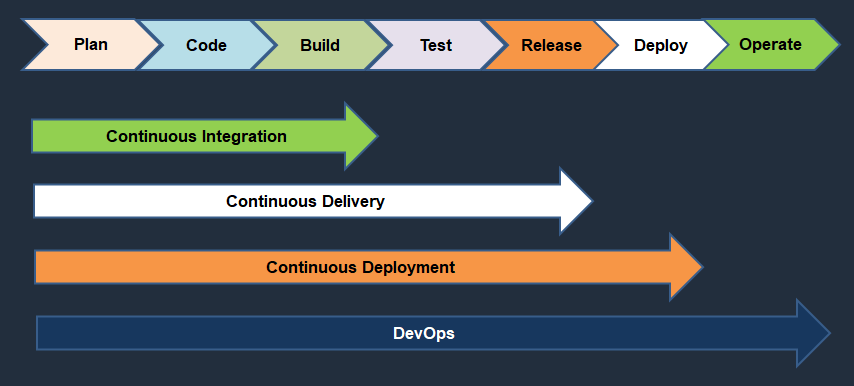
1. **What is Continuous Integration with reference to Jenkins?**

**Continuous Integration:**

* All Developers commit their source code changes to the shared Git repository
* Jenkins server checks the shared Git repository at specified intervals and detected changes are the taken into the build.
* The build results and test results are shared to the developers
* The build application is displayed on a test servers like Junit/Selenium and automated tests are ran.
* The clean and tested build is deployed to the production server.
* The commit, build,test, and deploy is a continuous process.

1. **What are the difference between Continuous Integration, continuous Delivery and Continuous Deployment.**



**Continuous Integration:**

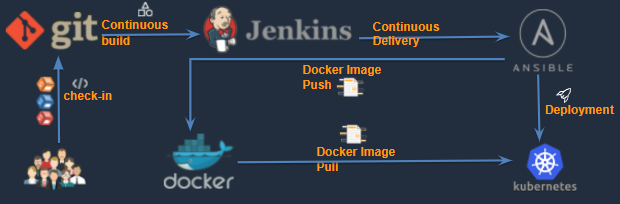
* Developers will push their code several times in a day to a central repository, every time there is code change it should be pulled, built, tested and notified.

**Continuous Delivery**

* The process of CI, the artifact has been generated.
* The artifact should be delivering to all servers in different environments like Dev, QA, Staging.
* It should be automatically delivered to QA servers where testers will do functional tests, load tests etc.
* After it passes the QA tests it should automatically deliver the code to staging area where customer or client or users check the changes and approval to deploy it to production.

**Continuous Deployment**

* If the approval is manual then code delivery is continuous Delivery but if the approval process becomes automated then after staging, the code change is done directly to Production systems.
* This is called as Continuous Deployment.



1. **What is workspace location in Jenkins**

* The workspace directory is where Jenkins builds your project.
* It contains the source code Jenkins checks out, plus any files generated by the build itself.
* This workspace is reused for each successive build.
* Default Workspace is JENKINS\_HOME =/var/lib/jenkins

1. **What are the few major plugins using in your project and explain each every plug-in use case**

Some of plugins are used are here

* + Green Balls
  + Git
  + Maven
  + pipeline
  + publish over ssh
  + ssh plugin
  + copy artifacts
  + Blue Ocean
  + Sonarqube scanner
  + artifactory
  + Monitoring
  + Docker
  + Kubernetes
  + ThinBackup

1. **Name the two components that Jenkins is mostly integrated with**

* Version Control System
* Build Tool (Maven)

1. **What is a multi branch pipeline?**

* The **Multibranch Pipeline** project type enables you to implement different Jenkinsfiles for different branches of the same project.
* In a Multibranch Pipeline project, Jenkins automatically discovers, manages and executes Pipelines for branches which contain a Jenkinsfile in source control.

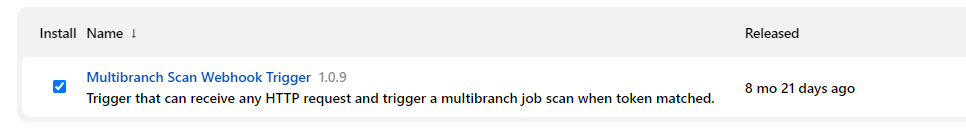
**Jenkinsfile:** <https://gitlab.com/arsravis/taxigrabber/-/blob/main/Jenkinsfile>

Refer link: https://www.youtube.com/watch?v=fo36b23cpIU

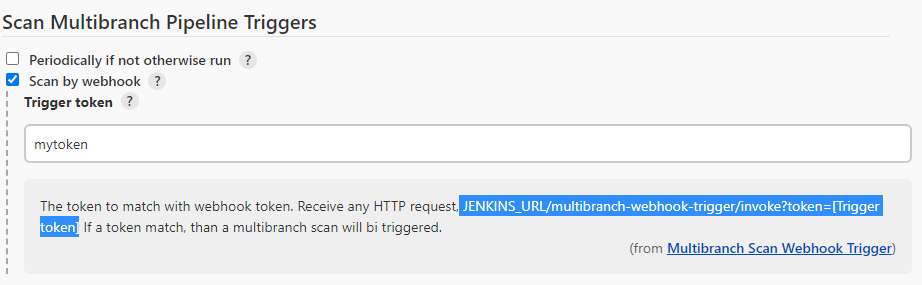
1. **How to enable webhook for Jenkins in Git.**

**Configure Web hook:**

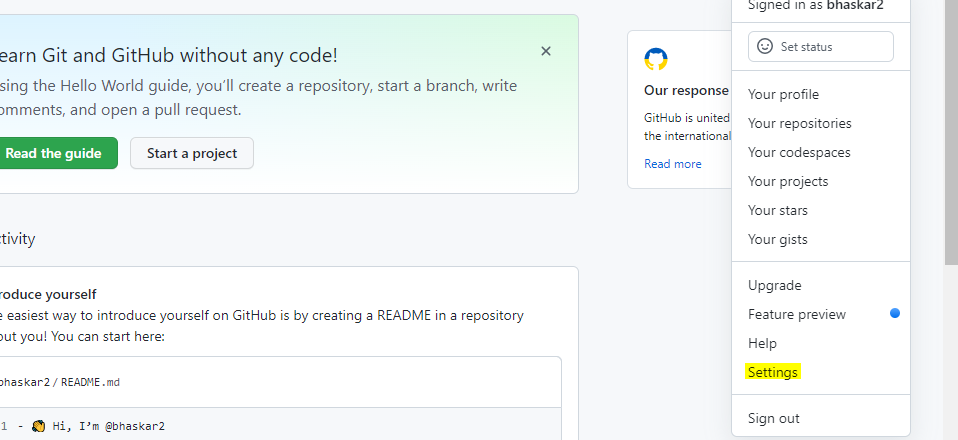
* We have to configure our Jenkins machine to communicate with our GitHub repository for that we need Hook URL of the Machine.
* Go to Mange Jenkins and Install Multibranch Scan Webhook Trigger plugin



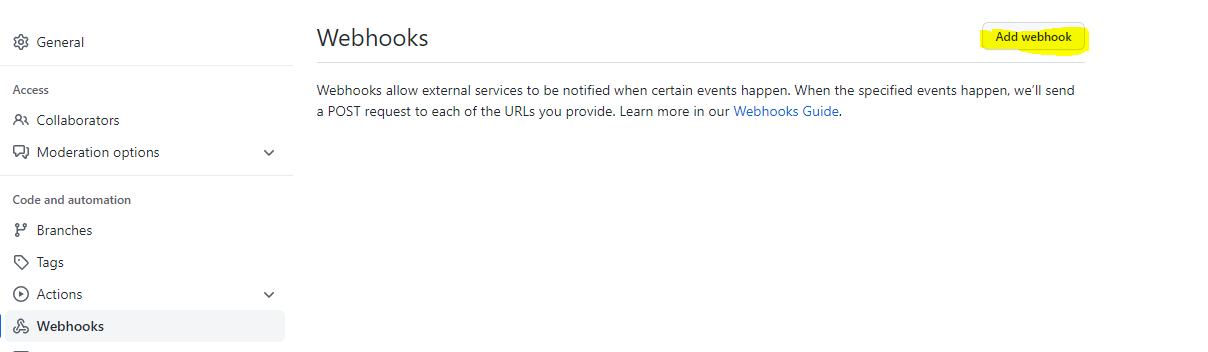
* Take the token from scan Multibranch Pipeline Triggers



* Go to GitHub and select setting



* Select the projects 🡪 settings-- > then Webhooks –Add Webhooks



1. **What is Jenkins shared library ?**

* As an organization starts using more and more pipeline jobs, there is a chance for more and more code being duplicated in every pipeline job, since a part of the build/automation processes will be the same for most of the jobs. In such a situation, every other new upcoming job should also duplicate the same piece of code.
* To avoid duplications, the Jenkins project brought in the concept of Shared Libraries, to code - DRY - Don't Repeat Yourself.
* Shared libraries are a set of code that can be common for more than one pipeline job and can be maintained separately. Such libraries improve the maintenance, modularity & readability of the pipeline code. And it also speeds up the automation for new jobs.

1. **How to create & use a Shared Library in Jenkins?**

Basic requirements for a Jenkins shared library to be used in a Pipeline Code are -

A Repository with pipeline shared library code in SCM.

An appropriate SCM Plugin configuration for the Jenkins instance.

Global Shared Library should be configured in Jenkins Global configuration.

Include the Shared Library in the Pipeline Code and use the methods defined in the Jenkins Shared Library.

E.g.

#!/urs/bin/env groovy

@Library('fs\_jenkins\_shared\_library@v2.0.7')\_

1. **How to setup Jenkins slave system.**

**What this in YouTube: https://www.youtube.com/watch?v=9RsmPNs7gT0**

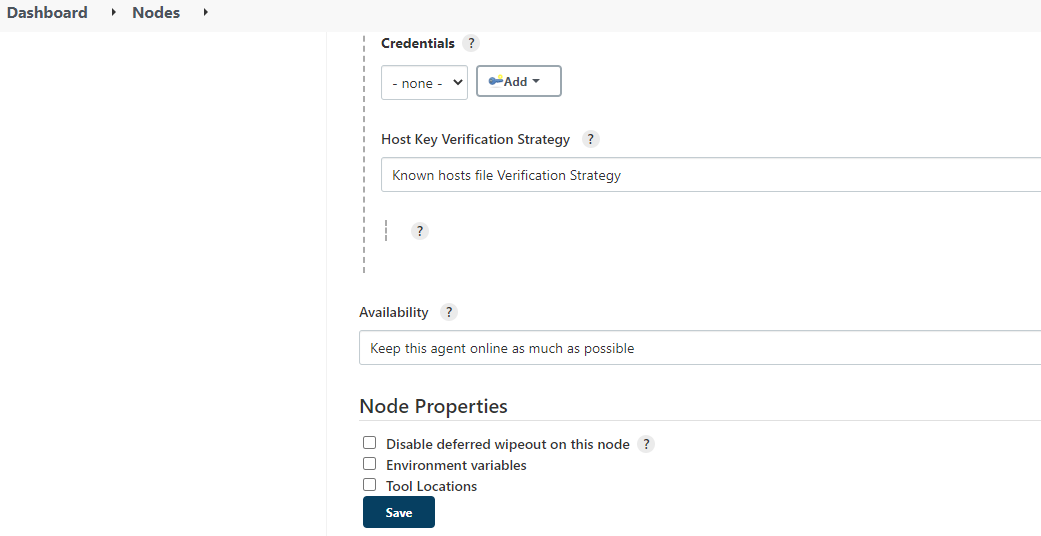
The Jenkins master acts to schedule the jobs and assign slaves and send builds to slaves to execute the jobs**.**

**Configure Jenkins master and salve nodes**

1. Click on **Manage Jenkins** in the left corner on the Jenkins dashboard
2. Click on Manage Nodes
3. Select New Node and enter the name of the node in the Node Name field
4. Select Permanent Agent and click on OK button.

Initially you will get only one option, “Permanent Agent”. Once you have one or more slaves you will get the copy existing node Option.

1. Enter required information
2. Enter the host name in the Host field.
3. Select the Add button to add credentials. And click Jenkins
4. Enter username, password, ID, and Description
5. Select the dropdown menu to add credentials in the Credential field.
6. Select the next dropdown to add the Host key verification Strategy under None verifying verification Strategy.
7. Keep this agent online as much as possible
8. Click the save button



1. **What is Jenkins Pipeline?**

* Instead of creating Jenkins jobs from user interface you would write a file or script to create a job.
* So it’s a “scripted file”
* Jenkins Pipeline is a suite of plugins that supports implementing and integrating continuous delivery pipelines into Jenkins.
* Document : https://www.jenkins.io/doc/book/pipeline/
* Creating a Jenkinsfile and committing it to source control provides a number of immediate benefits:
* Automatically creates a Pipeline build process for all branches and pull requests.
* Code review/iteration on the Pipeline (along with the remaining source code).
* Audit trail for the Pipeline.

**Pipeline Types:**

* Scripted Pipeline
* Declarative Pipeline (new feature)
* **Declarative Pipeline** is a more recent feature of Jenkins Pipeline which: provides richer syntactical features over Scripted Pipeline syntax, and is designed to make writing and reading Pipeline code easier.

1. **How to identify a pipeline is declarative or scripted?**
2. **git Pull vs Git fetch**

| **Git pull** | **Git Fetch** |
| --- | --- |
| This command pulls new changes from the currently working branch located in the remote central repository. | This command is also used for a similar purpose but it follows a two step process:  1. Pulls all commits and changes from desired branch and stores them in a new branch of the local repository.  current  2. For changes to be reflected in the current / target branch, git fetch should be followed by git merge command. |

1. **How to enable versioning for the artifacts in Jenkins. (BUILD\_ID)**
2. **How can you updated artifacts on to Artifactory?**
3. **What is your approach when Jenkins job failed?**
4. **How do you store credentials in Jenkins securely?**
5. **What is protected branch**
6. **What is pull request? How the approval process does go.**
7. **Explain types of testing done in your pipeline**
8. **What is local, virtual and remote artifact repositories ?**
9. **Can you walk through the Jenkins pipeline which you have written in your environment.**
10. **How frequently do you run your Jenkins jobs**
11. **What are maven goals are you using in your project**
12. **What is the source code you are using in your project**
13. **What is the code coverage what is the code coverage percentage do you use**
14. **What are the quality gateways do you use.**
15. **I want to create a dependent jenkins job. It should run when the prvious job is successful**
16. **Cron jobs importance?**