1) What is Jenkins pipeline?

Jenkins pipeline is a way to define and automate continuous delivery pipelines as code, enabling the creation of workflows that encompass the entire software delivery process.

2) What scripting language is Jenkins pipeline syntax based on?

   Jenkins pipeline syntax is based on the Groovy scripting language.

3) What are the different ways to trigger pipeline?

   A) SCM Trigger: The pipeline can be automatically triggered whenever changes are detected in the source code repository. This is typically done using SCM tools like Git.
   B) Timer Trigger: The pipeline can be scheduled to run at specific intervals using a cron-like syntax. This allows you to automate regular builds or deployments.
   C) Manual Trigger: A pipeline can be triggered manually by a user through the Jenkins web interface or by using Jenkins CLI. This gives users control over when to initiate the pipeline execution.
   D) Webhooks: Jenkins can be configured to listen for incoming webhooks from external services or systems. When a webhook is received, it triggers the associated pipeline, allowing for integration with external event-driven systems.
   E) Pipeline DSL: Jenkins Pipeline DSL allows you to define custom triggers within your pipeline script. You can include logic and conditions that determine when the pipeline should be triggered based on specific criteria or events.

4) what is different between parameter and jenkins env variable?
   A) Parameters:
      - User-defined inputs provided during job execution.
      - Customizable values that affect job behavior.
      - Can be specified during job configuration and accessed within the pipeline script.

   B) Environment Variables:
   - Predefined variables that store information about the Jenkins environment and build context.
   - Provide details such as current build number, workspace directory, and Git commit details.
   - Accessible throughout the pipeline script and used for conditional logic and integration with external systems.

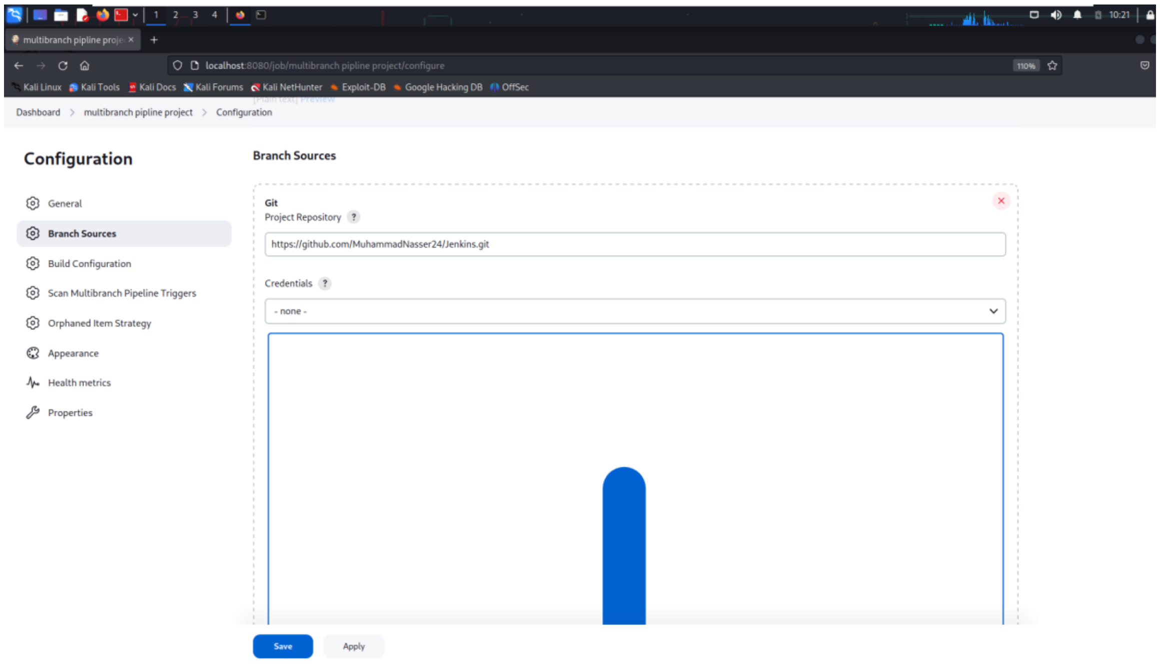
5) what is organization folder job and what is used for ?

Organization Folder job in Jenkins is designed to manage multiple related projects or repositories within a version control system, such as Git, in a hierarchical structure. It acts as a container or folder that holds multiple pipelines or projects, allowing you to define and manage a group of related jobs as a single entity. The Organization Folder job provides a centralized view and management interface for your projects, making it easier to organize and maintain pipelines and projects at scale.

6) Create jenkins pipeline for your repo and use script file (jenkinsfile) to write pipeline syntax, include parameter functions and env variable in it ?

https://github.com/MuhammadNasser24/Jenkins.git

7) Create another multibranch pipeline and filter branches to contain only (master , dev , test) ?

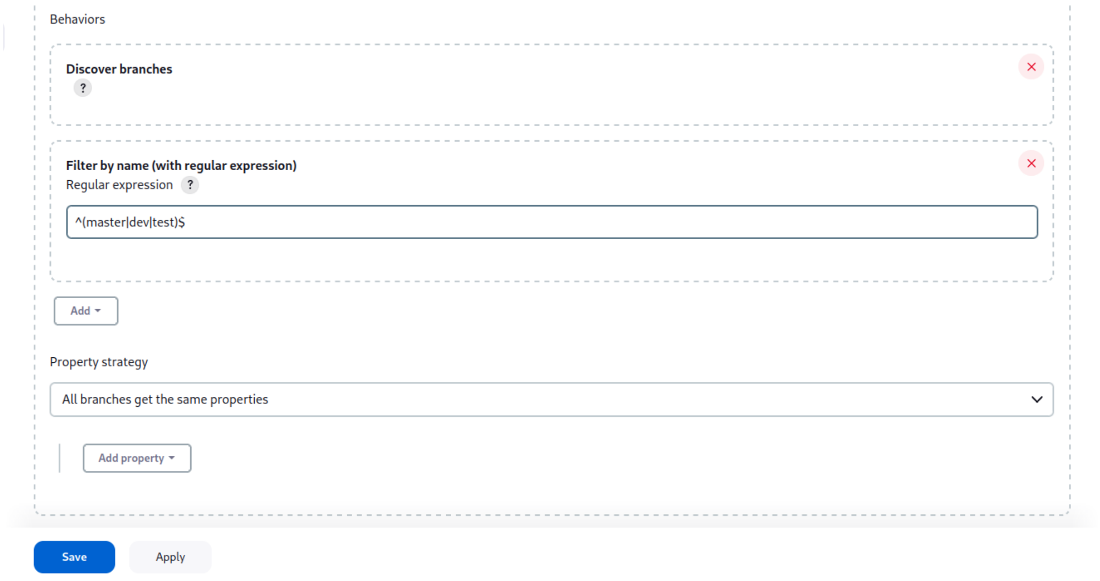


localhost:8080/job/multibranch pipline project/configure

Configuration

Branch Sources

Git
Project Repository: https://github.com/MuhammadNasser24/Jenkins.git
Credentials: - none -



Behaviors

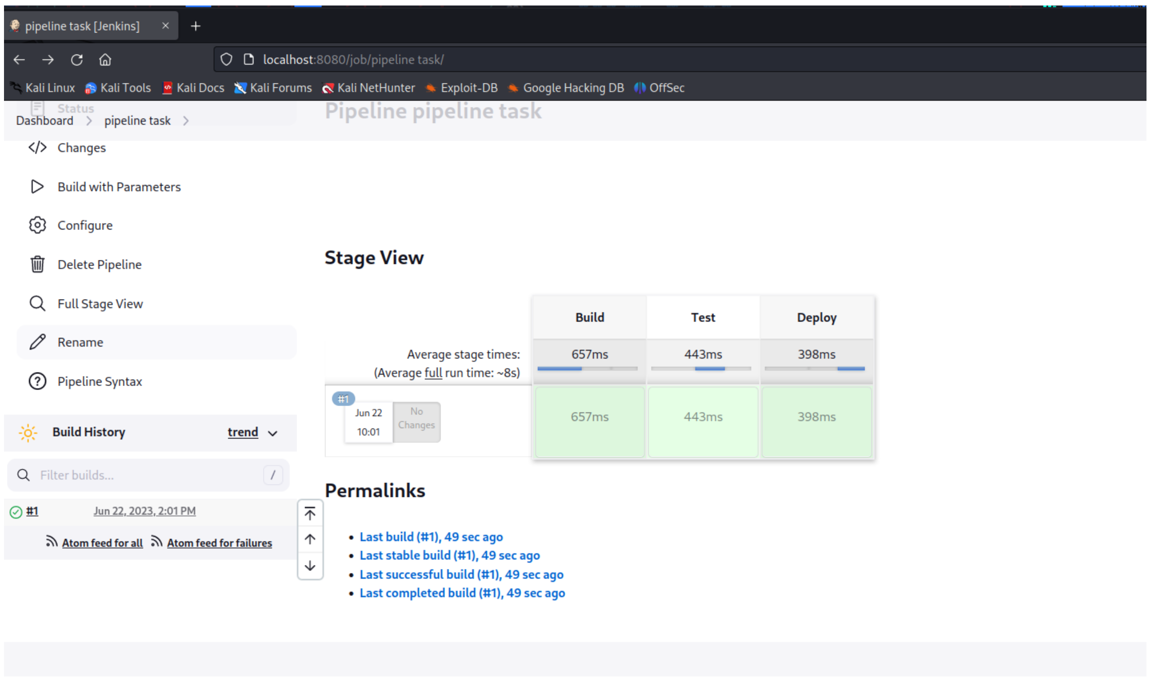
Discover branches

Filter by name (with regular expression)
Regular expression

```
^(master|dev|test)$
```

Property strategy

All branches get the same properties



pipeline task [Jenkins]

localhost:8080/job/pipeline task/

Pipeline pipeline task

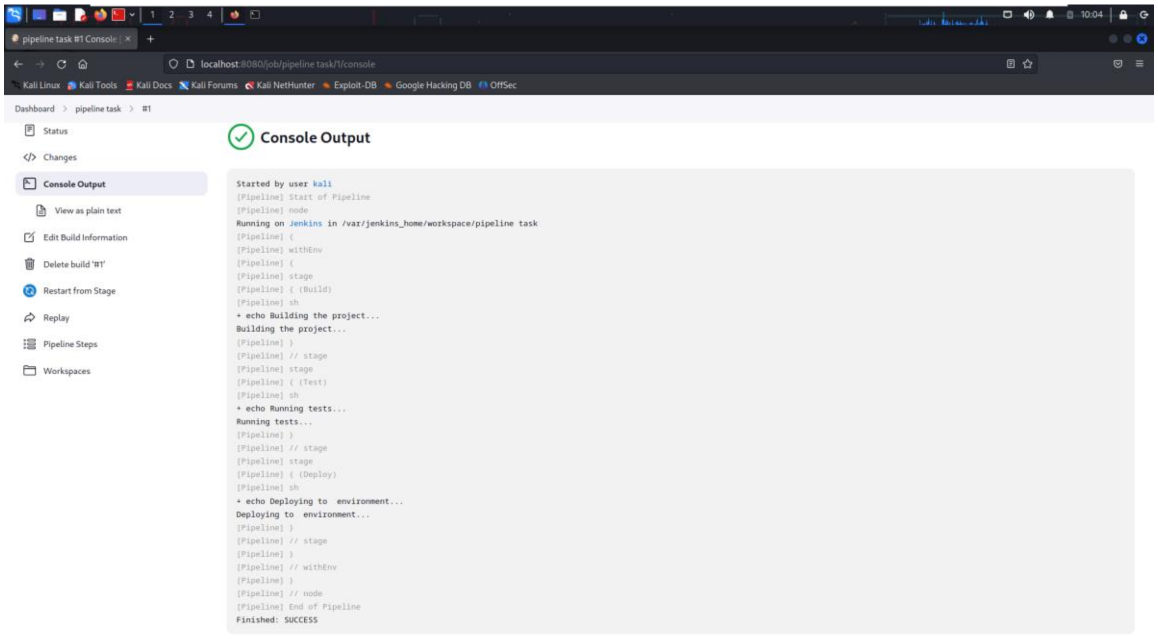
Stage View

	Build	Test	Deploy
Average stage times: (Average full run time: ~8s)	657ms	443ms	398ms
#1 Jun 22 10:01 No Changes	657ms	443ms	398ms

#1 Jun 22, 2023, 2:01 PM

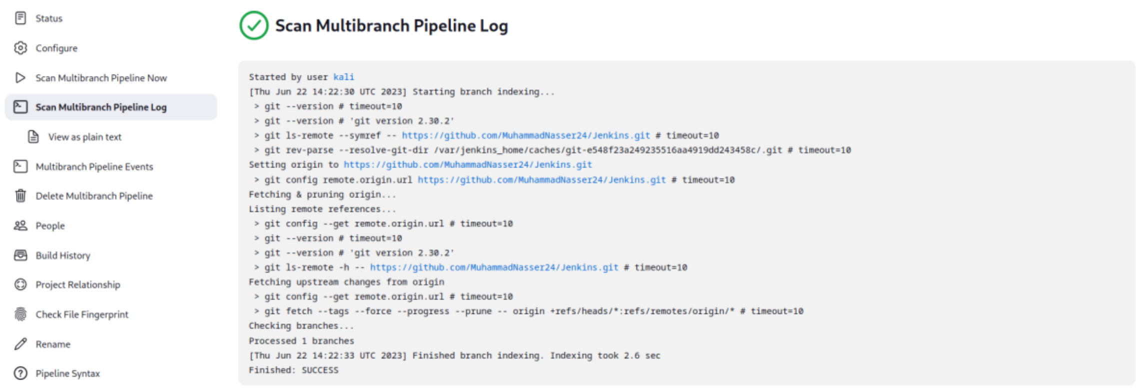
Permalinks

- Last build (#1), 49 sec ago
- Last stable build (#1), 49 sec ago
- Last successful build (#1), 49 sec ago
- Last completed build (#1), 49 sec ago



Console Output

```
Started by user kali
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/pipeline task
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
+ echo Building the project...
Building the project...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh
+ echo Running tests...
Running tests...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] sh
+ echo Deploying to  environment...
Deploying to  environment...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Scan Multibranch Pipeline Log

```
Started by user kali
[Thu Jun 22 14:22:30 UTC 2023] Starting branch indexing...
 > git --version # timeout=10
 > git --version # 'git version 2.30.2'
 > git ls-remote --symref -- https://github.com/MuhammadNasser24/Jenkins.git # timeout=10
 > git rev-parse --resolve-git-dir /var/jenkins_home/caches/git-e548f23a249235516aa4919dd243458c/.git # timeout=10
Setting origin to https://github.com/MuhammadNasser24/Jenkins.git
 > git config remote.origin.url https://github.com/MuhammadNasser24/Jenkins.git # timeout=10
Fetching & pruning origin...
Listing remote references...
 > git config --get remote.origin.url # timeout=10
 > git --version # timeout=10
 > git --version # 'git version 2.30.2'
 > git ls-remote -h -- https://github.com/MuhammadNasser24/Jenkins.git # timeout=10
Fetching upstream changes from origin
 > git config --get remote.origin.url # timeout=10
 > git fetch --tags --force --progress --prune -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking branches...
Processed 1 branches
[Thu Jun 22 14:22:33 UTC 2023] Finished branch indexing. Indexing took 2.6 sec
Finished: SUCCESS
```