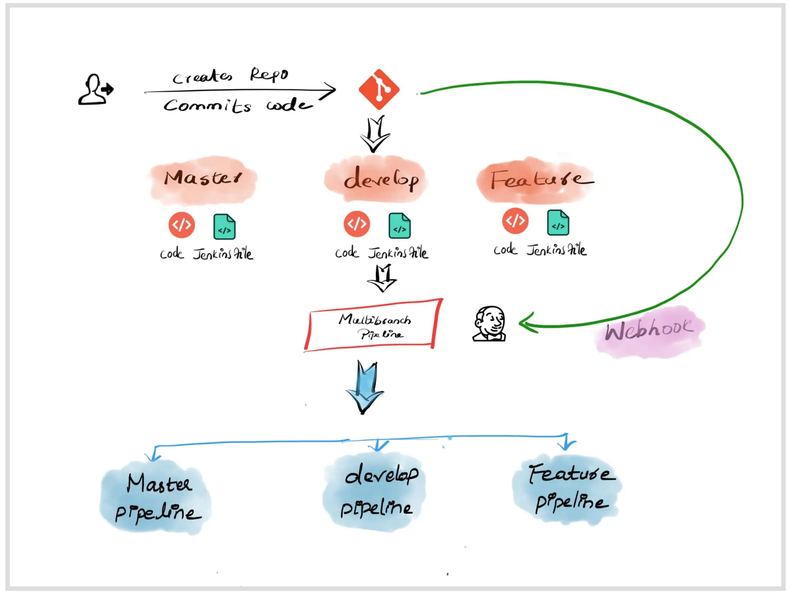
Jenkins Multibranch Pipeline Fundamentals

Let's start with the multi-branch pipeline basics. Specifically, in this section, I will cover the concept of a multi-branch pipeline and why it is essential to use it for all Jenkins CI/CD pipelines. I'll also show you how a multi-branch pipeline works with a detailed workflow diagram.

What is a Multi-branch Pipeline?

(Github) and automatically create a pipeline for that branch. When the pipeline build starts, Jenkins uses the Jenkinsfile in that branch for build stages.

SCM (Source Control) can be Github, Bitbucket, or a Gitlab repo.



You can choose to exclude selected branches if you don't want them to be in the automated pipeline with Java regular expressions.

Multi-branch pipeline supports PR based branch discovery. Meaning, branches get discovered automatically in the pipeline if someone raises a PR (pull request) from a branch. If you have this configuration enabled, builds will get triggered only if a PR is raised. So if you are looking for a PR based Jenkins build workflow, this is a great option.

You can add conditional logic to the Jenkinsfile to build jobs based on the branch requirement.

For example, if you want the feature branch to run only unit testing and sonar analysis, you can have a condition to skip the deployment stage with a when a condition, as shown below.

```
stage('Deploy for production') {
    when {
       branch 'production'
    }
    steps {
     ----
    }
}
```

So whenever the developer raises the PR from the feature branch to some other branch, the pipeline will run the unit testing and sonar analysis stages skipping the deployment stage.

Also, multi-branch pipelines are not limited to the continuous delivery of applications. You can use it to manage your infrastructure code as well.

One such example is having a continuous delivery pipeline for Docker image or a VM image patching, building, and upgrade process.

How Does a Multi-Branch Pipeline work?

I will walk you through a basic build and deployment workflow to understand how a multi-branch pipeline work.

Let's say I want a Jenkins pipeline to build and deploy an application with the following conditions.

- Development starts with a feature branch by developers committing code to the feature branch.
- Whenever a developer raises a PR from the feature branch to develop a branch, a Jenkins pipeline should trigger to run a unit test and static code analysis.
- After testing the code successfully in the feature branch, the developer merges the PR to the develop branch.
- When the code is ready for release, developers raise a PR from the develop branch to the master. It should trigger a build pipeline that will run the unit test cases, code analysis, push artifact, and deploys it to dev/QA environments.

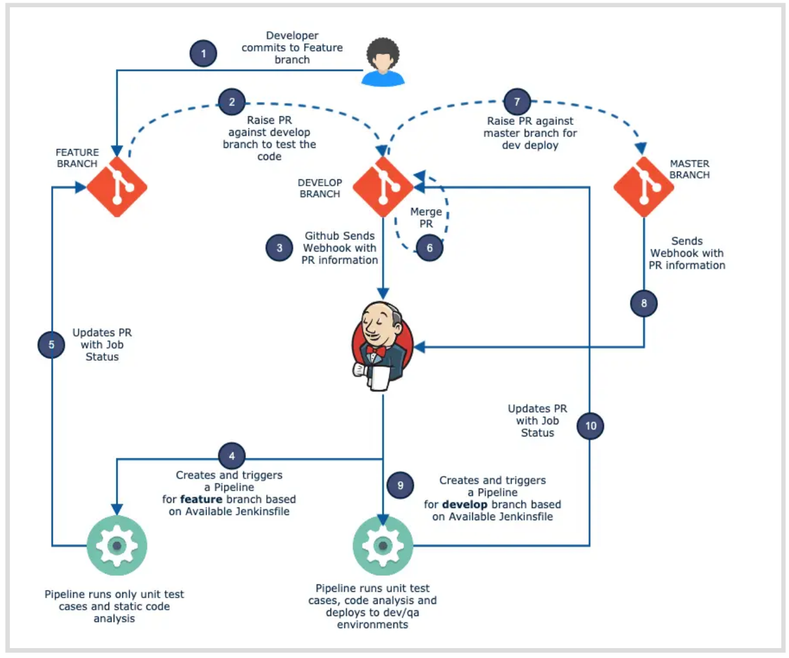
triggered automatically and run the required steps for that branch.

This workflow builds a great feedback loop for engineers and avoids dependence on the DevOps team to build and deploy in non-prod environments.

Developer can check the build status on Github and take decisions on what to do next.

This workflow can be achieved easily through a Jenkins multi-branch pipeline.

The following image shows how a multi-branch pipeline workflow would look like for the above example build process



Here is how the multi-branch pipeline works.

- When a developer creates a PR from a feature branch to develop a branch, Github sends a webhook with the PR information to Jenkins.
- Jenkins receives the PR and finds the relevant multibranch pipeline, and creates a feature branch pipeline automatically. It then runs the jobs with the steps mentioned in the Jenkinsfile from the feature branch. During checkout, the source and target branches in the PR gets merged. The PR merge will be blocked on Github until a build status from Jenkins is returned.
- Once the build finishes, Jenkins will update the status to Github PR. Now you will be able to merge the code. If you want to check the Jenkins build

Multibranch Pipleline Jenkinsfile

Before jumping into implementation, let's look at multibranch pipeline Jenkins example Jenkinsfile that can be used in the pipeline.

For the multibranch pipeline to work, you need to have the Jenkinsfile in the SCM repo.

If you are learning/testing, you can use the multibranch pipeline Jenkinsfile given below. It has a checkout stage and other dummy stages, which echoes the message.

Also, you can clone and use this Github repo which has this Jenkinsfile

> **Note**: Replace the agent label "master" with your Jenkins agent name. master will also work but wouldn't advise it running in actual project environments.

```
pipeline {

    agent {
        node {
            label 'master'
        }
    }

    options {
        buildDiscarder logRotator(
                    daysToKeepStr: '16',
                    numToKeepStr: '10'
            )
    }

    stages {

        stage('Cleanup Workspace') {
            steps {
                cleanWs()
                sh """
                echo "Cleaned Up Workspace For Project"
                """
            }
        }

        stage('Code Checkout') {
            steps {
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: '*/main']],
                    userRemoteConfigs: [[url: 'https://github.com/spring-
projects/spring-petclinic.git']]
                ])
            }
        }
```

```
                sh """
                echo "Running Unit Tests"
                """
            }
        }

        stage('Code Analysis') {
            steps {
                sh """
                echo "Running Code Analysis"
                """
            }
        }

        stage('Build Deploy Code') {
            when {
                branch 'develop'
            }
            steps {
                sh """
                echo "Building Artifact"
                """

                sh """
                echo "Deploying Code"
                """
            }
        }

    }
}
```

Setup Jenkins Multi-branch Pipeline

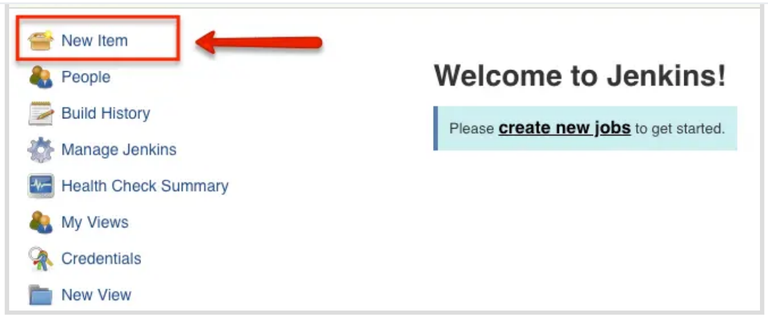
Here I will walk you through the step by step process of setting up a multi-branch pipeline on Jenkins.

This setup will be based on Github and latest Jenkins 2.x version.

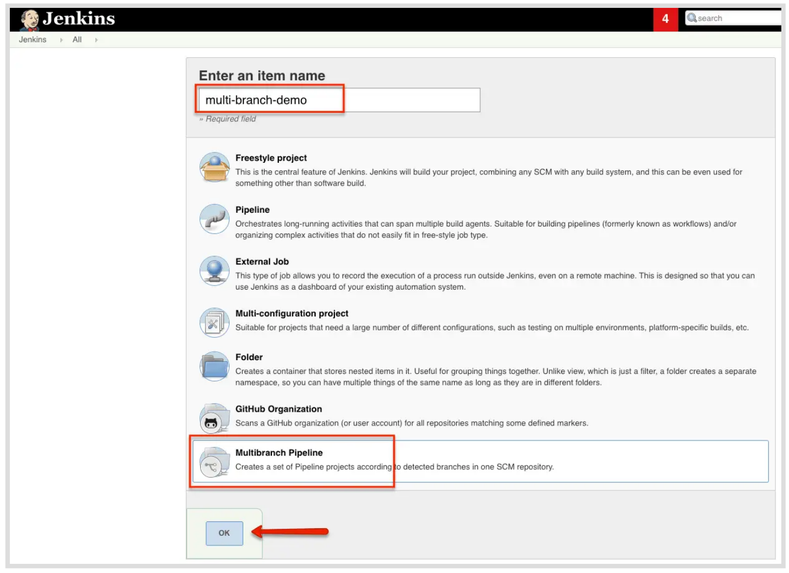
You can also use Bitbucket or Gitlab as SCM source for a multi-branch pipeline

Create Multibranch Pipeline on Jenkins (Step by Step Guide)

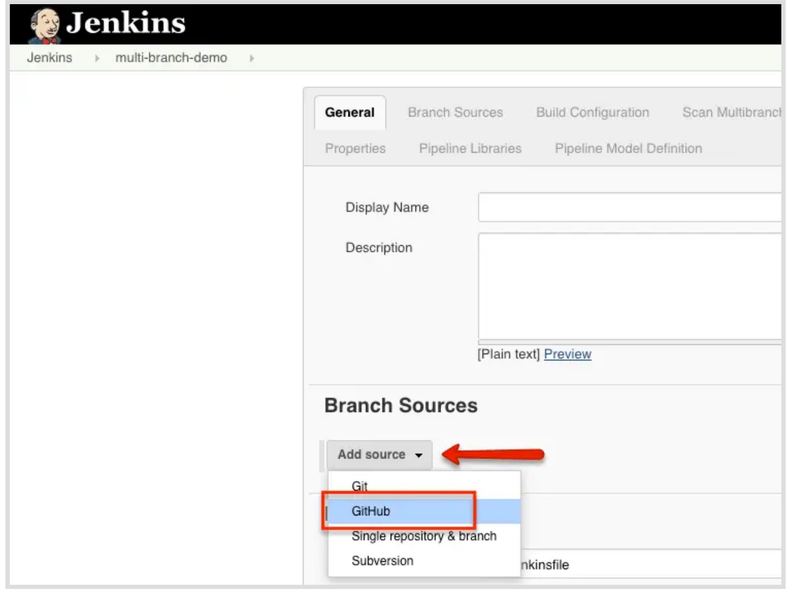
Step 1: From the Jenkins home page create a “new item”.

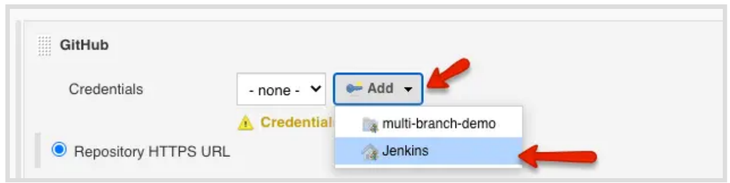


Step 2: Select the "Multibranch pipeline" from the option and click ok.



Step 3: Click "Add a Source" and select Github.





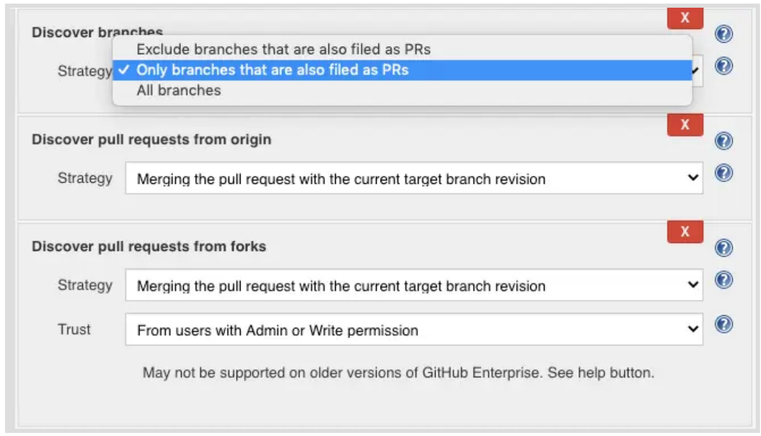
Step 5: Select the created credentials and provide your Github repo to validate the credentials as shown below.

If you are testing multi-branch pipeline you can clone the demo Github repo and use it. https://github.com/devopscube/multibranch-pipeline-demo.

Step 6: Under "Behaviours" select the required option matches your requirement. You can either choose to discover all the branches in the repo or only branches with a Pull Request.

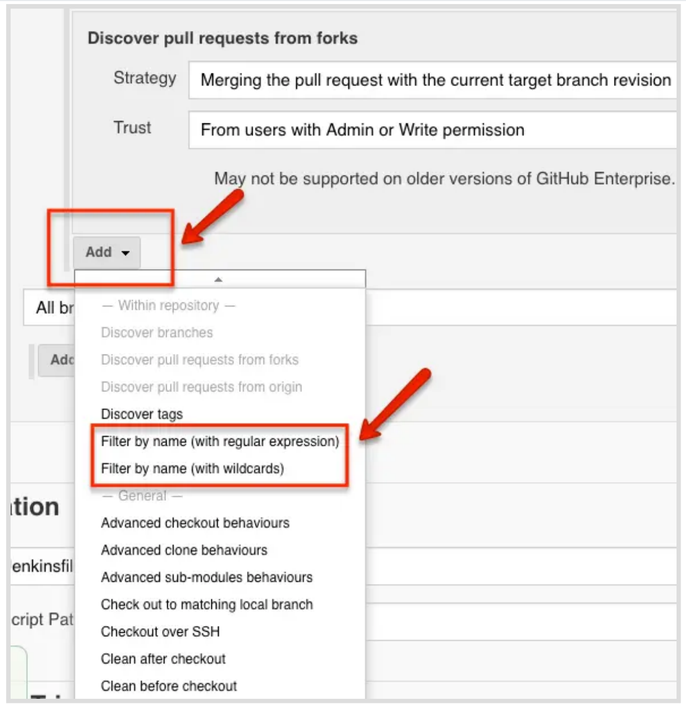
The pipeline can discover branches with a PR from a forked repo as well.

Choosing these options depends on your required workflow.

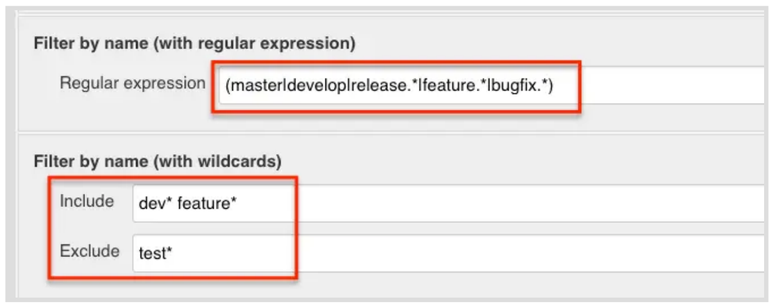


There are additional behavior you can choose from the "add" button.

For example, If you choose not to discover all the branches from the repo, you can opt for the regular expression or wildcard method to discover branches from the

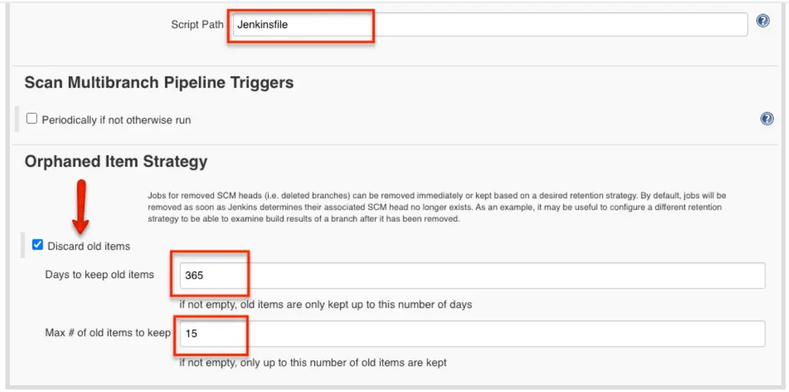


Here is a regex and wildcard example.



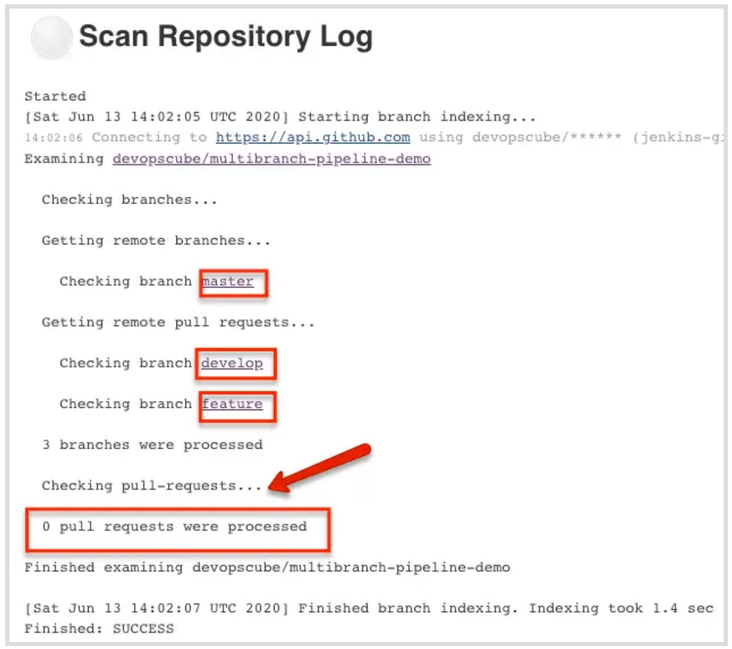
Step 7: If you choose to have a different name for Jenkinsfile, you can specify it in the build configuration. In the "Script Path" option, you can provide the required name. Ensure the Jenkinsfile is present in the repo with the same name you provide in the pipeline configuration.

Also, Enable "Discard old builds" to keep only required build logs as shown below.



Step 8: Save all the job configurations. Jenkins scans the configured Github repo for all the branches which has a PR raised.

The following image shows the job scanning the three branches, and since I haven't raised any pull request, Jenkins won't create any branch-based pipeline. I will show how to test the automatic pipeline creation after the webhook setup.



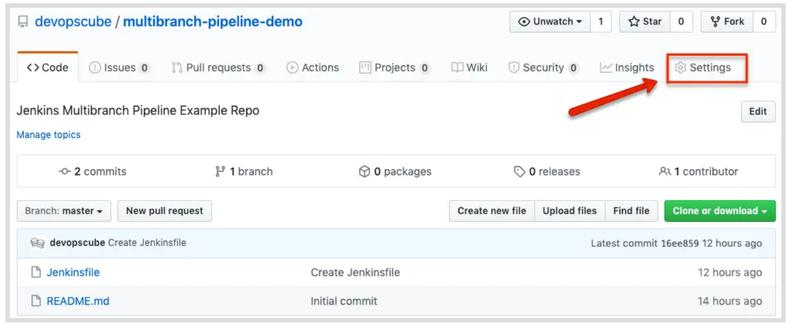
Till now, we have done configurations on the Jenkins side to scan branches based on the PR requests.

To have a complete workflow, we need to have a webhook configured in Github to send all the repo events (commits, PR etc) to Jenkins as the pipelines can be triggered automatically.

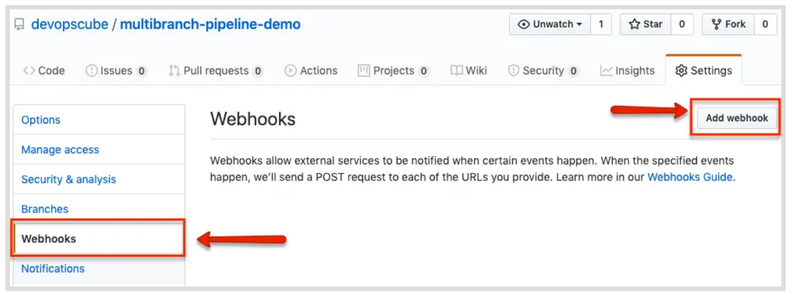
Pipeline

Follow the steps given below to setup the Jenkins webhook on the repo.

Step 1: Head over to the Github repo and click on the settings.

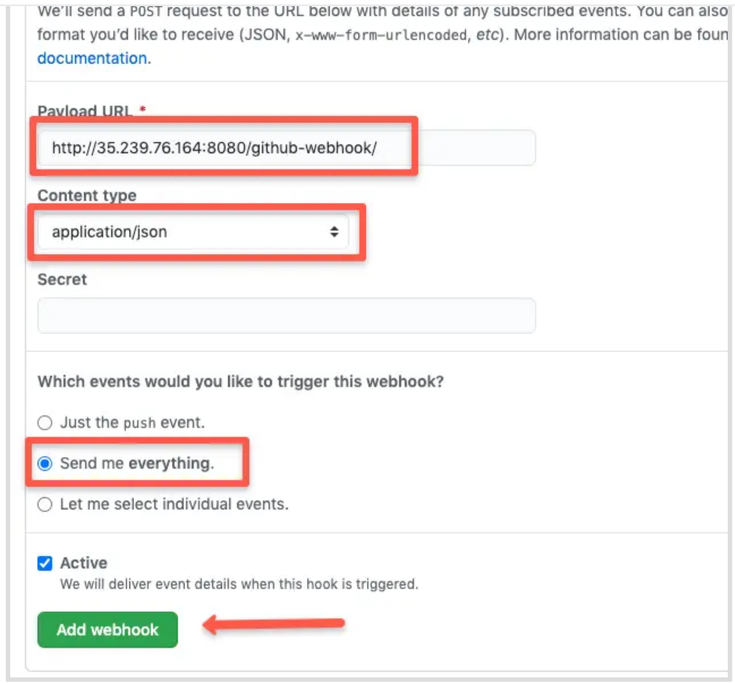


Step 2: Select the webhook option at the left and click “Add Webhook” button.

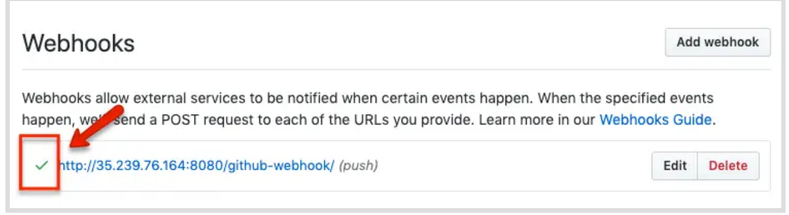


Step 3: Add your Jenkins URL followed by “/github-webhook/” under payload URL. Select the content type as “application/json” and click “Add Webhook”

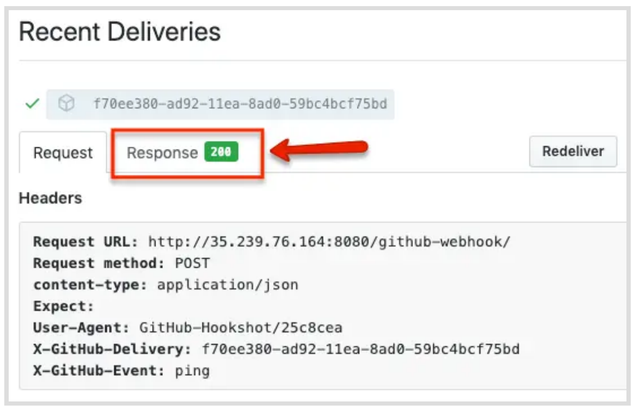
> **Note:** You can choose what type of webhook you want to receive in Jenkins. For example, you want to trigger the pipeline only during PR; then, you can select just the PR event from the “Let me select individual events” option.



You should see a green tick mark on a successful webhook configuration as shown below.



If you don't see a green tick or see a warning sign, click on the webhook link, scroll down to "Recent Deliveries," and click on the last webhook. You should be able to view why the webhook delivery failed with the status code.



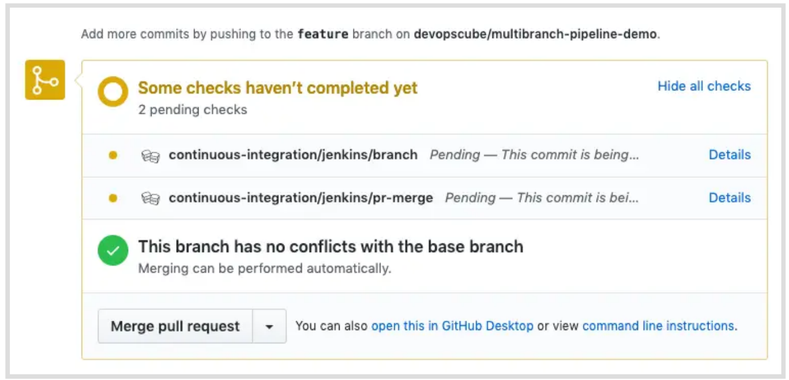
Test Multi-branch Pipeline

For demo purpose, I have chosen the option “Only Branches that are file as PR”. With this option, only the branches with a PR request gets discovered.

To play around with a multi-branch pipeline, you can use this repo with a sample Jenkinsfile –> Multibranch Pipeline Demo Repo

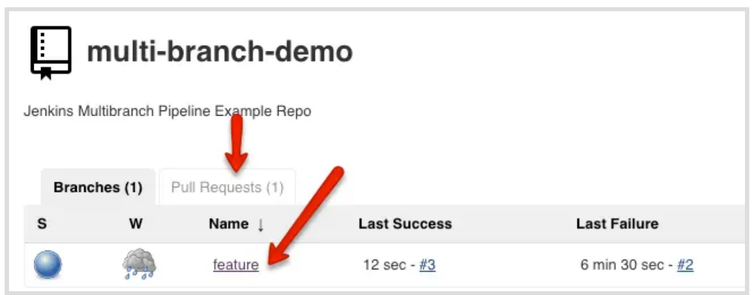
This repo has three branches. master, develop and feature.

Update some content in the readme file in the feature branch and raise a PR to develop. It will send a webhook to Jenkins and Jenkins will send back the Jenkins job details and the PR will go to check state as shown below.



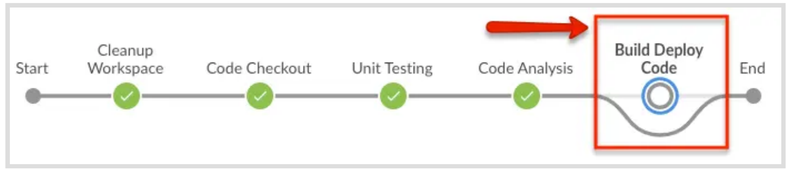
If you click the “Details” it will take you to the Jenkins build log. You can write custom check in your Jenkinsfile that can be used for the build reviews.

Now, if you check Jenkins you will find a pipeline for feature branch in Jenkins as shown below.



If the build fails, you can commit the changes to the feature branch and as long as the PR is open, it will trigger the feature pipeline.

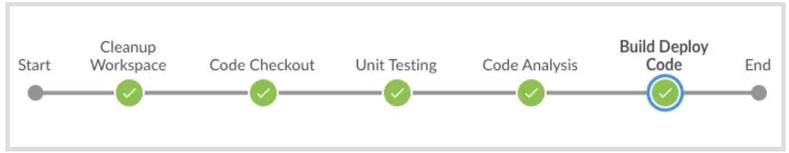
in the blue ocean dashboard you can clearly see the skipped deployment stage as shown below.



Now merge the feature branch PR and raise a new PR from develop to the master branch.

Jenkins will receive the webhook from Github for the new PR and the develop pipeline gets created as shown below.

For develop branch, the deploy stage is enabled and if you check the Blue Ocean build flow you can see all the stages successfully triggered.



Troubleshooting Multibranch Pipelines

I will talk about a few possible errors in a multibranch pipeline that you might encounter and how to troubleshoot them.

Branch Discovery Issue

Sometimes even after creating new branches in the SCM, it might not reflect in the Jenkins pipeline. You can try running the "Scan Repository Now" option to scan the repo again. Also, check the repository scan configurations in the pipeline.

PR Webhooks Not Triggering the Pipelines

logs from `Manage Jenkins --> System Logs --> All Jenkins logs` . If Jenkins is able to receive the webhook, the log should show the reason why the jobs are not getting triggered.

Commits Not Triggering Pipeline

If you want each commit to rigger the branch pipeline, then you should select the "**Discover All Branches**" option in the branch discovery configuration. So whenever you commit a change to the discoverable branches or raise a PR, the pipeline will automatically get triggered.

Multibranch Pipeline Best Practices

Let's have a look at some of the best practices for a multibranch pipeline.

Repo Branching – Have a Standard Structure

It is essential to have the standard branching structure for your repositories. Whether it is your application code or infra code, having a standard branching will reduce the inconsistent configurations across different pipelines.

Shared Libraries – Reusable Pipeline Code

Make use of shared libraries for all your multi-branch pipelines. Reusable libraries make it easy to manage all the pipeline stages in a single place.

Pull Request Vs Commit Triggers

Try to use a PR based pipeline rather than commit based. If a code repo gets continuous commits it might overwhelm Jenkins with many builds.

Commit based triggers are supported in PR based discovery as well. Here the commit trigger happens only when the PR is still open.

Jenkins Pipeline Vs. Multibranch Pipeline

A normal pipeline job is meant for building a single branch from the SCM and deploy to a single environment. However, you can

A multibranch pipeline is meant for building multiple branches from a repository and deploy to multiple environments if required.

Use pipeline job for adhoc jobs, parameterised job executions and to debug pipeline as code.

Do not use multibranch pipeline if you do not have a standard branching and CI/CD strategy.

Multibranch Pipeline Vs. Github Organization Job

Like a multi-branch pipeline, the Github organization folder is one of the Jenkins project types.

> **Note:** The organization folder is not just limited to Github. It can be use used for Gitlab, Bitbucket teams, or Gitea organization.

Mullitbranch pipleine can only configure pipelines for a single Git repository. Whereas a Jenins Github organization project can automatically configure multi-branch pipelines for all the repos in a Github organization.

It can discover all the repositories in the configured Github organization, with a Jenkinsfile.

Also, you can configure a generic webhook in the organizational level to avoid having webhooks in each repo.

The only difference between multi-branch and organization project is that organizations can configure multi-branch pipelines for multiple repos.

So which one should I use?

This totally depends on the workflow you need. If you have a standard pipeline and process of deploying applications or infra code, Github organization is great. Or else, configuring multibranch pipeline separately will be a good option.

I'd like to hear from you

That's all for my guide to multi-branch pipelines.

Do you think multi-branch pipeline will add value to your workflows?

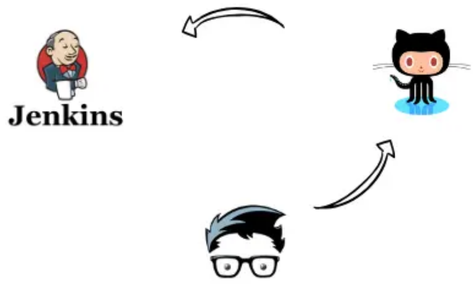
Or maybe you want stick to regular Jenkinsfile pipelines

Either way, let me know by leaving a comment.

Bibin Wilson

An author, blogger and DevOps practitioner. In spare time, he loves to try out the latest open source technologies. He works as an Associate Technical Architect

YOU MAY ALSO LIKE

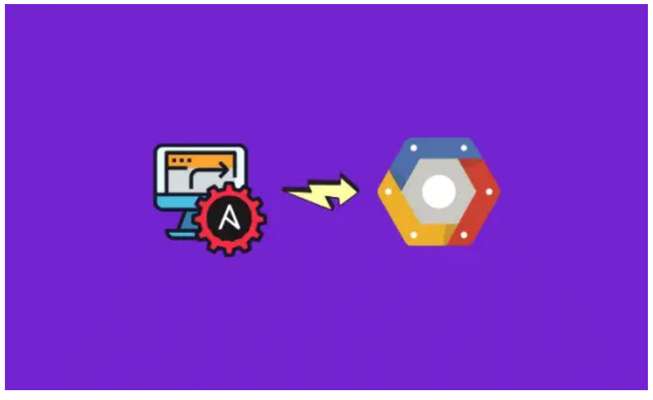


D — DEVOPS

Jenkins Automated Build Trigger On Github Pull Request

by **devopscube** · June 16, 2020

Building projects based on pull request is something you cannot avoid in CI/CD pipelines. Nowadays every team does...

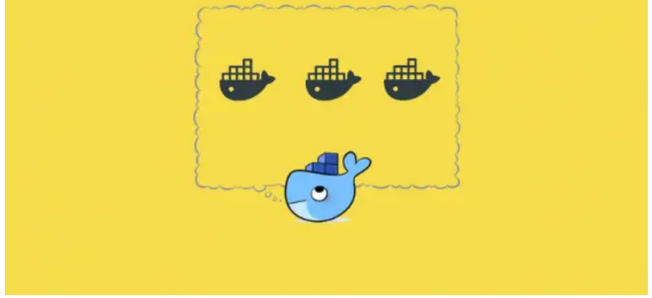


D — DEVOPS

How To Setup Ansible Dynamic Inventory For Google Cloud

by **Bibin Wilson** · June 20, 2020

The best way to manage and orchestrate VM instances in Google cloud using Ansible is through Dynamic inventory...



How To Run Docker in Docker Container [3 Easy Methods]

by **Bibin Wilson** · June 25, 2021

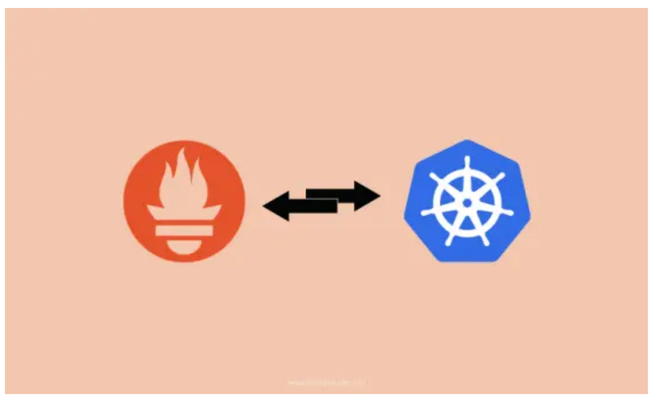
In this blog, I will walk you through the steps required to run docker in docker using three...

D — DEVOPS

How To Setup and Configure a Proxy Server – Squid Proxy

by **devopscube** · August 11, 2018

A proxy server has many use cases. it could range from personal internet access to restrict organization systems/servers...

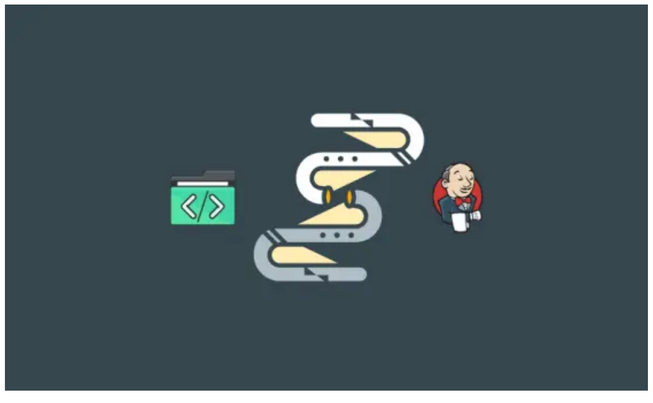


D — DEVOPS

How to Setup Prometheus Monitoring On Kubernetes Cluster

by **Bibin Wilson** · April 7, 2021

This article will guide you through setting up Prometheus on a Kubernetes cluster for monitoring the Kubernetes cluster....



D — DEVOPS

Jenkins Pipeline Tutorial For Beginners

by **Bibin Wilson** · April 11, 2020

Jenkins pipeline as code is a concept of defining Jenkins build pipeline in Jenkins DSL/Groovy format. This article...

DevopsCube