

## **1) What is the Jenkins pipeline?**

Jenkins Pipeline is a suite of plugins that provides an extensible, script-based automation framework for building continuous delivery pipelines. It allows developers to define a set of stages, each containing a series of steps, that can be used to build, test, and deploy software applications.

With Jenkins Pipeline, developers can use a domain-specific language (DSL) called "Jenkinsfile" to define their pipelines as code. This allows them to version control their pipeline definitions and to manage their pipelines in the same way they manage their application code.

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

Jenkins Pipeline syntax is based on Groovy, a powerful and flexible programming language that runs on the Java Virtual Machine (JVM). Groovy is a dynamic language that supports scripting, functional programming, and object-oriented programming paradigms.

Jenkins Pipeline uses a Groovy-based domain-specific language (DSL) to define pipelines as code. This DSL provides a set of functions and constructs that are specific to Jenkins Pipeline, such as the "node" function for defining an agent to run a pipeline stage on, or the "sh" step for executing shell commands.

## **3) What are the different ways to trigger the pipeline?**

1. SCM polling: Jenkins can be configured to poll a source code management (SCM) system, such as Git or Subversion, at regular intervals to check for changes. If changes are detected, Jenkins can automatically trigger a pipeline build.
2. Webhooks: Many SCM systems support webhooks, which are HTTP callbacks that can be triggered when certain events occur, such as a code push. Jenkins can be configured to listen for these webhooks and

automatically trigger a pipeline build when they are received.

3. Manual triggering: Developers can manually trigger a pipeline build from the Jenkins UI or using the Jenkins API. This can be useful for testing or debugging purposes, or for manually initiating a deployment.
4. Timer triggers: Jenkins pipelines can be scheduled to run at specific times using cron-like syntax. This can be useful for running nightly builds or other periodic tasks.
5. Upstream triggers: Jenkins pipelines can be configured to trigger downstream pipelines when they complete. This can be useful for creating complex multi-stage pipelines or for triggering deployments after a successful build.

#### **4) What is the difference between parameter and jenkins env variable?**

The main difference between Jenkins parameters and environment variables is that parameters are values that are passed into the pipeline at runtime and can be different for each pipeline run, while environment variables are predefined values that provide information about the pipeline execution environment and are constant throughout the pipeline run.

#### **5) What is the organization folder job and what is used for?**

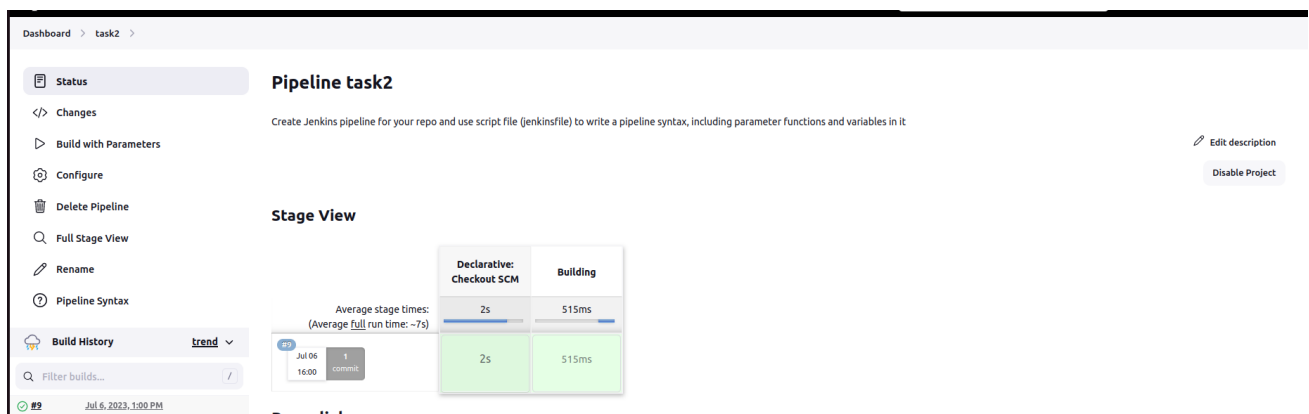
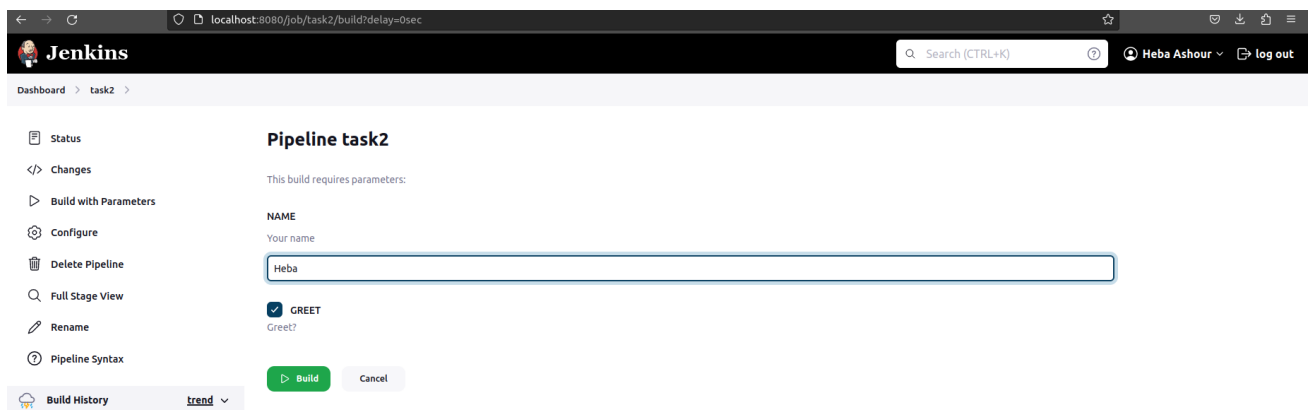
Organization folder jobs are typically used in large-scale software development environments where there are many different teams, projects, and repositories. They provide a centralized location to manage and configure pipelines or jobs across the organization, making it easier to maintain consistency and scalability.

An organization folder job is created by defining a set of rules or patterns that specify how to discover and configure pipelines or jobs across different repositories or projects. For example, you might define a rule that all repositories with a certain naming convention should have a pipeline called

"build" or "deploy", and that these pipelines should use a specific set of parameters or environment variables.

Once the organization folder job is set up, it will automatically discover and create pipelines or jobs based on the defined rules. It will also handle updates and changes to the pipelines or jobs as they are made in the repositories or projects.

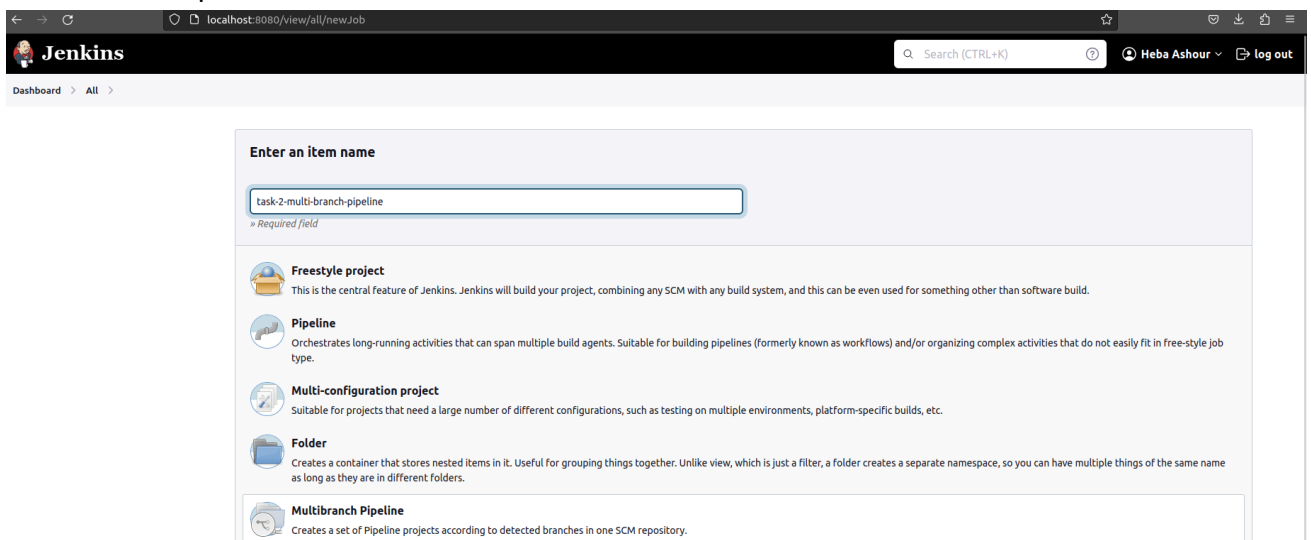
## 6) Create Jenkins pipeline for your repo and use script file (jenkinsfile) to write a pipeline syntax, including parameter functions and variables in it ?



```
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Building)
[Pipeline] echo
Hello, Heba!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

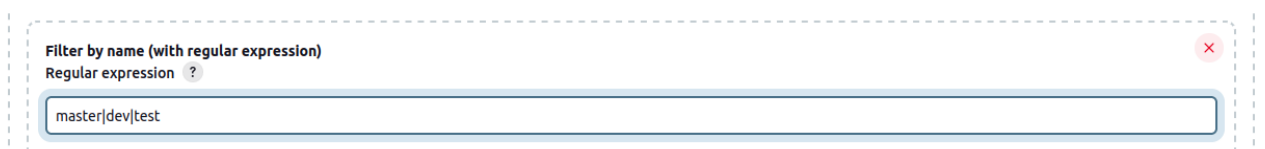
## 7- Create another multibranch pipeline and filter branches to contain only (master , dev , test )

- Create a new multibranch pipeline in Jenkins by clicking on "New Item" and selecting "Multibranch Pipeline".



Under "Behaviors", add the "Filter by Regular expression" behavior.

In the "Regular expression" field, enter a regular expression that matches the branches you want to build. For example, to only build the **master**, **dev**, and **test** branches



## multibranch pipeline




Folder name: task-2-multi-branch-pipeline

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

Disable Multibranch Pipeline

Branches (1)

Pull Requests (0)


S	W	Name ↓	Last Success	Last Failure	Last Duration	
		dev	1 min 53 sec <a href="#">#1</a>	N/A	1 min 8 sec	

Icon: S M L

Icon legend

 Atom feed for all

 Atom feed for failures

 Atom feed for just latest builds