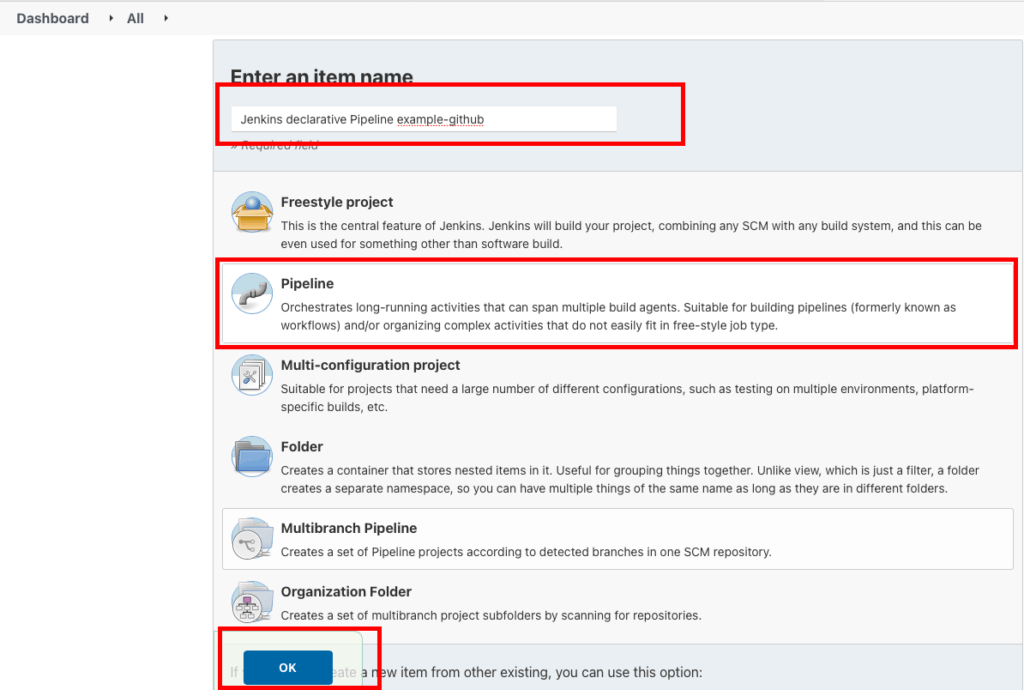


Jenkins Pipeline Github

## Jenkins Declarative Pipeline github

NE kete session do te download repon e git ne doker dhe do te exec bash script me permbaitien “Hello World”. Krijojm nje pipeline te ri:  
New Item > Pipeline > OK  


Dhe ne sesionin e pipeline vendosim kodin me posht:  


“””

pipeline {

agent any

stages {

stage('Build') {

steps {

// Get some code from a GitHub repository

git url: 'https://github.com/glejnhithi/jenkins-devops-tutorial.git', branch: 'main'

// Change file permisson

sh "chmod +x -R ./jenkins"

// Run shell script

sh "./jenkins/script/scripted\_pipeline\_ex\_2.sh"

}

}

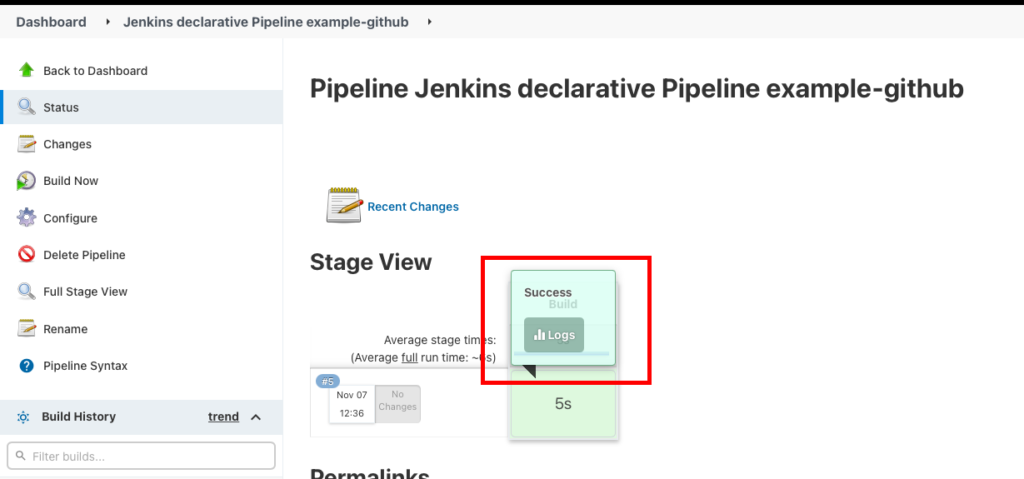
}

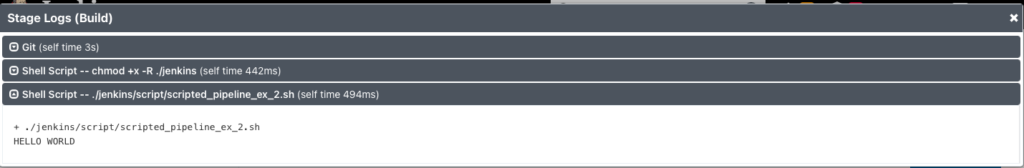
}

“”””

Save & Build Now

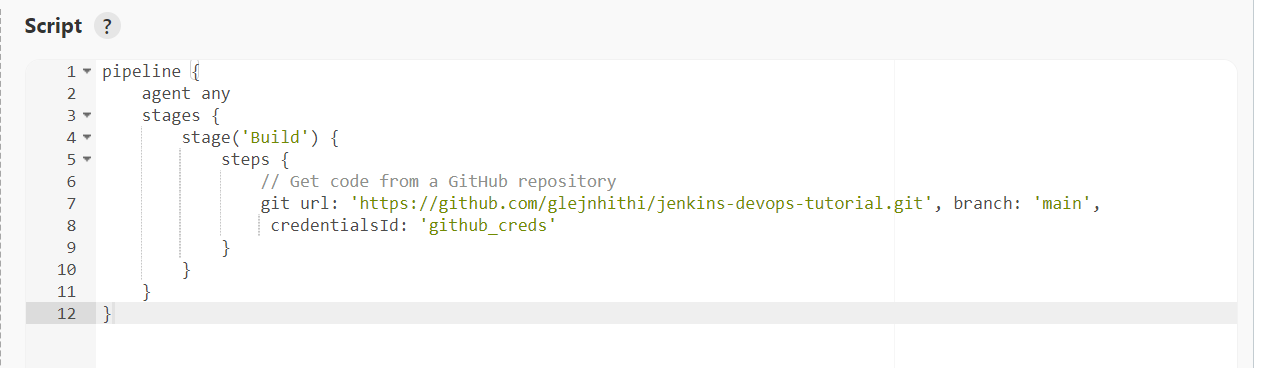
Rezultat I output te job do te jete “HELLO WORLD”





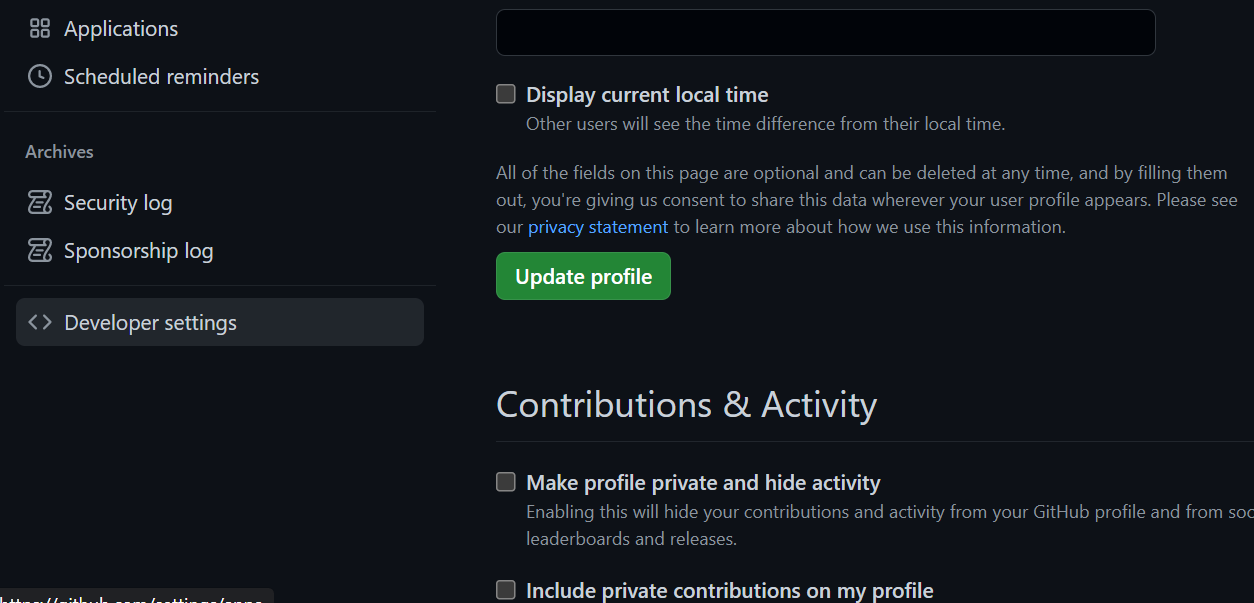
## Jenkins pipeline github with credentials

Nede repo eshte privare na nevoiten kredenciale per te aksesuar ate.Me poshte do te arrim te krijojm kto kredenciale ne jendins dhe ti perdorim ne pipeline.

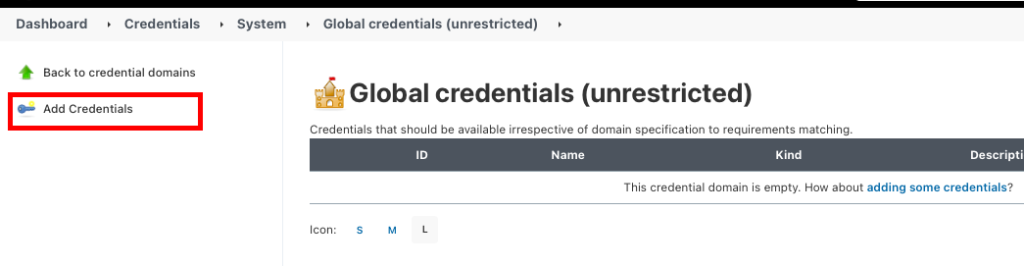


### Jenkins github add credentials

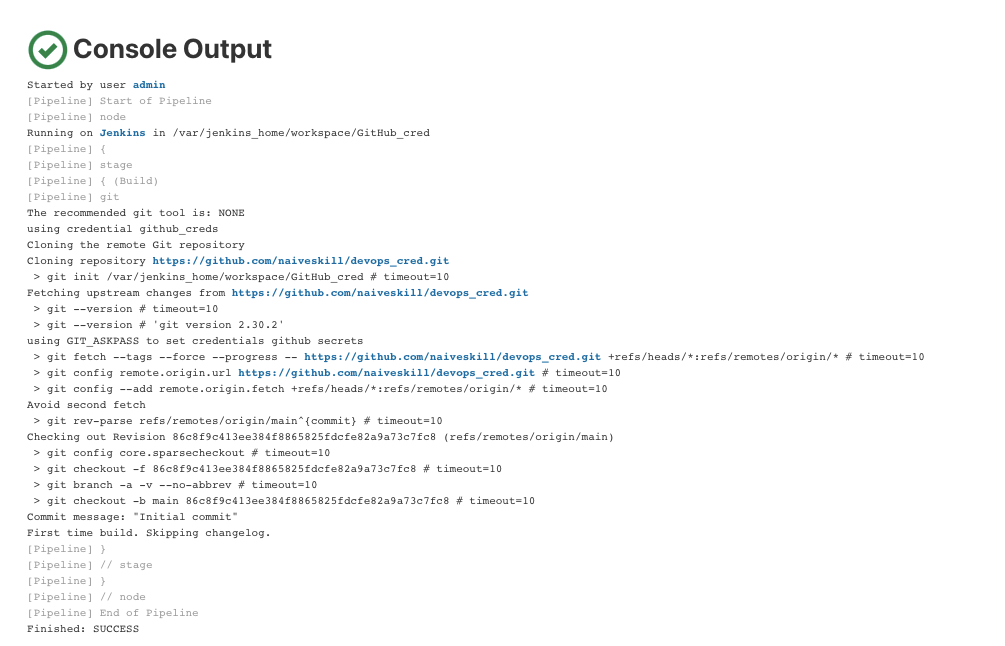
Per te krijuar nje kredencioa ne na duhet e gjenerojm nje TOKEN, i cili do te sherbej si paasword.  
Per te krijuar ate Login ne GitHub dhe shkojm te settings.  


Settings > Developer > Personal access token > Tokens > Generate new token  


Jenkins > Dashboard> Credential



Save & Build



## Jenkins pipeline git checkout SCM

Per te perdorur SCM na nevoitet github plugging qe mund te perdoret me **Pipeline SCM checkout step.**

**Kjo metodper doret per te bere check repo dhe sygjerohet te jet nje step me vete.  
**

**Save & Build**

## Jenkins sample ci/cd pipeline using github

Ne mund te perdorim Jenkin dhe GitHub per te ndertuar nje CI/CD pipeline.Kto pipeline perdoren per te bere clone,Build,Test & Deploy te app me ndryshimet e fundit ne repo.

CI/CD GitHub

pipeline {

agent any

options {

skipDefaultCheckout true

}

stages {

stage('Clone Repository') {

steps {

git branch: "${env.BRANCH\_NAME}", url: "${env.REPO\_URL}"

}

}

stage('Build') {

steps {

sh './gradlew build'

}

}

stage('Test') {

steps {

sh './gradlew test'

}

}

stage('Deploy') {

steps {

sh './deploy.sh'

}

}

}

post {

always {

githubStatus context: 'continuous-integration/jenkins', state: 'success'

if (env.CHANGE\_ID) {

githubComment message: "The pipeline completed successfully!"

githubLabel labels: ['approved']

}

}

}

}