Jenkins Job-DSL

Groovy file is a scripting language (is a programming language, that's used to automate any process in the software) which runs on Java Virtual Machine (JVM). Groovy file is built into Jenkins to run pipeline and used by Job-DSL (Domain Specific Language) plugin. It can combine different languages together meaning teams in your project can be contributing in different languages. Groovy source code is converted to JVM bytecode so it can run on any platform. Groovy takes the data type based on the value assigned to the object rather than explicitly mentioning it.

Jenkinsfile can be created in the following way:-

1. You can directly enter the code in the configuration of the pipeline.
2. In SCM, Jenkinsfile can be written and pushed to the control repository. The file's path can be used to find the Jenkinsfile.

Jenkins have 2 pipelines:-

1. Scripted pipeline
2. Declarative pipeline

Scripted pipeline is groovy based DSL. It provides greater flexibility and scalability for Jenkins users than the declarative pipeline. Groovy scripts are not suitable for all users, so Jenkins created declarative pipeline.

```
//Jenkinsfile for Scripted pipeline

pipeline{

        node any //node { <docker image / file>}

                stage(stage-1){

                        echo 'Jenkins Job-DSL'

                }

}
```

```
//Jenkinsfile for Declarative pipeline

pipeline{

        agent any //agent { <docker image / file>}

                stage(stage-1){

                        echo 'Jenkins Job-DSL'

                }

}
```

node/agent:- It defines where to execute the code on which machine or platform. Sometimes there can multiple node/agent, to distribute the job on Jenkins, for different kinds of projects.

Stage:- There can be one or multiple stages in Jenkinsfile, where each stage section has different steps and commands to follow. It's recommended that stages contain atleast one stage directive, like, Cloning stage, jobdsl, job_dsl_multibranch, Building & Running Image.

```
pipeline {
  agent {docker{image 'jfrog.asux.aptiv.com/corecomp_devops-
aptiv00000000-docker-local/studenttest:1.0'}}
  stages {
      stage('Cloning stage') {
              steps {
                   sh 'git clone http://github.com/madler/zlib.git'
                   sh "pwd"
                   dir("zlib"){
                           sh "pwd"
                   }
              }
       }

       stage('jobdsl'){
              steps{
                  jobDsl(targets:'jobdsl.groovy')
              }
      }

      stage('job_dsl_multibranch'){
             steps{
                  jobDsl(targets:'job_dsl_multibranch.groovy')
             }
      }

      stage('Building & Running Image') {
             steps {
                  dir("zlib"){
                        sh './configure'
                        sh 'make test'
                  }
            }
      }
  }
}
```

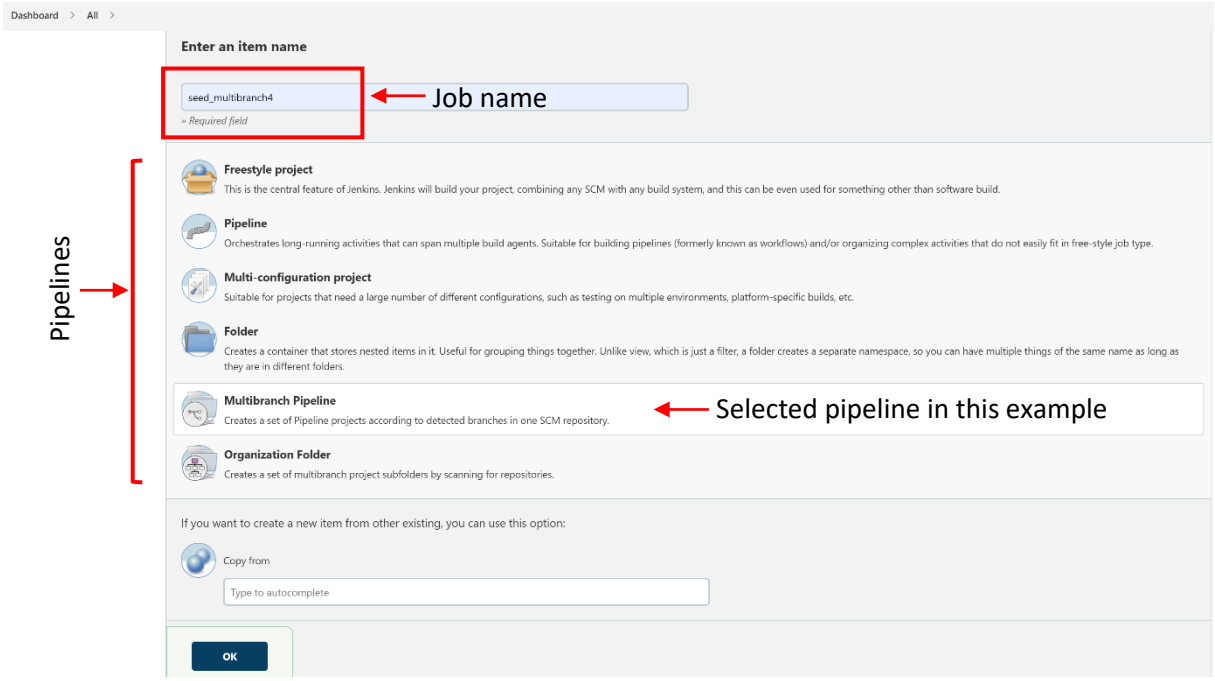
Step:- Important and the fundamental part of pipeline is Step. Pipelines are made up of multiple steps that allow you to build, test and deploy applications. It tells Jenkins what exactly it has to perform. Scripted pipeline doesn't specifically describe steps as part of it's syntax, but in the pipeline steps reference document, steps involved in the pipeline and it's plugin are described in detail. Steps will allow you to automate the process. When all the steps in the pipeline are completed, then it's considered to be successfully executed the pipeline.

Triggers:- It defines, the automated way of how pipeline automates the triggers, i.e., after what span of time the pipeline should automatically look for the changes in the repository. According to Jenkins documentation, cron, pollSCM, upstream are the triggers available.

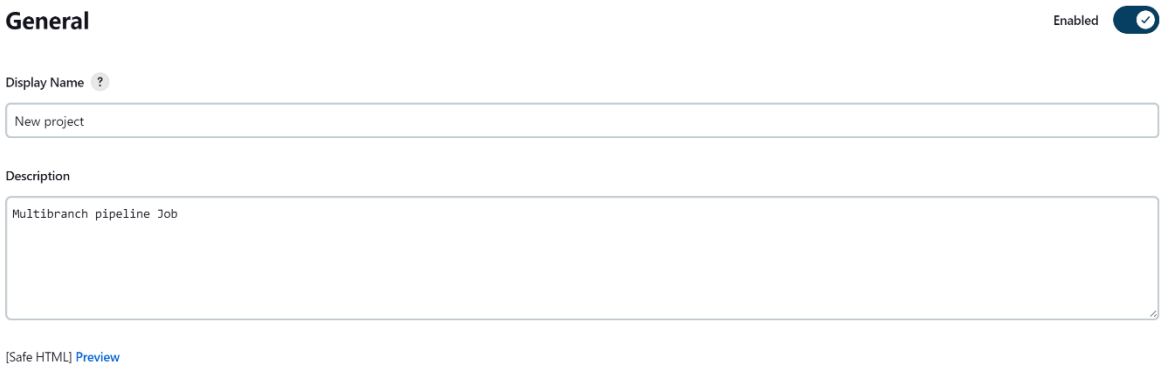
Jenkins uses Job-DSL plugin to automate creation of job using Groovy file. Jenkins is good for building, testing, deploying, as the number of jobs increases, maintaining them becomes difficult. So, Job-DSL plugin helps to solve this problem by allowing jobs to be defined in automated and programmatic form in human readable file. Job-DSL helps in creating multiple jobs under one main/seed job.

How create automated job using Job-DSL:-

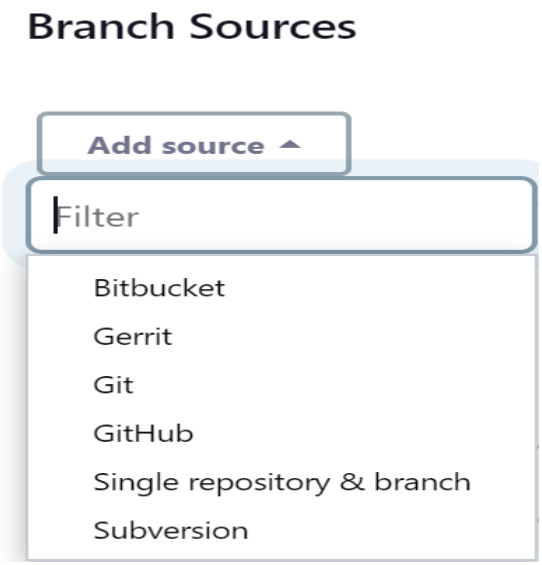
1. Install Job-DSL plugin, Dashboard>Manage Jenkins>Manage Plugins>under Available section, search for Job-DSL plugin>Select it>Click “Install without restart”. The plugin will be installed.
2. After the installation of plugin, in Dashboard>click on “New Item”>Enter the name of the item by selecting the respective pipelines (Freestyle project or Maven project or Multibranch Pipeline or Pipeline etc.,) based the application>then click ok. The main/seed job will be created and leads to Configuration page.



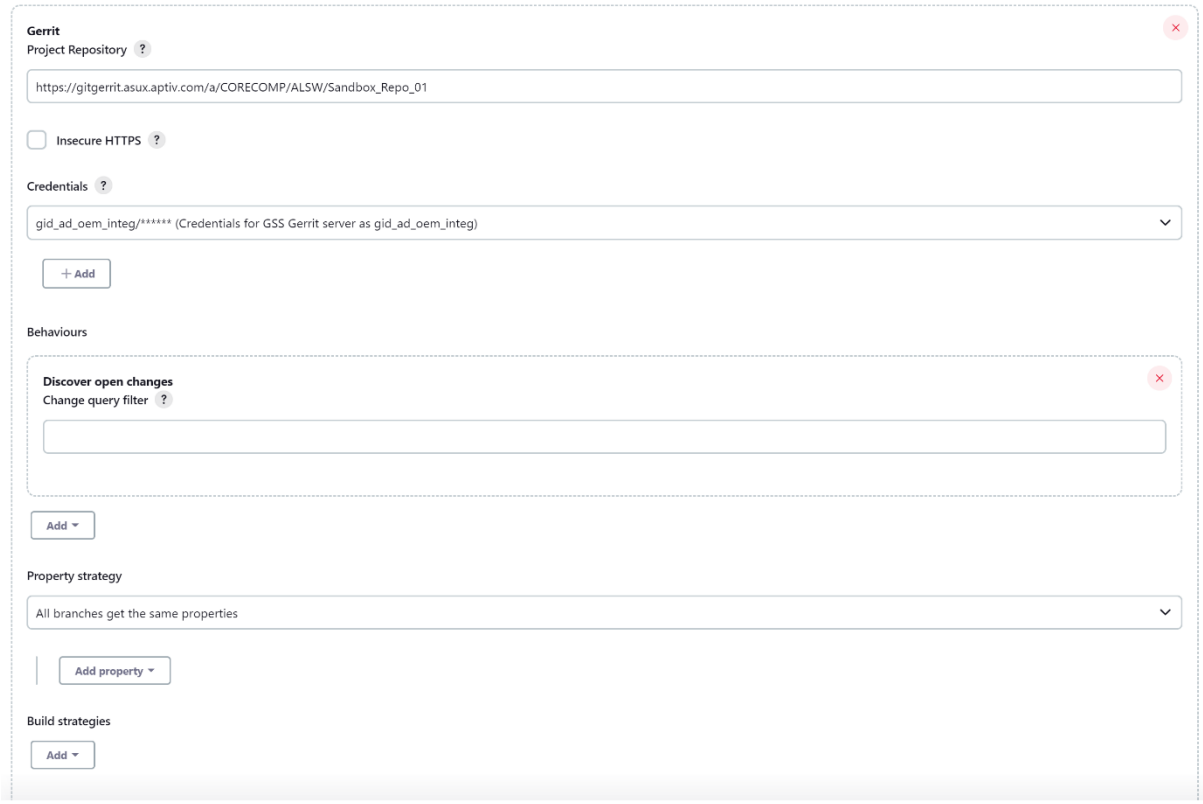
3. Configuration page is used to configure the job for building, testing and deploying. It helps us to add the Source code management, add Build triggers, add Post-build actions etc., let’s take an example of multibranch pipeline job.
   General:- Display name and description can be given about the job or the project (it can be optional).



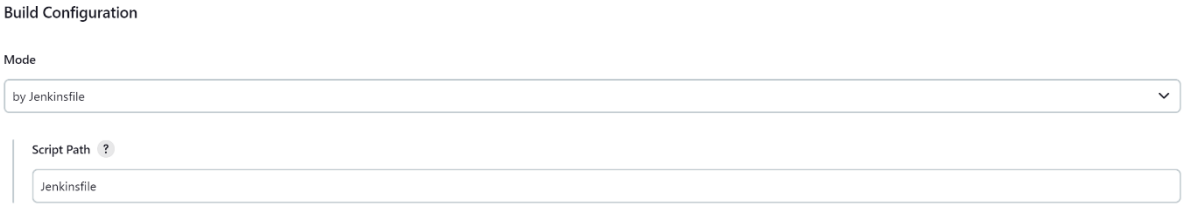
Branch Sources:- addition of Source code management is done.



Source code management used in this example is Gerrit, so, branch sources is selected to be as Gerrit. For the respective selected branch source, certain piece of information has to be given to link between Jenkins and Gerrit, i.e., Gerrit project repository (specify the URL of the remote repository), Credentials (is specified to access the repository, where it scans the branches and checkout the sources), the rest of the information is left as it is, but one can add based on the application.

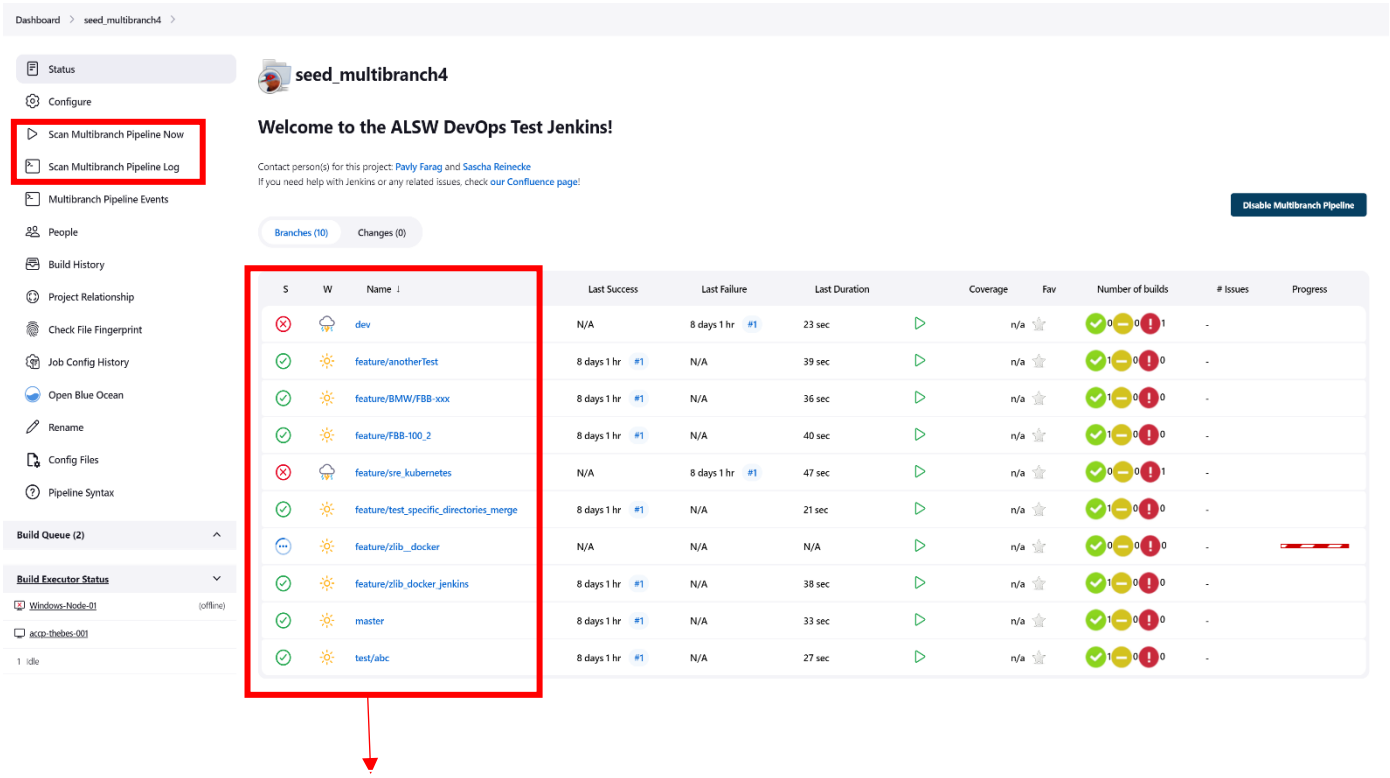


Build Configuration:- Here we have to specify the relative location of the pipeline script (Jenkinsfile) which is present in the repository. The pipeline script will always run inside Groovy sandbox. By default, script path will be Jenkinsfile, if it's left empty.



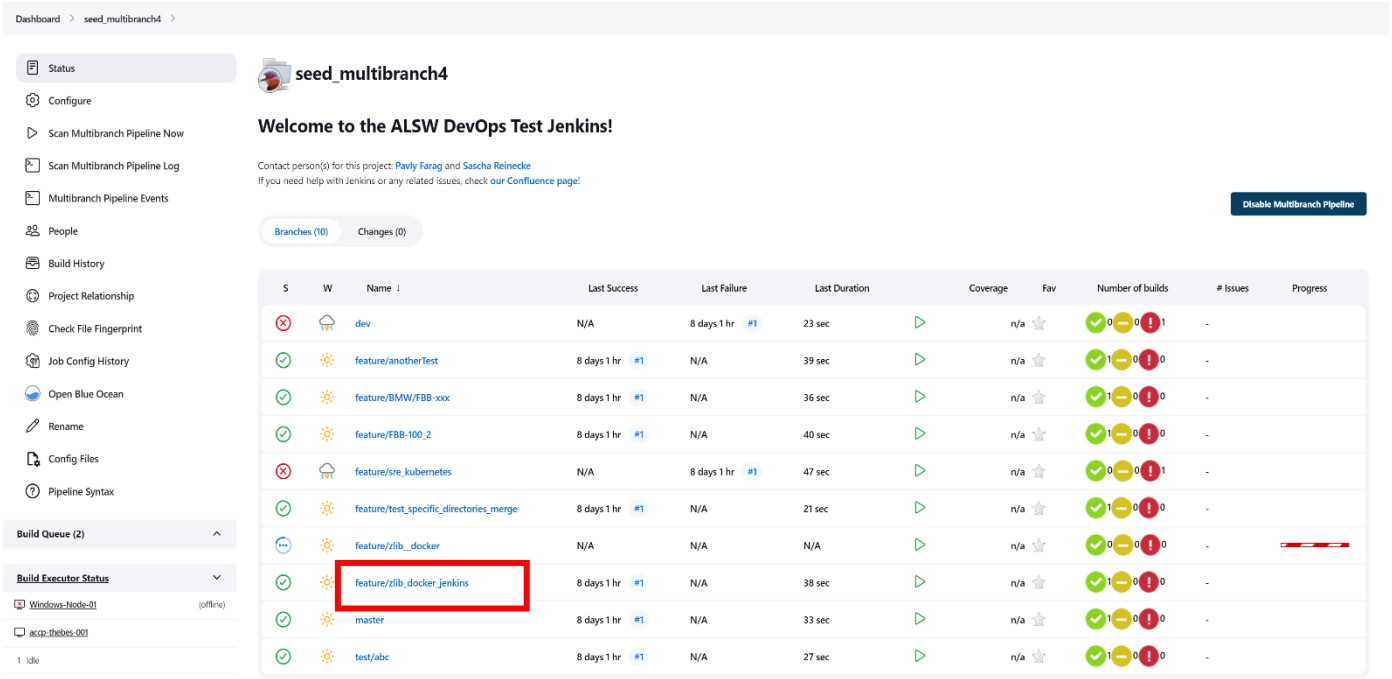
The rest of the configuration has been unchanged, it can be changed or added based on your applications.
Click “Apply”>click “Save”
The configuration for the respective job is done.

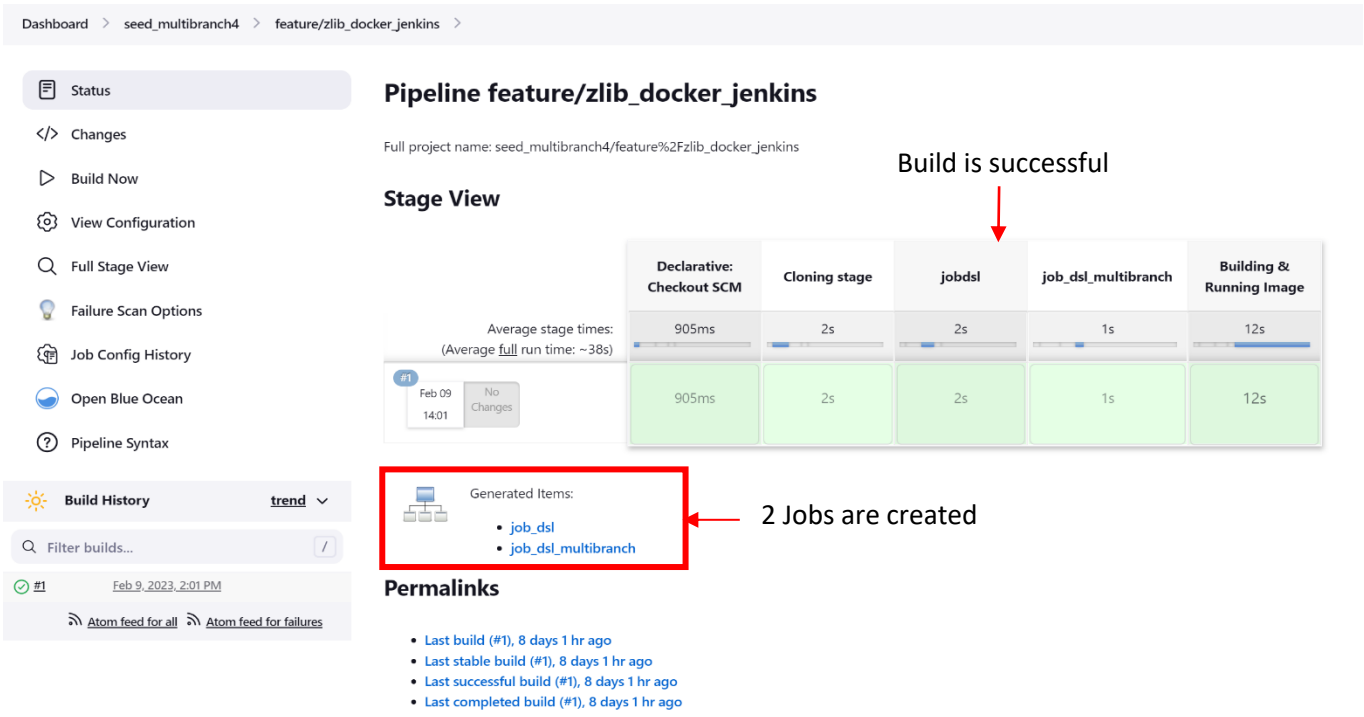
4. Now, when the pipeline is scanned (using, “Scan Multibranch Pipeline Now”), if the scan is successful (which can been seen in “Scan Multibranch Pipeline Log”), all the branches, can be seen inside this job folder.



Branches in my Gerrit repository

5. Once the respective branch is build, if the build is successful, then the job inside the seed job will be created. So in this example, I’m interested in feature/zlib_docker_jenkins branch, which contains my Jenkinsfile to create jobs automatically and 2 Job-DSL groovy files to create pipeline and multibranch pipeline jobs. Hence, once it’s build, the result can be seen below.





So, by using the Job-DSL plugin, 2 automated jobs are created, where one job is pipeline (job_dsl) and the other is multibranch pipeline (job_dsl_multibranch).

With the Jenkinsfile, Job-DSL groovy files for pipeline job and multibranch pipeline job are shown below.

Jenkinsfile:-

```
pipeline {
        agent {docker{image 'jfrog.asux.aptiv.com/corecomp_devops-aptiv-00000000-docker-local/studenttest:1.0'}}

        stages {
                stage('Cloning stage') {
                        steps {
                                sh 'git clone http://github.com/madler/zlib.git'
                                sh "pwd"
                                dir("zlib"){
                                        sh "pwd"
                                }
                        }
                }

                stage('jobdsl'){
                        steps{
                                jobDsl(targets:'jobdsl.groovy')
                        }
                }

                stage('job_dsl_multibranch'){
                        steps{
                                jobDsl(targets:'job_dsl_multibranch.groovy')
                        }
                }

                stage('Building & Running Image') {
                        steps {
                                dir("zlib"){
                                        sh './configure'
                                        sh 'make test'
                                }
                        }
                }
        }
}
```

Job-DSL groovy file for Pipeline job:-

```
pipelineJob('job_dsl') {

        def repo = 'https://gitgerrit.asux.aptiv.com/a/CORECOMP/ALSW/Sandbox_Repo_01'

        description("Pipeline for $repo")

        definition {
            cpsScm {
                scm {
                        git {
                                remote { url(repo) }
                                branches('feature/zlib_docker_jenkins')
                                scriptPath('Jenkinsfile')
                                extensions { }
                        }
                }
            }
        }
}
```

Job-DSL groovy file for Multibranch pipeline job:-

```
multibranchPipelineJob('job_dsl_multibranch') {
        def repo = 'https://gitgerrit.asux.aptiv.com/a/CORECOMP/ALSW/Sandbox_Repo_01'
        def cred = 'GSS-GERRIT-gid_ad_oem_integ'
        description("Multibranch Pipeline for $repo")

        branchSources{
                branchSource{
                        source{
                                gerrit{
                                        id('5caad651-12fb-47da-8cb9-ca0916c65315')
                                        remote(repo)
                                        credentialsId(cred)
                                }
                        }
                }
        }

        factory {
                workflowBranchProjectFactory {
                        scriptPath('Jenkinsfile')
                }
        }

        orphanedItemStrategy {
                discardOldItems {
                        numToKeep(7)
                }
        }
}
```