Jenkins Pipeline As Code

Topics

- Introduction
- Pipeline Basic
- Variables
- parameters
- option sets
- trigger builds
- schedule jobs
- parallel
- post jobs
- tools
- conditional and loop statements
- other examples
- sample maven build
- archive artifacts and finger prints
- uses of credentials option
- checkos_andexecutesteps
- input section
- scm git
- when
- sample local deployment

Introduction

Pipeline script

- Another way of job configurartion with help of code

Advantages:

- Can divide the jobs into parts (build /test /deploy/..) & each part can run in each agent.
- Parallel execution of stages are easy configure so that we can save time
- Each stage can execute with different version of JDK/MVN versions
- Can retrigger from failed stage
- visualize the build flow
- Build can hold for user input(specific user can eneter, explain LDAP concept)
- Version control,code review
- pause, restart the build
- In multibranch pipeline scripts will automatically create in sunbranches

Types of Pipeline

- Declarative
- scripted

Difference between Declarative and scripted

- Declarative pipeline is a recent addition.
- More simplified and opinionated syntax when compared to scripted

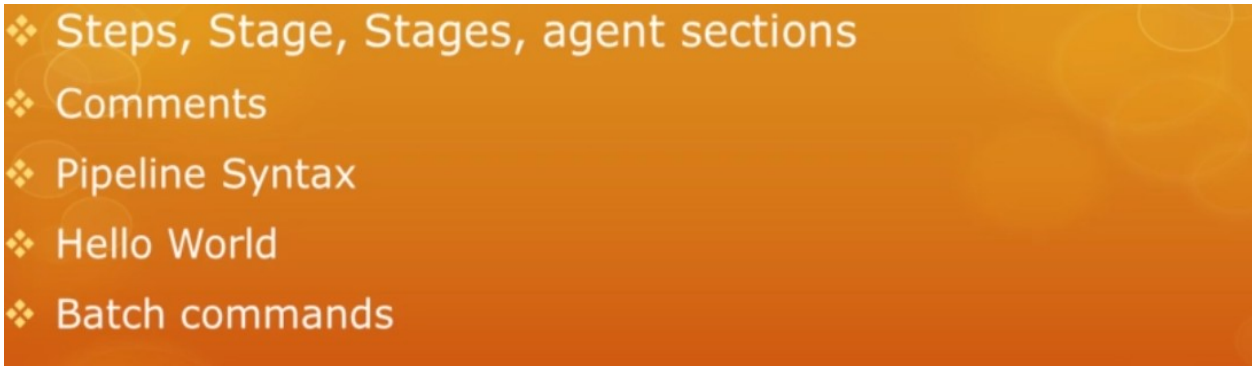
Declarative syntax

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                //
            }
        }
        stage('Test') {
            steps {
                //
            }
        }
        stage('Deploy') {
            steps {
                //
            }
        }
    }
}
```

Scripted Syntax

```
node {
   stage('Build') {
      //
   }
   stage('Test') {
      //
   }
   stage('Deploy') {
      //
   }
}
```

PIPELINE BASIC



Steps

- We need to write step inside the stage directive
- steps contains (command/scripts) that we used in the build
- One steps directive should be there in stage directive

Stage

- Define particular stage (build/test/deploy/..) of our job
- atleast one stage has to be there
- name will be display on the jenkins dashboard

stages

- contains sequence of stages
- atleast one stage has tobe there

Agent

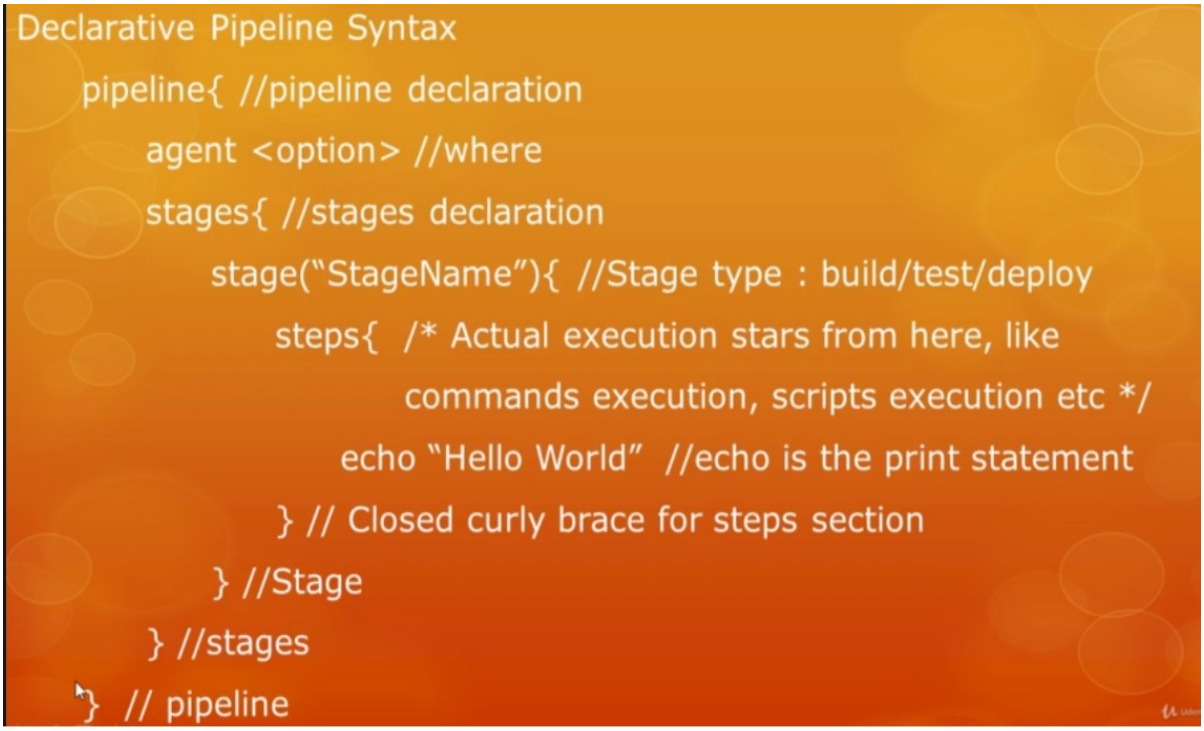
- where (master/slave/container..)we need to run our pipeline script

Stage colors

- White (stage is not executed)
- Green (stage is success)
- Blue lines (stage is executing)
- Redlines or red line (stage is failed)
- Red (fews stage success, any one is failed, few remain sucess stage will show red)

Comments
Single line comment **:** //

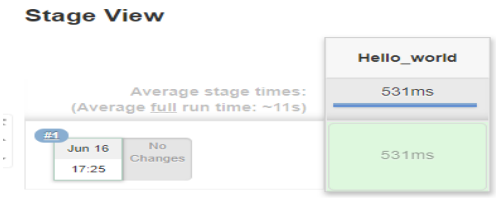
Multiline comment**:** /*
==========
*/



Simple Hello world pipeline:

```
pipeline{
   agent any
   stages{
      stage('Hello_world'){
         steps{
            echo 'Hello world'
         }
      }
   }
}
```

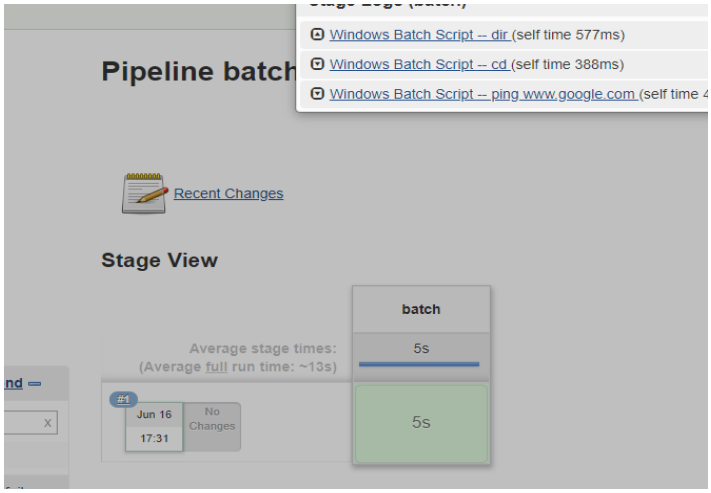
O/P



Batch commands

```
pipeline{
    agent any
    stages{
        stage('batch'){
            steps{
                bat "dir"
                bat "cd"
                bat "ping www.google.com"
            }
        }
    }
}
```

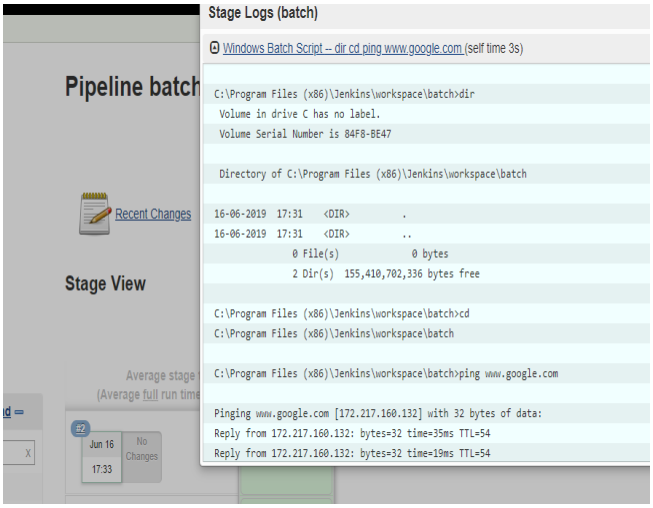
O/P



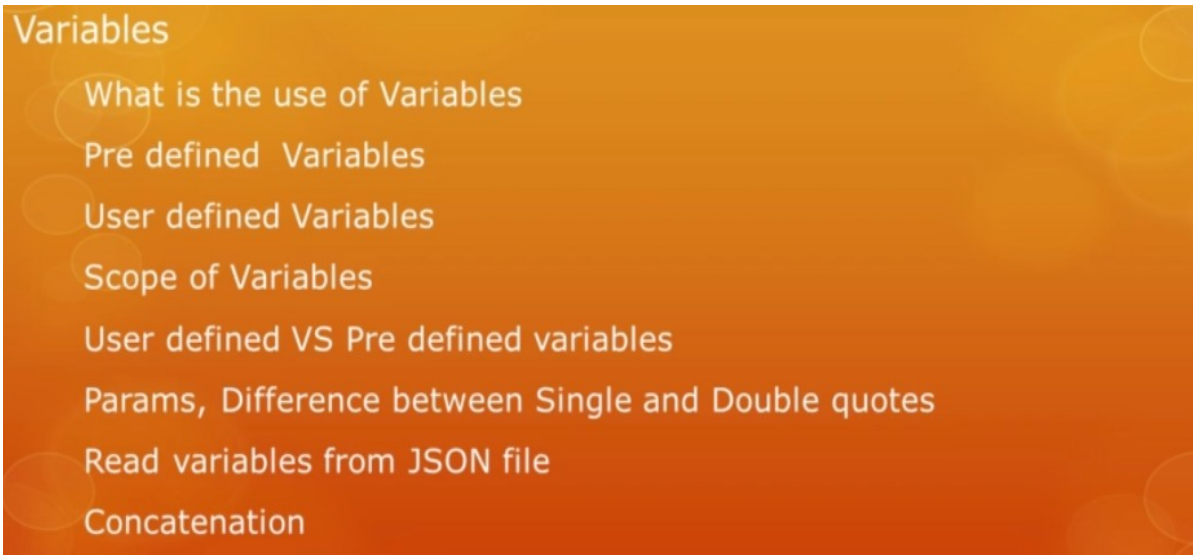
Multiline bat command

```
pipeline{
    agent any
    stages{
        stage('batch'){
            steps{
                bat """
                  dir
                  cd
                  ping www.google.com
                """

            }
        }
    }
}
```

O/P



VARIABLES



What is variable?

Variable is used to store the value.

<variable name> = <variable value>

Types

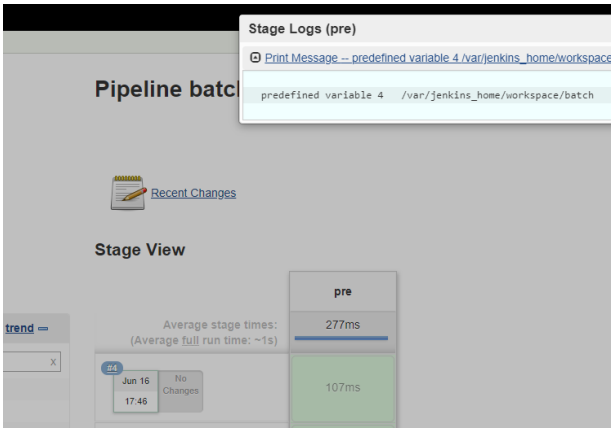
- Predefined variable
- User variable

Predefined:

http://localhost:8080/env-vars.html

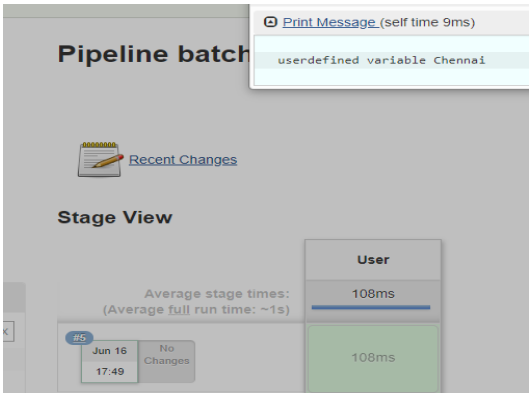
Predefined

```
pipeline{
    agent any
    stages{
        stage('pre'){
            steps{
                echo " predefined variable $BUILD_NUMBER   $WORKSPACE "
            }
        }
    }
}
```



Userdefined : variable we can define in rootlevel or stage level

```
pipeline{
    agent any
    environment{
        MYHOME="Chennai"
    }
    stages{
        stage('User'){
            steps{
                echo " userdefined variable $MYHOME    "
            }
        }
    }
}
```



User defined variables in

- Global level
- stage level
- script level

Global level

```
pipeline{
   agent any
   environment{
      MYHOME="Chennai"
   }
   stages{
      stage('User'){
         steps{
            echo " userdefined variable $MYHOME   "
         }
      }
   }
}
```

Stage level

```
pipeline{
   agent any
   stages{

      stage('User'){
            environment{
             MYHOME="Chennai"
            }
         steps{
            echo " userdefined variable $MYHOME   "
         }
      }
   }
}
```

Script level

```
pipeline{
    agent any
    stages{

        stage('User'){

            steps{
                script{
                 MYHOME="Chennai"

                }
               echo " userdefined variable $MYHOME   "

            }
        }
    }
}
```

```
pipeline{
    agent any
    stages{

        stage('User'){

            steps{
                script{
                 MYHOME="Chennai"
                 echo " userdefined variable $MYHOME   "
                }


            }
        }
    }
}
```

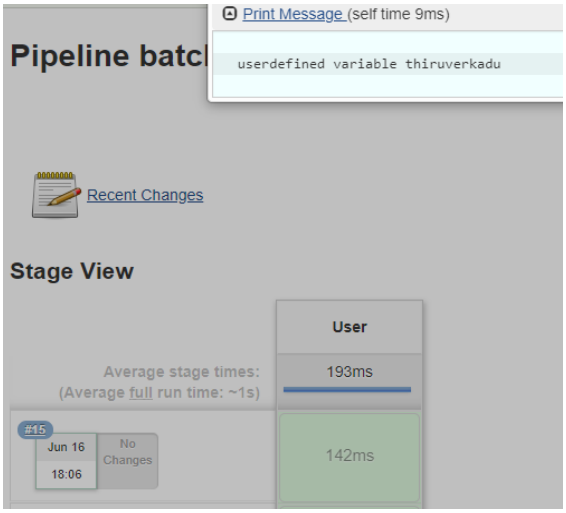
Scope of the Variables : priority order first (script),second(stage),third(global or root)

if you defined the same varaible in global ,stage and , it will pick up stage.

```
pipeline{
    agent any
    environment{
                 MYHOME="Chennai"
                }
    stages{

        stage('User'){
              environment{
                 MYHOME="thiruverkadu"
```

```
            }
        steps{
            echo " userdefined variable $MYHOME   "
        }
    }
  }
}
```

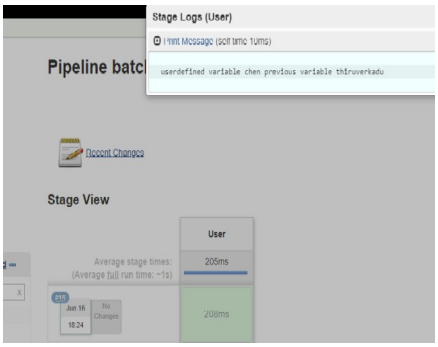
O/P



Predefined vs user defined values:

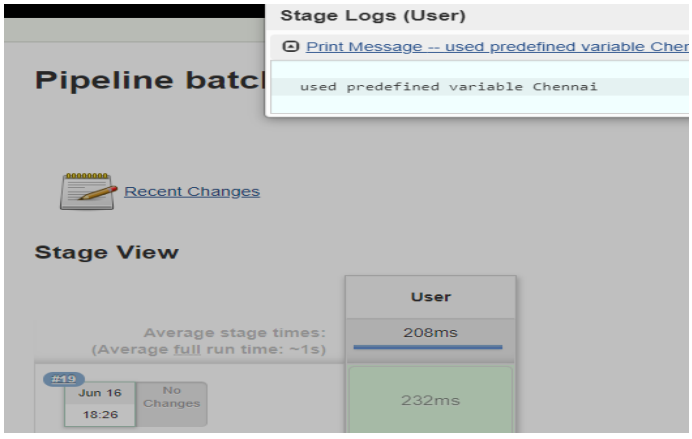
if you defined diff values in variable , we can call above stage variable by ${env.variablename}

```
pipeline{
  agent any
  environment{
            MYHOME="Chennai"
           }
  stages{

    stage('User'){
           environment{
            MYHOME="thiruverkadu"
           }
        steps{
           script{
              MYHOME="chen"
           }
           echo " userdefined variable $MYHOME previous variable ${env.MYHOME}  "
        }
    }
  }
}
```



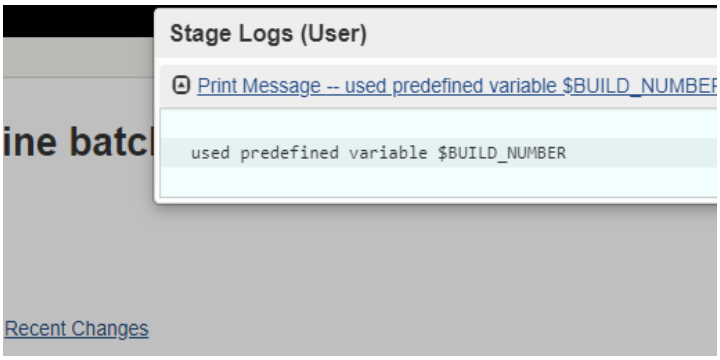
Eventhough it predefined variable if we change for custom, priority for user defined

```
pipeline{
   agent any
   environment{
            BUILD_NUMBER="Chennai"
           }
   stages{

      stage('User'){
           environment{
            MYHOME="thiruverkadu"
           }
         steps{
           script{
              MYHOME="chen"
           }
           echo " used predefined variable $BUILD_NUMBER  "
         }
      }
   }
}
```

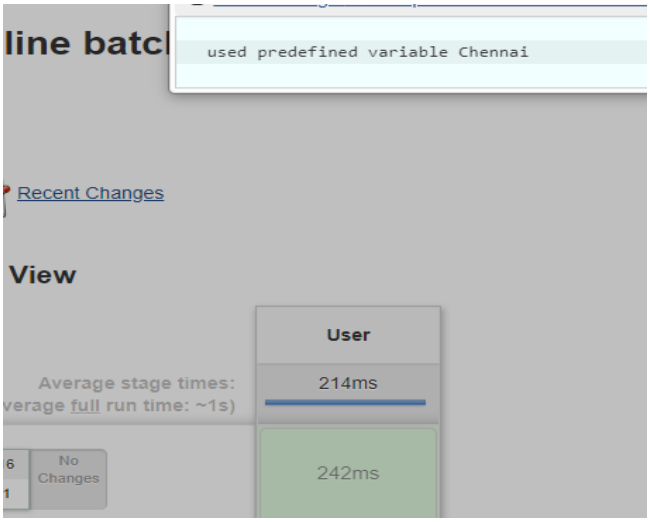


Diff B/W Single and Double quotes

if we defined in single quote it will take as string

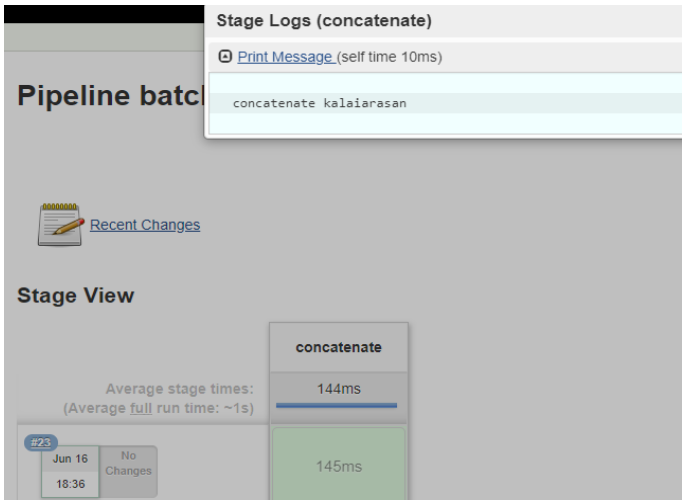


If we defined in double quotes, it will take as variable name



Concatenate
process of combining two or more string by '+' operator in jenkins

```
pipeline{
   agent any
   environment{
            name1="kalai"
            name2='arasan'
           }
   stages{

     stage('concatenate'){

        steps{
           script{
              Name= name1 + name2
           }
           echo " concatenate $Name"
        }
     }
   }
}
```



PARAMETERS

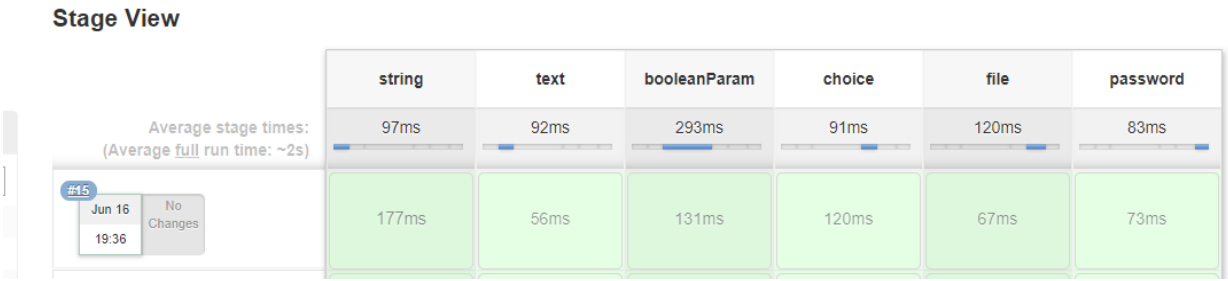
Syntax:

$VARIALENAME and params. VARIALENAME is same.

```
pipeline{
   agent any
  parameters {
                    string(name: 'DEPLOY_ENV', defaultValue: 'staging', description: '')

                    text(name: 'DEPLOY_TEXT', defaultValue: 'One\nTwo\nThree\n', description: '')

                    booleanParam(name: 'TOGGLE', defaultValue: true, description: 'Toggle this value')

      choice(name: 'CHOICE', choices: ['One', 'Two', 'Three'], description: 'Pick something')

      file(name: 'FILE', description: 'Some file to upload')

      password(name: 'PASSWORD', defaultValue: 'SECRET', description: 'A secret password')
```

```
    }

    stages{
        stage('string'){

            steps{
                echo " string $DEPLOY_ENV"
            }
        }
                    stage('text'){

            steps{
                echo " text $DEPLOY_TEXT"
            }
        }
                    stage('booleanParam'){

            steps{
                                script{
                                 if(TOGGLE){
                echo " now execute,  booleann is true"
                                    }else{
                                      echo " Dont execute, boolean is true"
                                    }
            }
                                }
        }
                    stage('choice'){

            steps{
                                script{
                                    if(DEPLOY_ENV=='staging'){
                                            echo " choice $CHOICE"
                                    }
            }
                                }
        }
                    stage('file'){

            steps{
                echo " file $FILE"
            }
        }
                    stage('password'){

            steps{
                echo " password $PASSWORD"
            }
        }
    }
}
```

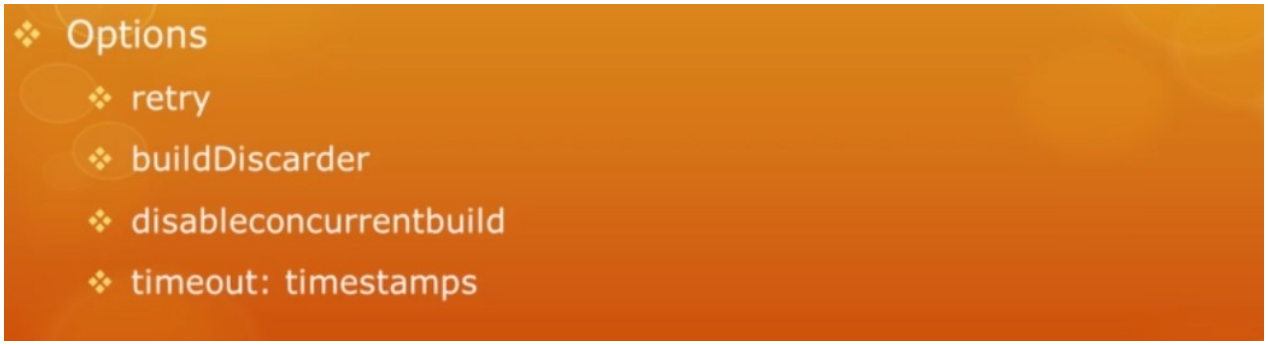
O/P



Dryrun

Dryrun is mainly used for first time of parameter build, before getting build with parameter.

OPTION SET



Options stage level or pipe level

- Retry: before failing the job, will run the job again to specified times
- buildDiscarder : used to delete old build logs in number or days
- disableConcurrentBuilds: used to disable concurrent build

- Timeout:Time set for particular build
- timestamp: will add the time to the build process

Retry Stage based

```
pipeline {
   agent any
   stages {
      stage('Deploy') {
                    options {
                       retry(3)

            timeout(time: 5, unit: 'SECONDS')

          }
         steps {

               sh 'echo hello'
          sleep(10)

         }
      }
   }
}
```

Retry: step based

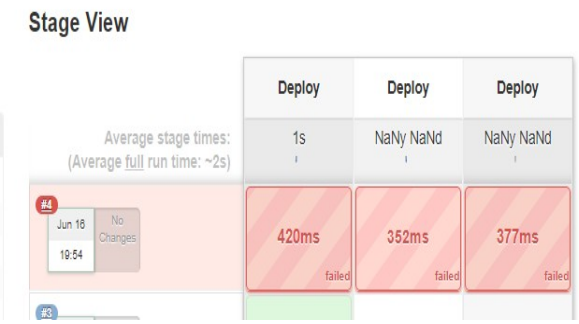
```
pipeline {
   agent any
   stages {
      stage('Deploy') {
         steps {
            retry(3) {
               sh 'echo hello'
            }

         }
      }
   }
}
```

Retry: global based

```
pipeline {
   agent any
         options{
retry(3)
}
   stages {
      stage('Deploy') {
```

```
        steps {

            sh 'echo hello'

        }
    }
  }
}
```

if any eror or timeout it will execute 3 times

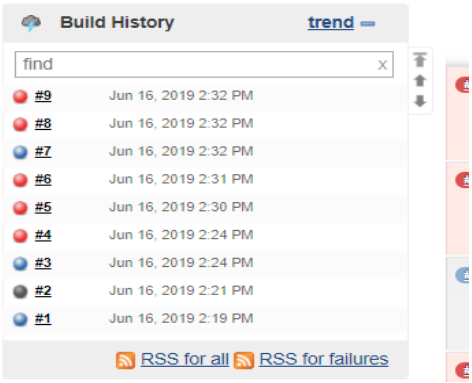


buildDiscarder

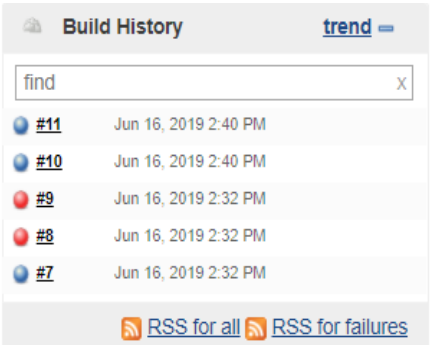
- numbers: options { buildDiscarder(logRotator(numToKeepStr: '5')) }
- days: options {buildDiscarder(logRotator(daysToKeepStr: '7'))})

```
pipeline {
   agent any
 options { buildDiscarder(logRotator(numToKeepStr: '5')) }

   stages {
      stage('Deploy') {

         steps {

            sh 'echo hello'

         }
      }
   }
}
```

before : buildDiscarder execution



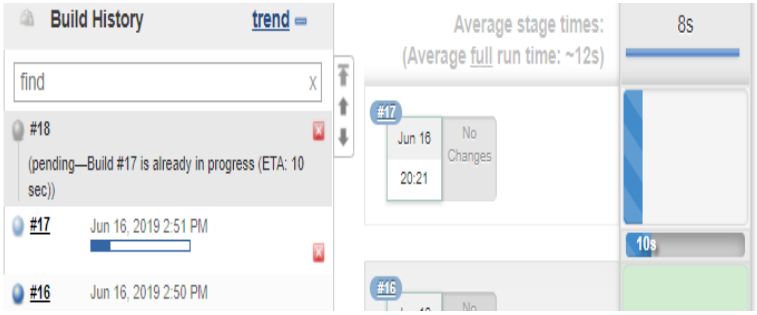
After : buildDiscarder execution



disableConcurrentBuilds

if execute the build if it takes time to complete again paralley , we trigger b4 complete the previous build, again build get start to execute, due to this job will get conflicts with nodes.

```
pipeline {
   agent any
          options {
          buildDiscarder(logRotator(numToKeepStr: '5'))
          disableConcurrentBuilds()

          }
   stages {
      stage('Deploy') {
         steps {

               sh 'echo hello'
                                          sleep(10)

         }
      }
   }
}
```



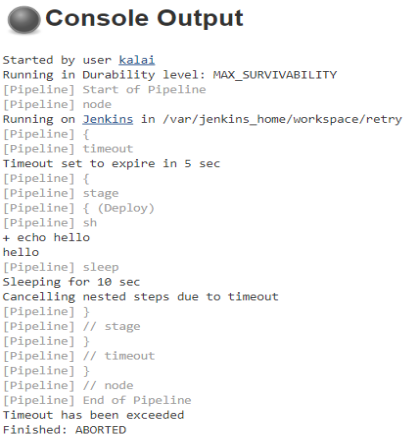
Timeout:

timeout(time: 30, unit: 'MINUTES')
timeout(time: 30, unit: 'SECONDS')
timeout(time: 30, unit: 'HOURS')

Syntax

```
pipeline {
   agent any
          options {
          buildDiscarder(logRotator(numToKeepStr: '5'))
          disableConcurrentBuilds()
          timeout(time: 5, unit: 'SECONDS')

          }
   stages {
      stage('Deploy') {
         steps {

              sh 'echo hello'
            sleep(10)

         }
      }
   }
}
```

its aborted after the timelimit



Timeout Stage based:

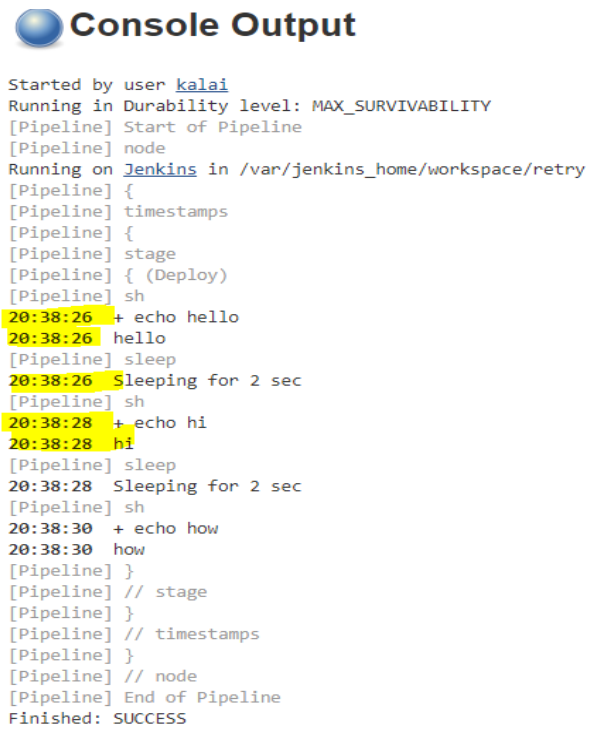
```
pipeline {
   agent any

   stages {
      stage('Deploy') {
                  options {
                     retry(3)

            timeout(time: 5, unit: 'SECONDS')

          }
         steps {

               sh 'echo hello'
                                                sleep(10)

         }
      }
   }
}
```

Timestamp:

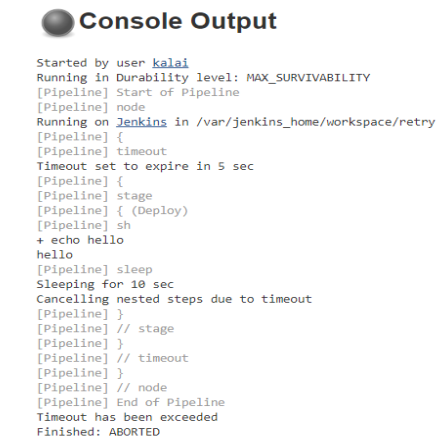
```
pipeline {
   agent any
         options {
         buildDiscarder(logRotator(numToKeepStr: '5'))
         disableConcurrentBuilds()
         timestamps()

         }
   stages {
      stage('Deploy') {
         steps {

               sh 'echo hello'
                  sleep(2)
                  sh 'echo hi'
               sleep(2)
```

```
            sh 'echo how'
           }
       }
   }
}
```

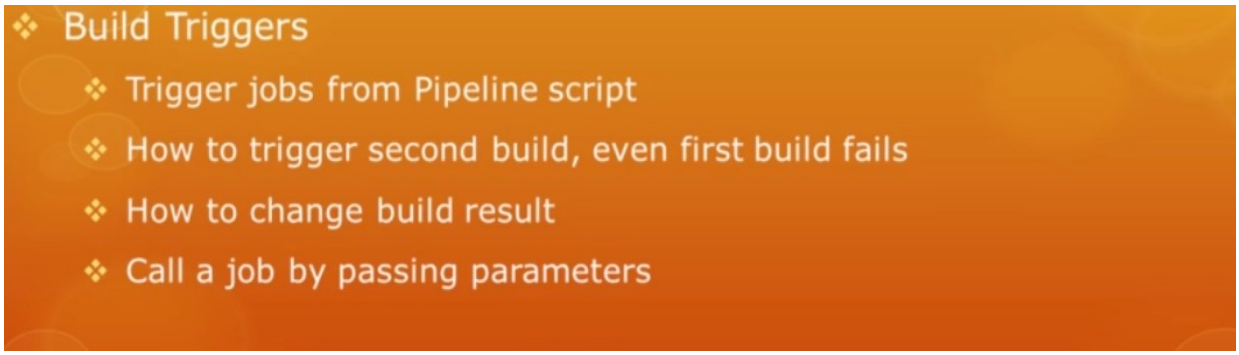
With timestamp



Without timestamp



TRIGGER BUILDS



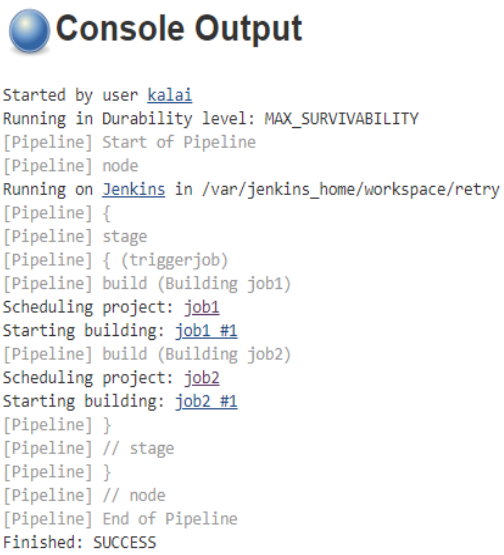
Trigger Other Jobs

- we used build('jobname') option

syntax

```
pipeline {
   agent any

   stages {
      stage('triggerjob') {
         steps {

               build('job1')
               build('job2')
         }
      }
   }
}
```

O/P

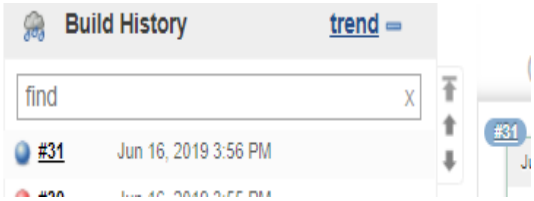


Trigger second job even first job fails

if we triggering two job, if first job got failed, it wont trigger the second job.so we gng say propagate:false, so even though job failed, second job will get trigger.

```
pipeline {
   agent any

   stages {
      stage('triggerjob') {
         steps {

               build(job:'job1', propagate:false)
              build('job2')
         }
      }
   }
}
```

even though job1 failed, its showing succes status.



change build result

while using the below function, it will store the status in jobresult, now eventhough job failed, it will run triffer both job, but it will show unstable result status

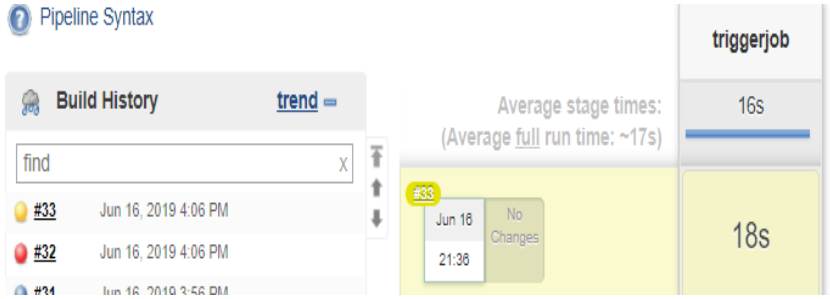
```
jobresult = build(job:'jobname', propagate:false).result
```

syntax

```
pipeline {
   agent any

   stages {
      stage('triggerjob') {
         steps {
             script{
               jobresult = build(job:'job1', propagate:false).result
               build('job2')
               if(jobresult=='FAILURE'){
                currentBuild.result='UNSTABLE'
                                        }
         }
                          }
```

```
    }
  }
}
```

O/P



Trigger other job with parameters

Already job is created, it contains parameter data to build job

Running job pipeline script

```
pipeline {
   agent any
      parameters {
         choice(
            name: 'Nodes',
            choices:"Linux\nMac",
            description: "Choose Node!")
         choice(
            name: 'Versions',
            choices:"3.4\n4.4",
            description: "Build for which version?" )
         string(
            name: 'Path',
            defaultValue:"/home/pencillr/builds/",
            description: "Where to put the build!")
   }
   stages {
      stage("build") {
         steps {
            script {

                                    echo  "$Nodes"
                                    echo  "Versions"
                                    echo "Path"

            }
         }
      }
   }
}
```

triggering job pipeline script:

```
pipeline {
   agent any

   stages {
      stage("build") {
         steps {
            script {

                  build(job: "builder-job",
                  parameters:
                  [string(name: 'Nodes', value: "Linux"),
                  string(name: 'Versions', value: "3.4"),
                  string(name: 'Path', value: "/home/pencillr/builds/}")])


            }
         }
      }
   }
}
```

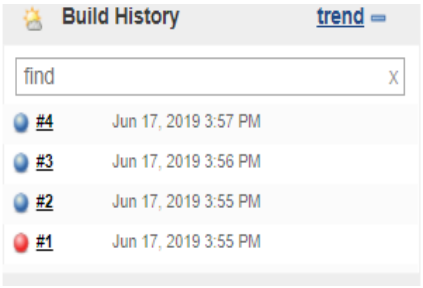
Schedule Jobs

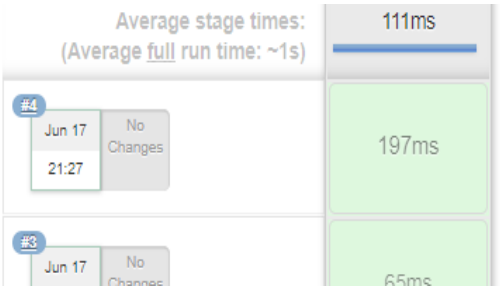
Cron - trigger job will run depends up the cron schedule

```
pipeline {
   agent any
         options{
         timestamps()
         }
   triggers{
          cron('* * * * *')
         }
   stages {
      stage("cron") {
         steps {
```

```
            echo "heloo"
            }
                        }
    }
}
```

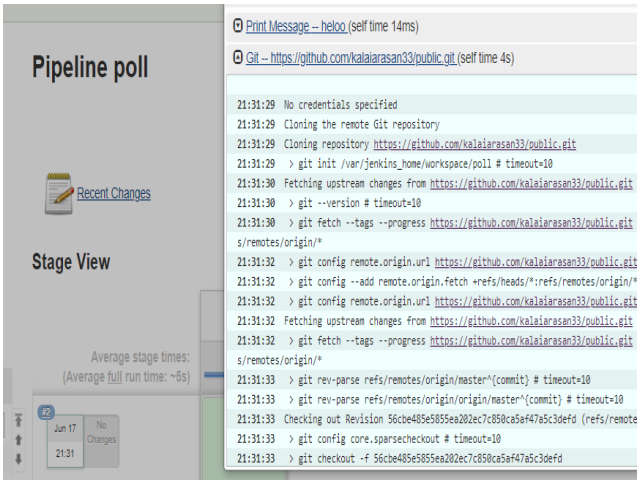
job is running every min



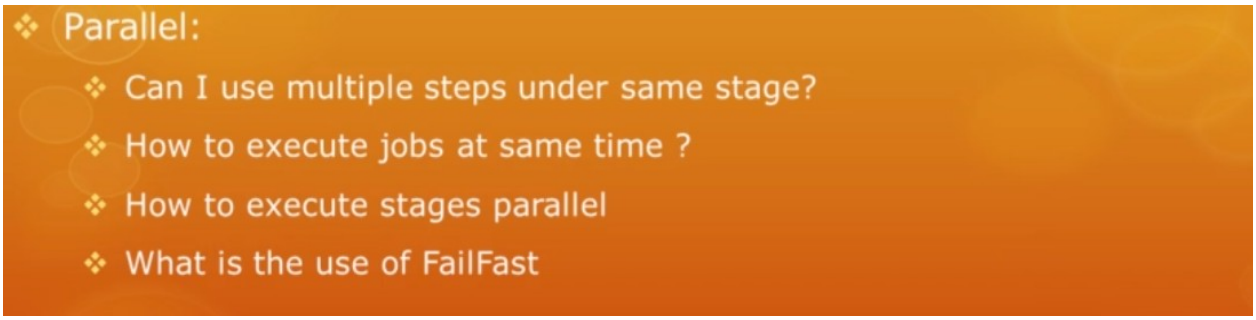


Poll SCM- will trigger the job depends up the changes in code, if there is no commit it wont run.

```
pipeline {
    agent any
            options{
            timestamps()
            }
    triggers{
             pollSCM('* * * * *')
            }
    stages {
        stage("cron") {
            steps {
                echo "heloo"
              git url:"https://github.com/kalaiarasan33/public.git"
                }
            }
        }
    }
```



Parallel



Multiple steps sections under same stage

No we cant use multiple steps in same stage.like below

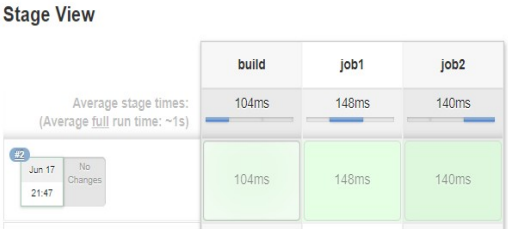
```
stages {
      stage("cron") {
         steps {
               echo "step1"
                  }
       steps {
               echo "step2"
                  }
               }
      }
```

Parallel builds -- it will trigger the build parallely

pipeline {

```
    agent any

    stages {
        stage("build") {
                        parallel{
                            stage('job1'){
                                steps{
                                echo "job1"
                                }
                            }
                            stage('job2'){
                                steps{
                                echo "job2"
                                }
                            }

        }
        }
    }
}
```

build Job is triggering parallely



Parallel stages:

```
pipeline {
    agent any
    options{
        timestamps()
    }

        stages {
            stage("stage1") {
                        parallel{
                            stage('stage1job1'){
                                steps{
                                echo "stage1job1"
                                sleep(10)
                                }
                            }
                                    stage('stage1job2'){
```

```
                steps{
            echo "stage1job2"
            sleep(10)
                }
        }

    }
    stage("stage2") {
            parallel{
            stage('stage2job1'){
            steps{
            echo "stage2job1"
            sleep(5)
            }
        }
            stage('stage2job2'){
                steps{
            echo "stage2job2"
            sleep(5)
                }
        }

    }
  }
 }
}
```

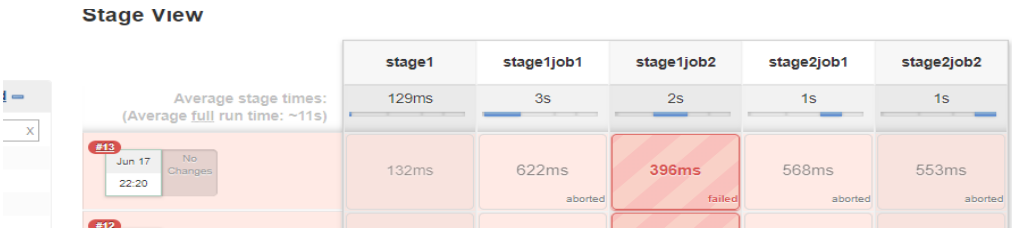
O/P of parallel build

```
[Pipeline] echo
[stage1job1] 22:02:46  stage1job1
[Pipeline] sleep
[stage1job1] 22:02:46  Sleeping for 10 sec
[Pipeline] echo
[stage1job2] 22:02:46  stage1job2
[Pipeline] sleep
[stage1job2] 22:02:46  Sleeping for 10 sec
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // parallel
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (stage2)
[Pipeline] parallel
[Pipeline] { (Branch: stage2job1)
[Pipeline] { (Branch: stage2job2)
[Pipeline] stage
[Pipeline] { (stage2job1)
[Pipeline] stage
[Pipeline] { (stage2job2)
[Pipeline] echo
[stage2job1] 22:02:57  stage2job1
[Pipeline] sleep
[stage2job1] 22:02:57  Sleeping for 5 sec
[Pipeline] echo
[stage2job2] 22:02:57  stage2job2
[Pipeline] sleep
[stage2job2] 22:02:57  Sleeping for 5 sec
[Pipeline] }
```

failFast

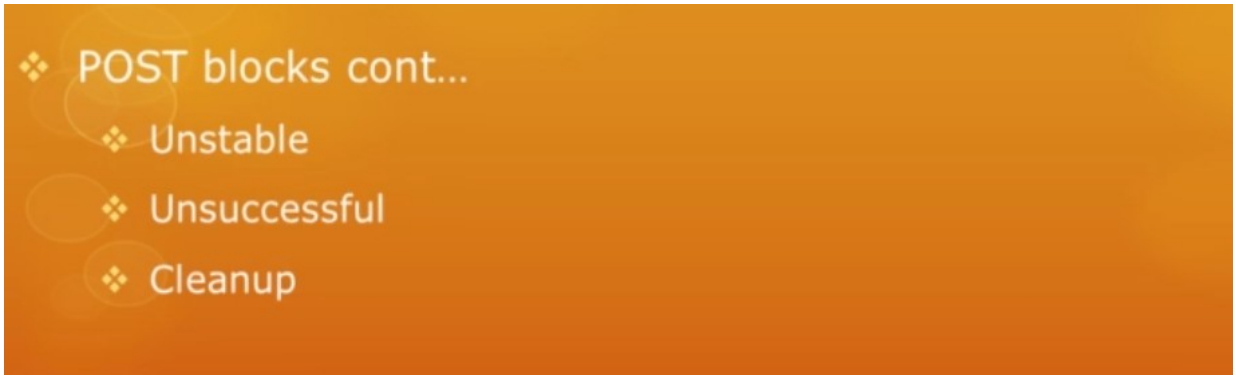
In parallel, eventhough any job is failed, it wont stop, it will execute other job. If we want any job is failed, it should stop the other build means we need to use failFast

```
pipeline {
   agent any
   options{
      timestamps()
   }

       stages {
          stage("stage1") {
             failFast true
                          parallel{
                           stage('stage1job1'){
                             steps{
                             echo "stage1job1"
                             sleep(10)
                              }
                          }
                                stage('stage1job2'){
                                  steps{
                             eecho "stage1job2"
                             sleep(10)
                                   }
                          }
                stage('stage2job1'){
                             steps{
                             echo "stage2job1"
                             sleep(5)
                              }
                          }
                                stage('stage2job2'){
                                  steps{
                             echo "stage2job2"
                             sleep(5)
                                   }
                          }
             }
       }

      }
   }
```



POST JOBS

post will execute after the completion of pipeline's stages section contains the following blocks



Post stage and stages level

Always : Runs always, wont depend upon the build result

changed: Runs only if current build status is changed when compare to previous

Fixed: current status is success and previous status is failed

Regression: if current status is fail/unstable/aborted and previous run is successful.

Aborted: if the current status is aborted

Failure : Runs only if the current build status is failed.

Success : current build is success

Unstable : current build is unstable

cleanup : like always, will execute at every time in the last (if you want to delete any workspace and cleaup any folder , we can use this)

```
pipeline {
   agent any
   options{
      timestamps()
   }

      stages {
         stage("stage1") {
                        steps{
                        sh "ls -l"
                     }
                  post{
                              always{
                                 echo " action  always "
                              }
                              changed{
                                 echo " action always Changed from previous state"
                              }
                              fixed{
                                 echo " action  Fixed when previous state is failure"
                              }
                           regression{
                                 echo " action when current state is fail/unstable/aborted , previous state is
success"
                              }
                              aborted{
                                 echo " action always aborted"
                              }
                              failure{
                                 echo " action always failure"
                              }
                              success{
                                 echo " action always success"
                              }
                              unstable{
                                 echo " action  unstable"
                              }
                              cleanup{
                                 echo " action similar like always , it is using to cleanup folder or
workspace"
                              }


                        }


         }



   }
}
```

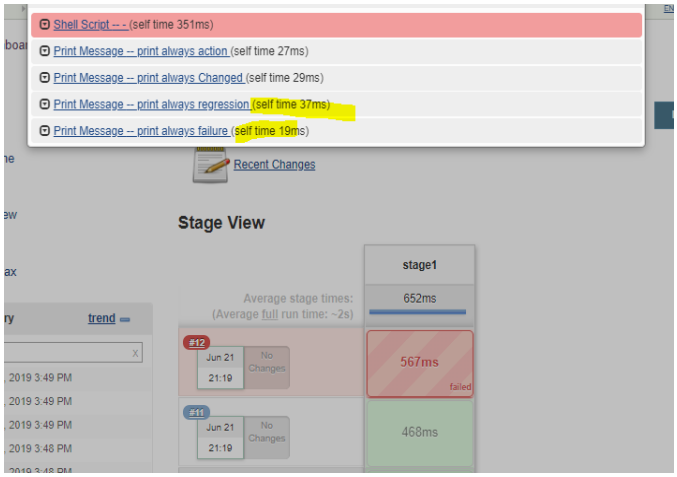
Previous build is failed, current build success O/P.

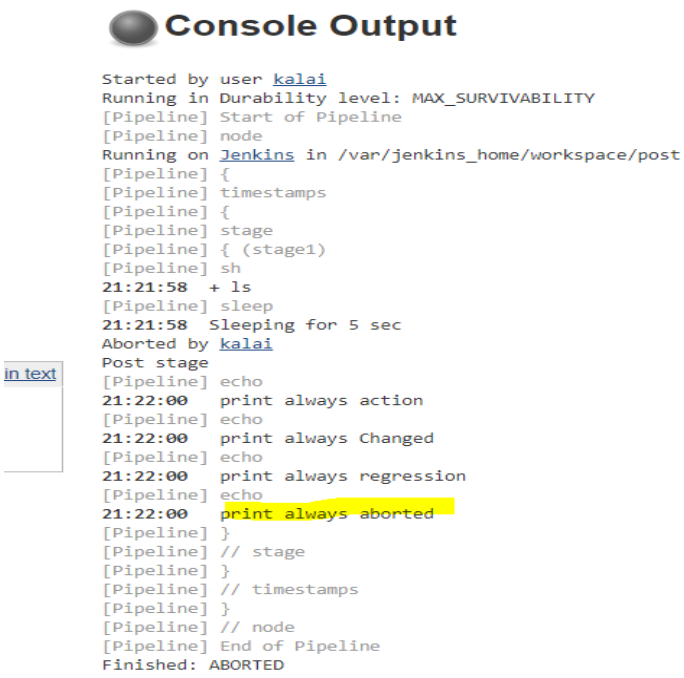
So Always , change (changes in state from previous state), fixed (previous buil failed, current passed), all executed

Previous build is success O/P

So always only executed, there no action for change and fixed


Stage Logs (stage1)
Shell Script -- ls -l (self time 326ms)
Print Message -- print always action (self time 40ms
ne post
ecent Changes
iew
stage1
Average stage times:
821ms
571ms
1s
No Changes

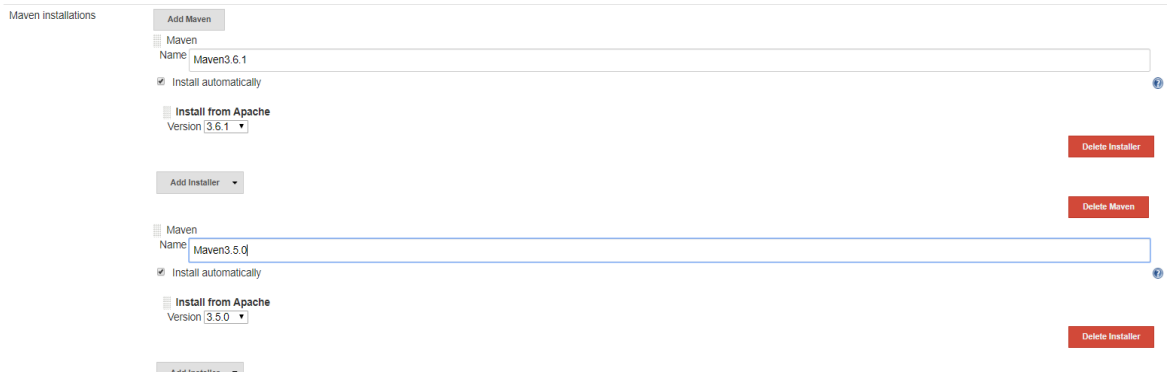

Shell Script -- - (self time 351ms)
Print Message -- print always action (self time 27ms)
Print Message -- print always Changed (self time 29ms)
Print Message -- print always regression (self time 37ms)
Print Message -- print always failure (self time 19ms)
Recent Changes
Stage View
stage1
Average stage times:
(Average full run time: ~2s)
652ms
#12
Jun 21
21:19
No Changes
567ms
failed
#11
Jun 21
21:19
No Changes
468ms
trend



TOOLS

If you want to run specific version of tools to use in pipeline for specific job, so we using tools.

Ex: maven in two version




```
pipeline{
    agent any
    tools{
        maven 'Maven3.6.1'
    }
    stages{
        stage('tools_version'){
            steps{
                sh 'mvn --version'
            }
        }
    }
}
```

O/P

Different version maven in job

```
pipeline{
    agent any
    tools{
        maven 'Maven3.5.0'
    }
    stages{
        stage('tools_version'){
            steps{
                sh 'mvn --version'
            }
        }
    }
}
```

Tools in Stage level :

```
pipeline{
    agent any
    tools{
        maven 'Maven3.6.1'
    }
    stages{
        stage('tools_version'){
            steps{
                sh 'mvn --version'
            }
        }
                stage('diff_version_stage_level'){
                    tools{
                        maven 'Maven3.5.0'
                    }
            steps{
                        echo "stage level"
                sh 'mvn --version'
            }
        }
    }
}
```

```
Running on Jenkins in /var/jenkins_home/workspace/Tools
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (tools_version)
[Pipeline] tool
[Pipeline] envVarsForTool (hide)
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ mvn --version
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T19:00:29Z)
Maven home: /var/jenkins_home/tools/hudson.tasks.Maven_MavenInstallation/Maven3.6.1
Java version: 1.8.0_212, vendor: Oracle Corporation, runtime: /usr/local/openjdk-8/jre
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-51-generic", arch: "amd64", family: "unix"
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (diff_version_stage_level)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] echo
stage level
[Pipeline] sh
+ mvn --version
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-03T19:39:06Z)
Maven home: /var/jenkins_home/tools/hudson.tasks.Maven_MavenInstallation/Maven3.5.0
Java version: 1.8.0_212, vendor: Oracle Corporation
Java home: /usr/local/openjdk-8/jre
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-51-generic", arch: "amd64", family: "unix"
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

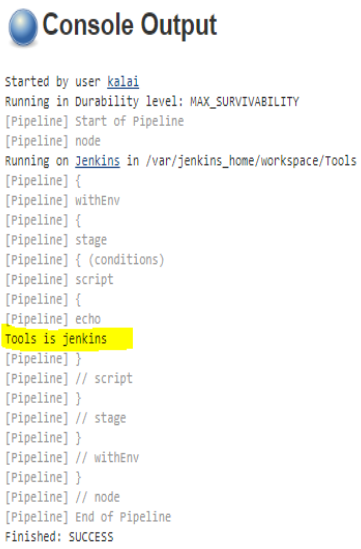
Conditional and Loop Statements

IF Condition:

We can use Groovy coding functionalities using script {...} section.

```
pipeline{
   agent any
   environment{
                Tools='Jenkins'
         }
   stages{
      stage('conditions'){
         steps{
            script{
                                    if(Tools == 'Jenkins'){
                                        echo 'Tools is jenkins'
                                    }else{
                                        echo 'Tools is not jenkins '
                                    }
```

```
                                        }
            }
        }

    }
}
```

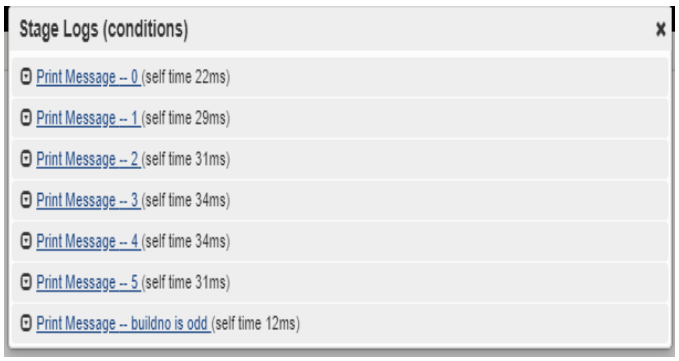


Demo : check build number even or Odd

```
pipeline{
    agent any
    environment{
                    Tools='Jenkins'
            }
    stages{
        stage('conditions'){
            steps{
                script{
                                        int buildno="$BUILD_NUMBER"
                                        if(buildno %2 == 0){
                                                echo 'builno is even'
                                            }else{
                                                echo 'buildno is odd'
                                            }
                                    }
            }
        }

    }
}
```

Demo: For loop

```
pipeline{
   agent any
   environment{
                Tools='Jenkins'
         }
   stages{
      stage('conditions'){
         steps{
            script{
               for(i=0;i<=5;i++){
                  println i

               }
                                    int buildno="$BUILD_NUMBER"
                                    if(buildno %2 == 0){
                                          echo 'builno is even'
                                    }else{
                                          echo 'buildno is odd'
                                    }
                              }
         }
      }

   }
}
```

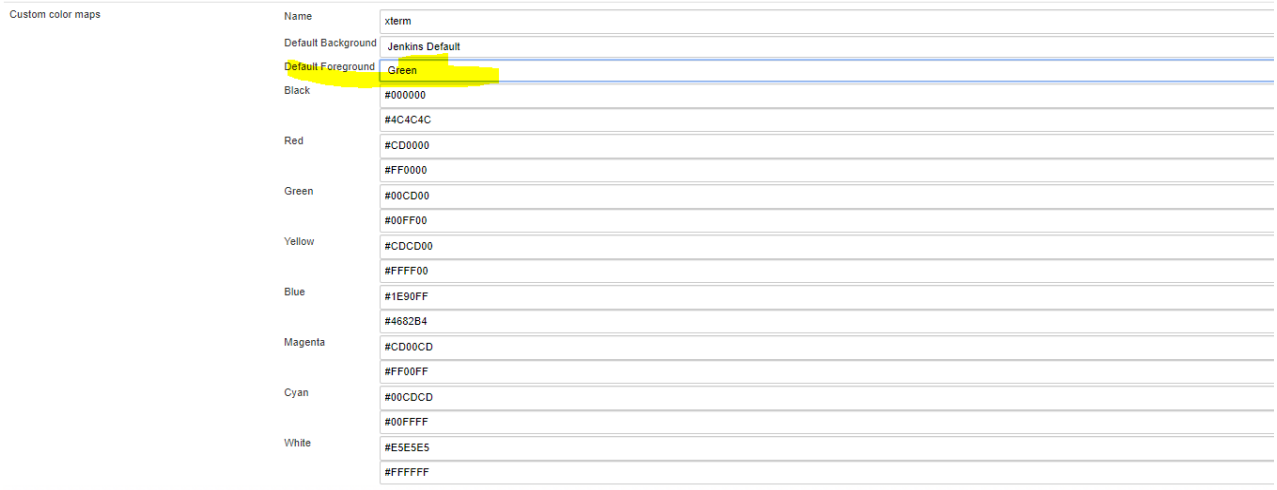


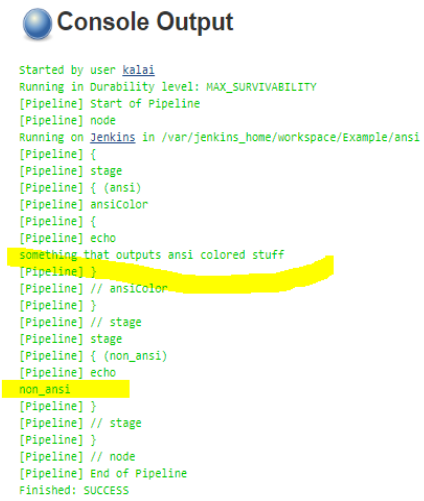
Other Example

Ansicolor:

we need to install the plugin first , then set the ansi in configuration , jenkins foreground

```
pipeline{
    agent any
    stages{
        stage('ansi'){
            steps{
            ansiColor('xterm') {
                echo 'something that outputs ansi colored stuff'
               }
            }

        }
                    stage('non_ansi'){
            steps{
                echo 'non_ansi'
            }

        }
    }
}
```



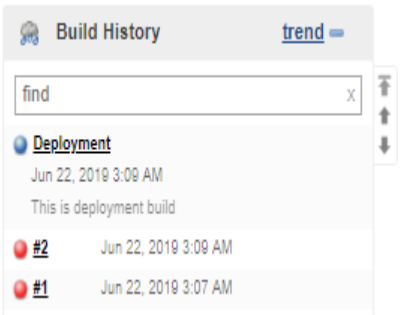


Change Build Number to Name

This is used to define the name for the job and description.

```
pipeline{
   agent any
   stages{
      stage('buid_name'){
         steps{
                                script{
                                   currentBuild.displayName = "Deployment"
                                   currentBuild.description = "This is deployment build"
                                         }
            echo 'build name changing'
         }

      }
   }
}
```

O/P



dir, cleanws

create folder inside workspace -> job name -> (creating folder) -> job output is here

```
kalai@jenlinux:~/jenkins_home/workspace/Example$ cd delete_WS/  --> job name
kalai@jenlinux:~/jenkins_home/workspace/Example/delete_WS$ ls
build_one  build_one@tmp  -> folder we created with dir function
kalai@jenlinux:~/jenkins_home/workspace/Example/delete_WS$ cd build_one
kalai@jenlinux:~/jenkins_home/workspace/Example/delete_WS/build_one$ ls
hello.txt  --> output created
kalai@jenlinux:~/jenkins_home/workspace/Example/delete_WS/build_one$ pwd
/home/kalai/jenkins_home/workspace/Example/delete_WS/build_one
```

Creating output in workspace -> jobname -> build_one --> outputfiles

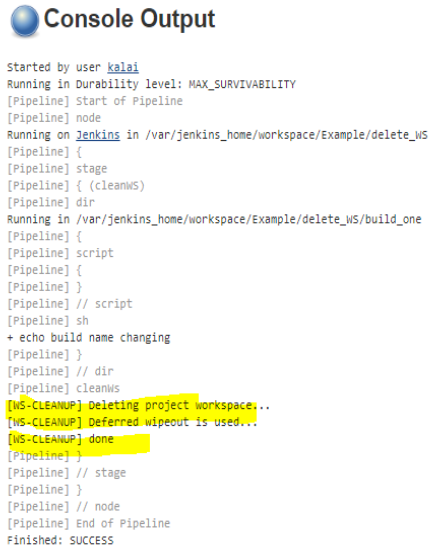
```
pipeline{
   agent any
   stages{
      stage('cleanWS'){
         steps{
                                    dir('build_one'){
                               script{
                                 currentBuild.displayName = "Deployment"
                                       currentBuild.description = "This is deployment build"
                                       }
            sh "echo dir creation and delete WS > hello.txt"
        }

                     }

      }
   }
}
```

Creating output in workspace -> jobname -> build_one --> outputfiles --> deleted job workspace

```
kalai@jenlinux:~/jenkins_home/workspace/Example/delete_WS/build_one/..$ cd ..
kalai@jenlinux:~/jenkins_home/workspace/Example$ ls
change_build_name  change_build_name@tmp
kalai@jenlinux:~/jenkins_home/workspace/Example$
```

```
pipeline{
   agent any
   stages{
      stage('cleanWS'){
         steps{
                                    dir('build_one'){
                               script{
                                 currentBuild.displayName = "Deployment"
                                       currentBuild.description = "This is deployment build"
                                       }
            sh "echo build name changing > hello.txt"
        }
                     cleanWs()
```

```
                }

        }
    }
}
```



Write file_Jenkins syntax

Creating file in jenkins syntax

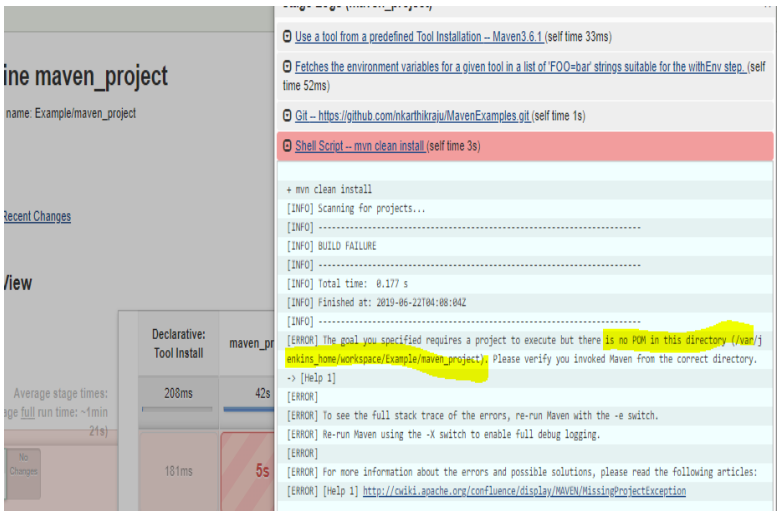
```
pipeline{
    agent any
    stages{
        stage('write_file'){
            steps{

                                writeFile file: 'newfile.txt' ,  text:"my file content is very small"
                                archiveArtifacts '*.txt'


                }

        }
    }
}
```

Sample Maven Build

Build the maven project

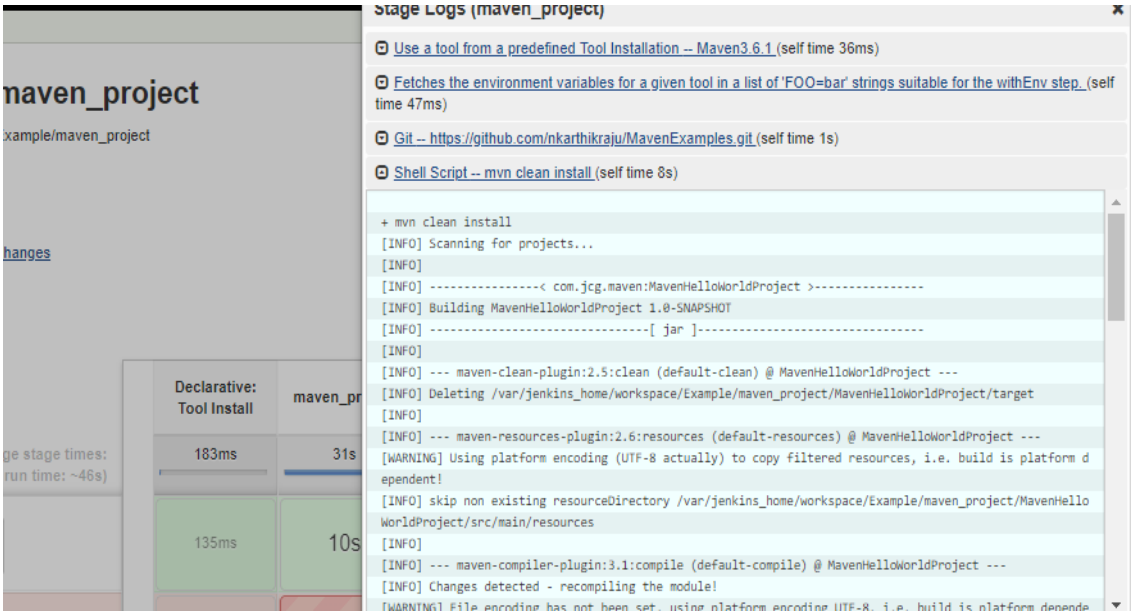
```
pipeline{
    agent any
    tools{
        maven 'Maven3.6.1'
    }
    stages{
        stage('maven_project'){
            steps{
                git url:"https://github.com/nkarthikraju/MavenExamples.git"
                sh  "mvn clean install"
            }
        }
    }
}
```

without moving into the directory , it will show error , no pom file



now moved to directory wth help of dir function then executing maven clean install

```
pipeline{
   agent any
   tools{
      maven 'Maven3.6.1'
   }
   stages{
      stage('maven_project'){
         steps{

            git url:"https://github.com/nkarthikraju/MavenExamples.git"
            dir('MavenHelloWorldProject'){
                                    sh  "mvn clean install"
         }
                        }

      }
   }
}
```

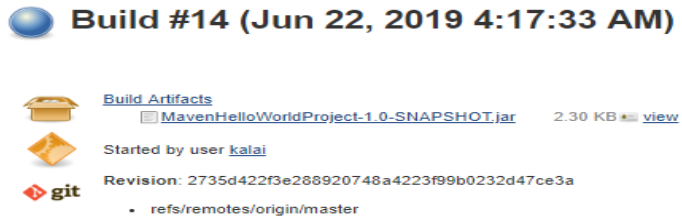


Archive artifacts and finger prints

getting archiveArtifacts when build is success

```
pipeline{
   agent any
   tools{
      maven 'Maven3.6.1'
   }
   stages{
      stage('maven_project'){
         steps{
```

```
            git url:"https://github.com/nkarthikraju/MavenExamples.git"
            dir('MavenHelloWorldProject'){
                                    sh  "mvn clean install"
        }
                }
        post{
                    success{
                        archiveArtifacts "MavenHelloWorldProject/target/*.jar"
                        }
                }
    }
}
```

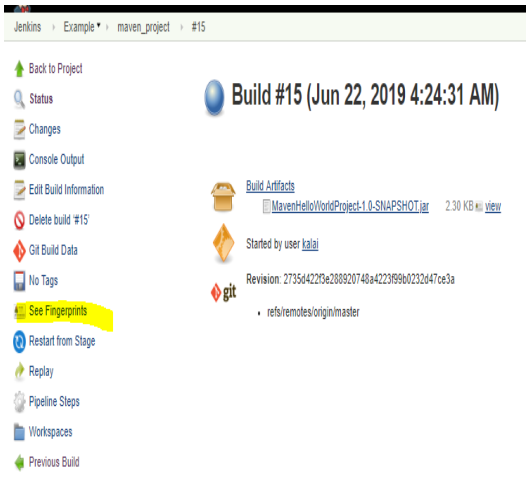


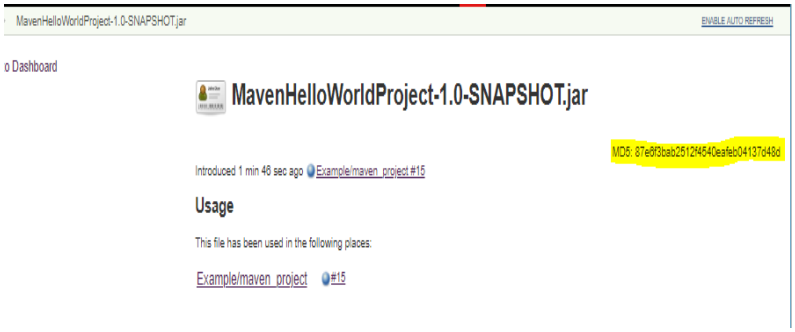
Fingerprint:

if we execute the same job 10 times or etc, it will give same name output , for the identificaton or record the artifacts with fingerprint it will create checksum with build.

```
pipeline{
    agent any
    tools{
        maven 'Maven3.6.1'
    }
    stages{
        stage('maven_project'){
            steps{

            git url:"https://github.com/nkarthikraju/MavenExamples.git"
            dir('MavenHelloWorldProject'){
                                    sh  "mvn clean install"
        }
                }
        post{
                    success{
                        archiveArtifacts artifacts:"MavenHelloWorldProject/target/*.jar" , fingerprint:true
                        }
                }
    }
}
```

O/P

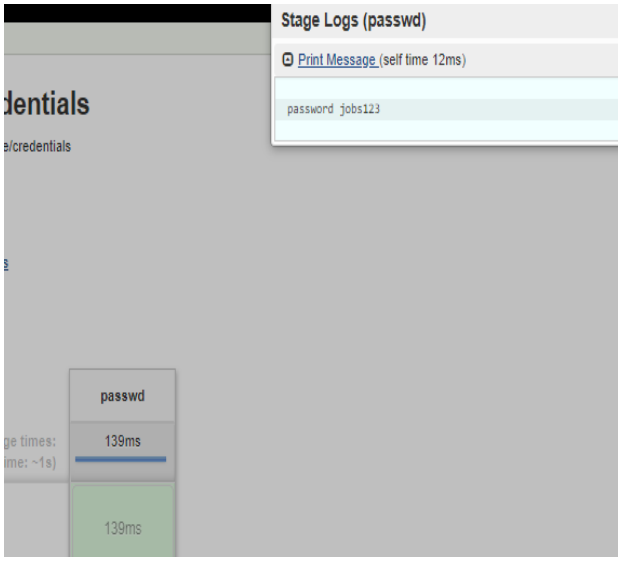




Uses of Credentials Option

if we pass the password it will transparent

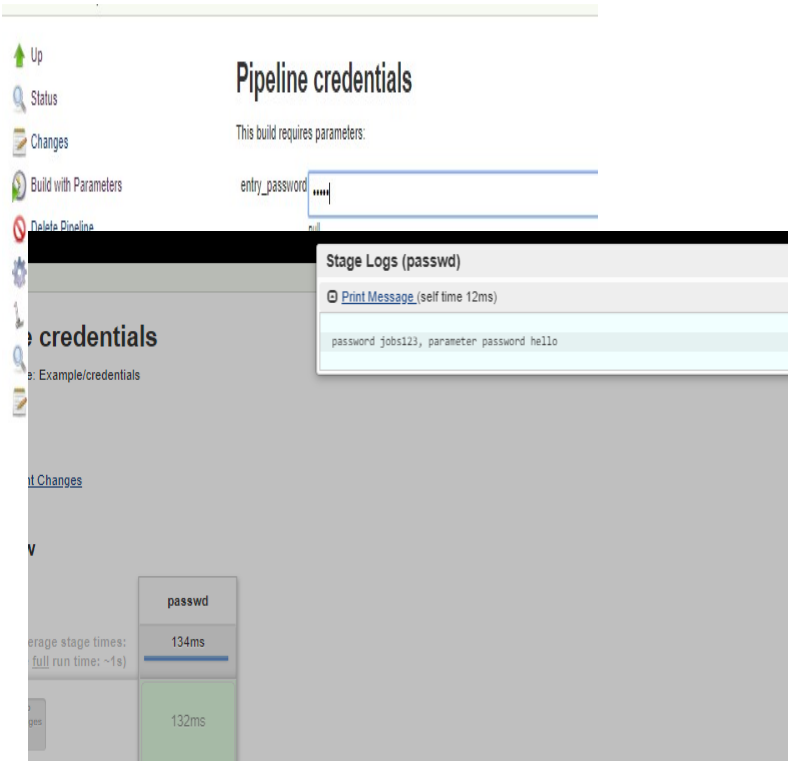
```
pipeline{
    agent any
    environment{
        pass="jobs123"
    }
    stages{
        stage('passwd'){
            steps{
                echo "password $pass"
            }
        }
    }
}
```



another option passing the password as parameter with (password parameter)

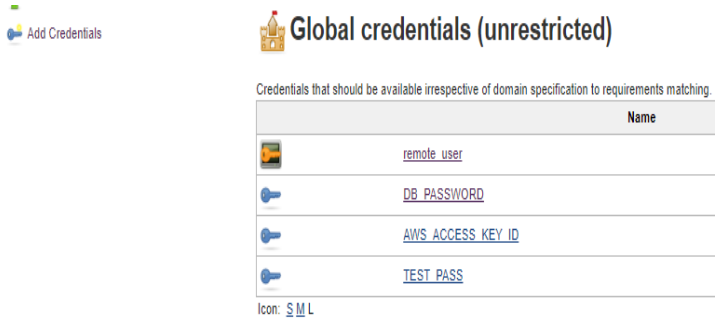
```
pipeline{
    agent any
    environment{
        pass="jobs123"
    }
    parameters{
        password(name:'entry_password')
    }
    stages{
        stage('passwd'){
            steps{
                echo "password $pass, password parameter $entry_password"
            }
        }
    }
}
```

O/P

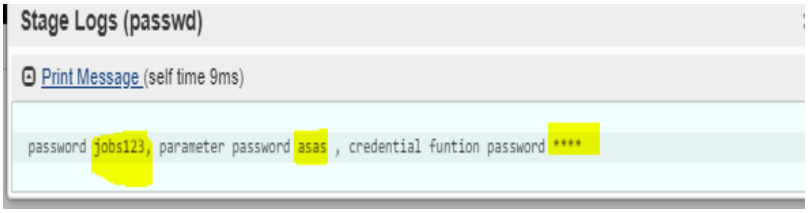


Now password with credential function

create sceret text in credentials in jenkins



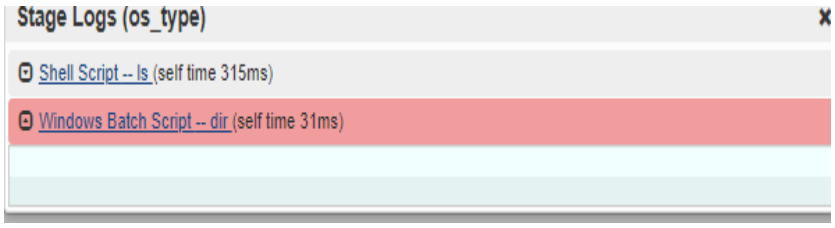

```
pipeline{
    agent any
    environment{
        pass="jobs123"
    passwod=credentials('DB_PASSWORD')
    }
    parameters{
        password(name:'entry_password')
    }
    stages{
        stage('passwd'){
            steps{
                echo "password $pass, parameter password $entry_password , credential funtion password
$passwod"
            }
        }
    }
}
```



CheckOS_AndExecuteSteps

if we execute like below it will show error, so we need to check the os type with of function

```
pipeline{
   agent any

   stages{
      stage('os_type'){
         steps{
            sh "ls"
           bat "dir"
         }
      }
   }
}
```



now it will check the ostype and then it will execute

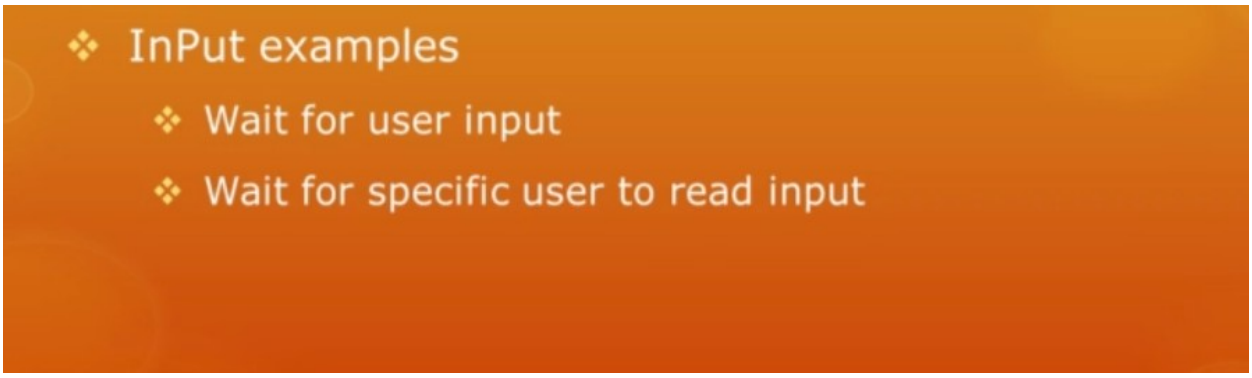
```
pipeline{
   agent any

   stages{
      stage('os_type'){
         steps{
            script{
               if(isUnix()){
                  sh "ls"
               }else{
                  bat "dir"
               }
            }
         }
      }
   }
}
```

This is linux machine, so linux command executed

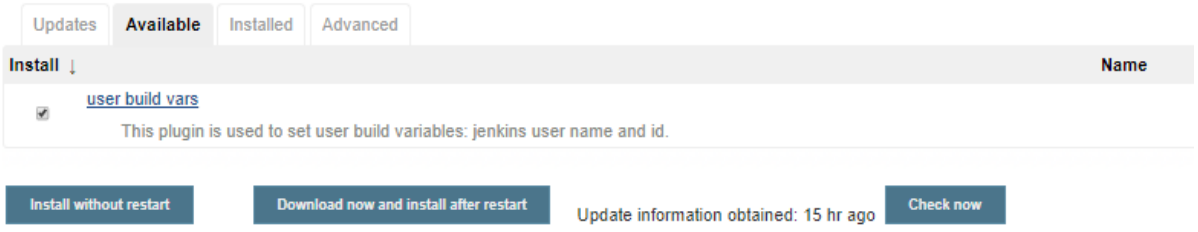
Trim

```
pipeline{
    agent any
    environment{
        tools='jenkins'
        }
    stages{
        stage('trimming_string'){
            steps{
                script{
                    t1=tools[0..6] //jenkins
                    t2=tools[3..5] //kin
                    echo "$t1 , $t2"
                }

            }

        }
    }
}
```

InPut Section



Install the plugin



```
pipeline {
  agent any

  stages {
    stage('build user') {
      steps {
        wrap([$class: 'BuildUser']) {
          sh 'echo "${BUILD_USER}"'
        }
      }
    }
  }
}
```

it will pull the user name

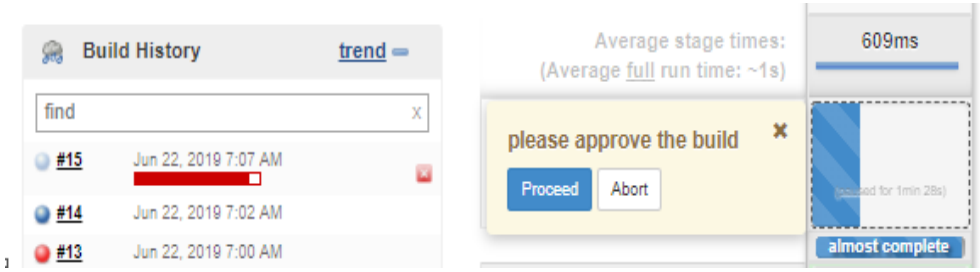
Build in specific user

```
pipeline{
                agent any
                stages{
                 stage('user_input'){
                      steps{
                              wrap([$class: 'BuildUser']){
                                      script{

                                              def name1="${BUILD_USER}"
                                              echo "${BUILD_USER}, $name1"
                                              if(name1=='kalai'){
                                                 echo "only kalai can able to build"
                                              }else{
                                                echo "others cant able to build"
                                              }
                                             }
                                      }
                              }
                }
        }
}
```

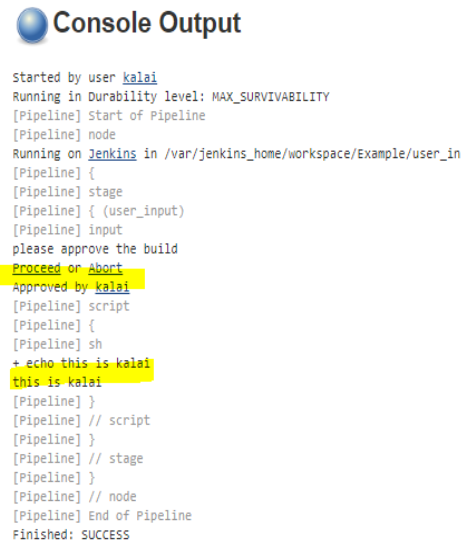
User input procced or abort

```
pipeline{
                agent any
                stages{
                 stage('user_input'){
                      steps{

                                    input("please approve the build")
                                    script{
                                      sh "echo this is kalai"
                                    }

                              }
                }
        }
}
```

user I/P procced or abort



if proceed it will start

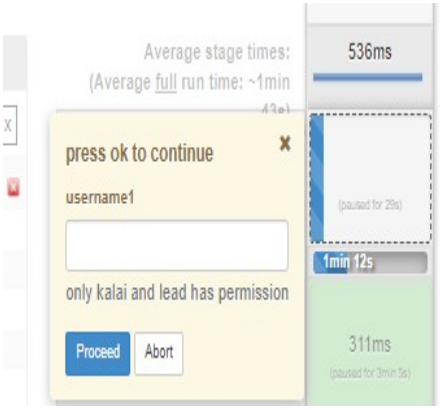


Read Input From Specific user:

we can give list of user permission to proceed, other cant give proceed and get specific string from submitter.

```
pipeline{
                agent any
                stages{
                 stage('user_input'){
                      input{
                                  message "press ok to continue"
                                  submitter "kalai,lead"
                                  parameters{
                                  string(name:'username1',description: "only kalai and lead has permission")
                                  }
                      }
                     steps{

                                  echo "User : ${username1} said ok"

                          }
                }
        }
}
```

can pass the string:



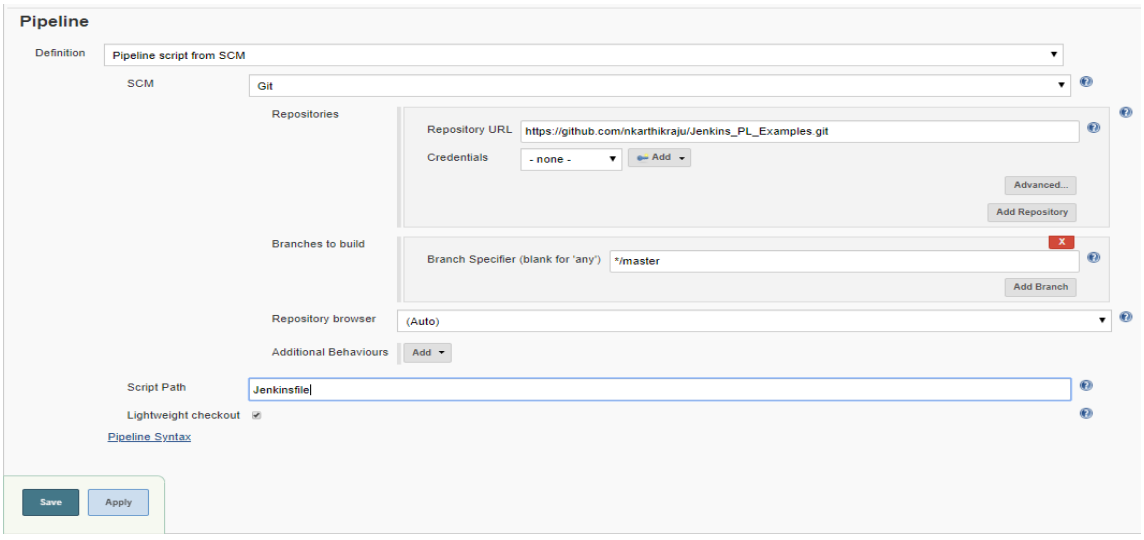
here is said manager

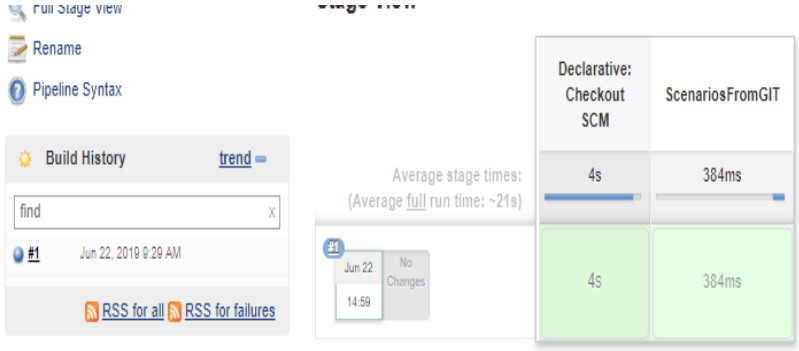
Console Output

```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Example/user_in
[Pipeline] {
[Pipeline] stage
[Pipeline] { (user_input)
[Pipeline] input
Input requested
Approved by kalai
[Pipeline] withEnv
[Pipeline] {
[Pipeline] echo
User : manager said ok
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

SCM GIT

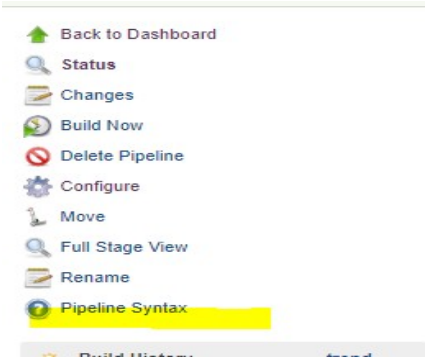
We can check out the jenkinsfile in scm whether it is git or SVN.and can maintain with version control.



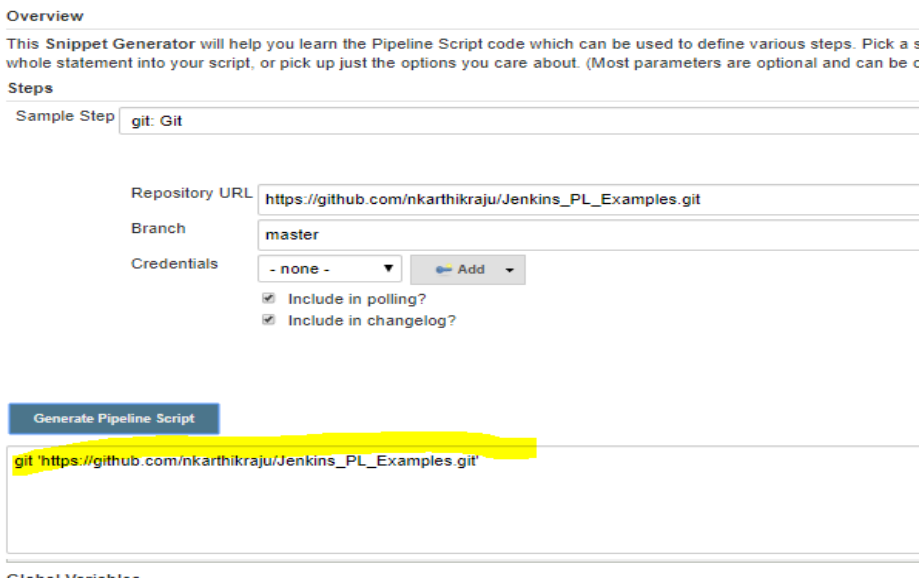


GIT checkout with the help of Pipeline syntax option

We can genereate the pipiline code from pipeline syntax option



We can generate the pipeline syntax from passing the parameter to plugin , then with generate option



Commit Code

This script will read the file and write and push to the git

```
pipeline{
        agent any
        stages{
                stage('commitcode'){
                        steps{
                                cleanWs()
                                dir('comit_from_jenkins'){
                                        git 'https://github.com/kalaiarasan33/jenkins_commit.git'
                                        script{
                                                oldv=readFile('file.txt')
                                                newv=oldv.tpInteger() + 1
                                        }
                                        writeFile file:"file.txt", text:"$newv"
                                        sh """
                                                git add file.txt
                                                git commit -m "files commited"
                                                git push

                                        """
                                }
                        }
                }
        }
}
```

Create tag for every build.

```
pipeline{
        agent any
        stages{
                stage('commitcode'){
                        steps{
                                git 'https://github.com/kalaiarasan33/jenkins_commit.git'
                                dir('tag_jenkins'){

                                        sh      "git checkout master"
                                        sh      "git tag Deployement.$BUILD_NUMBER"
                                        sh      "git push origin Deployement.$BUILD_NUMBER"

                                }
                        }
                }
        }
```

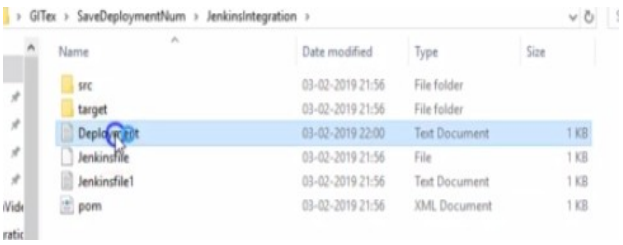
}

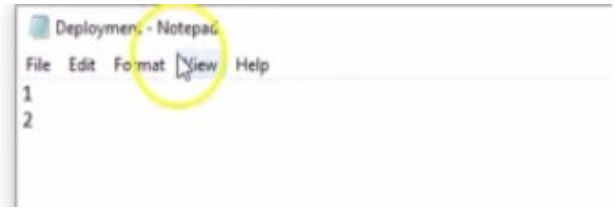
Save Build Numbers which are used for Deployment in the file

```
pipeline{
        agent any
        stages{
                stage('builnum'){
                        steps{
                                git 'https://github.com/kalaiarasan33/jenkins_commit.git'
                                sh "echo $BUILD_NUMBER >> Deployment.txt"
                                dir('save_builnum_jenkins'){

                                        sh """
                                                git add file.txt
                                                git commit -m "updating with build num commited"
                                                git push

                                        """

                                }
                        }
                }
        }
}
```

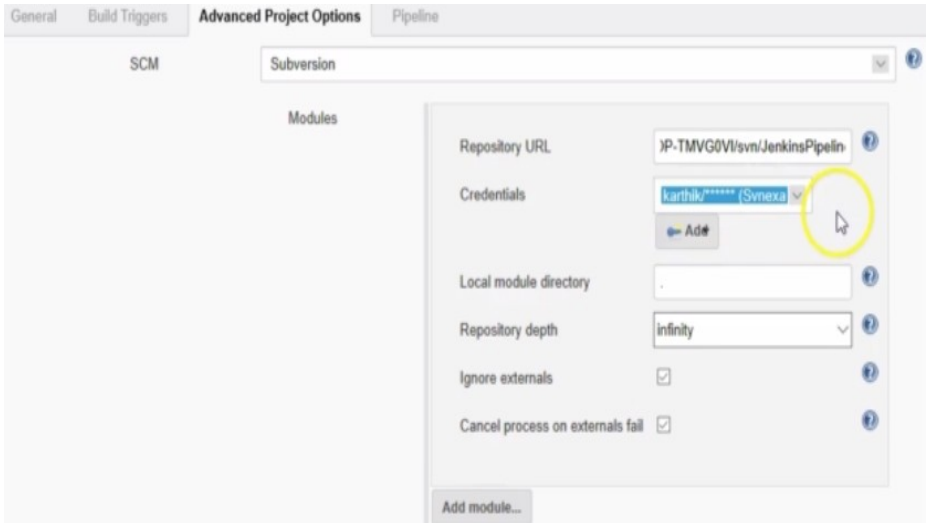
O/P

Buid no will add in the deployment file

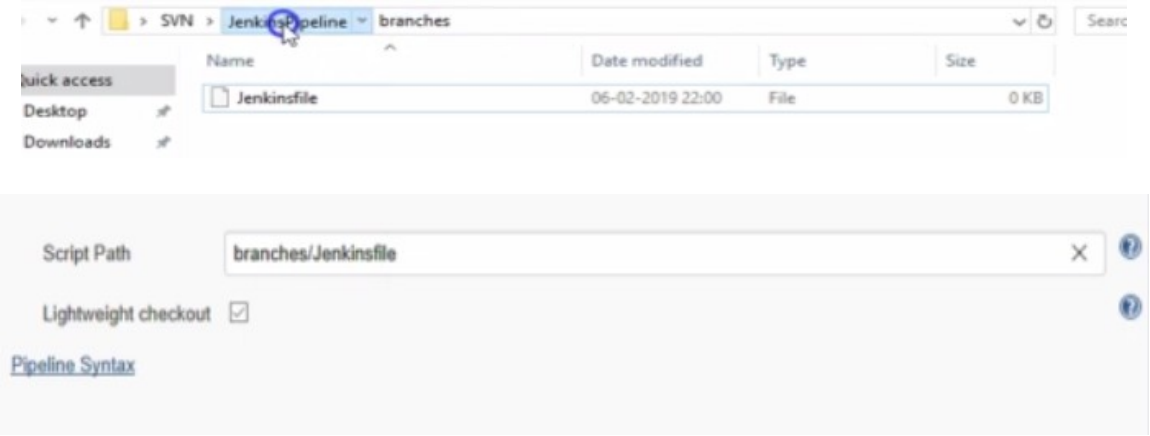




SVN examples



jenkins file in brancher folder



Stages based on When condition

```
pipeline{
        agent any
        stages{
                stage('svn'){
                when{
                        changelog 'build'
                }
                        steps{

                                                sh """
```

```
                                                    echo "found build keyword in the commit, so proceeding futher satge "
                                                """

                                            }
                                        }
                                    }
                                }
}
```

If you commit with build message, then it will build

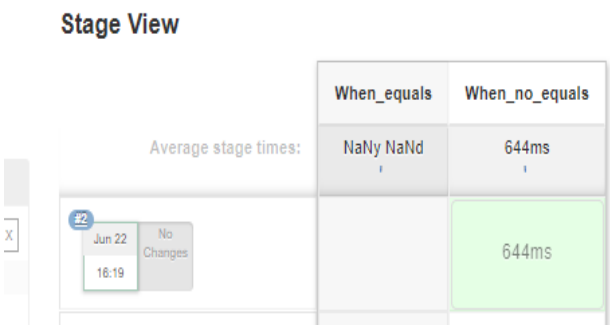
WHEN

When is similar to If condition, but its more adavanced with in-built condition

Equals and not Equals:

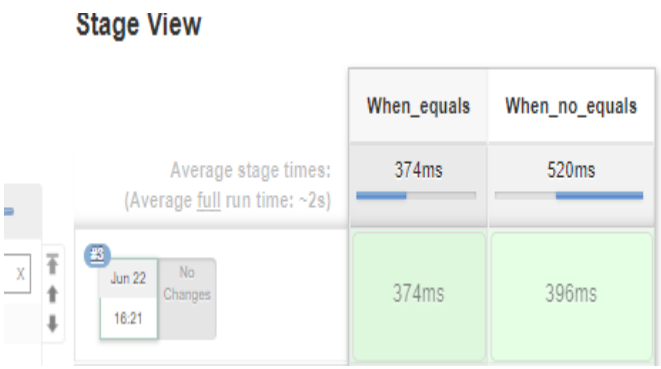
```
pipeline{
        agent any
        environment{
                Tool="Jenkins"
        }
        stages{
                stage('When_equals'){
                when{
                        equals expected:'Docker' , actual: "$Tool"
                }
                        steps{

                                        sh """
                                                echo " if when equal "
                                        """

                                }
                        }
                stage('When_no_equals'){
                        when{
                                not {
                                environment name:Tool , value:"Jenkins"
                                }

                }
                        steps{

                                        sh """
                                                echo " if when not equal "
                                        """

                                }
                        }
                }
        }
```

O/P equal expected is not matched , not equals is matched



O/P equal expected is matched , not equals is matched.

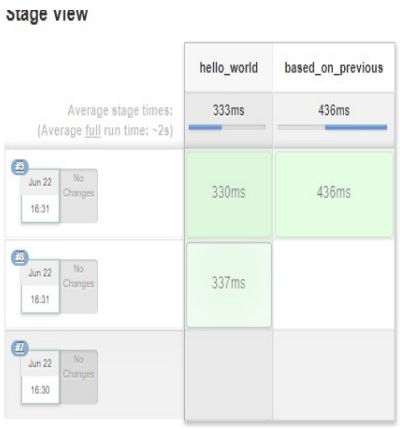
```
pipeline{
        agent any
        environment{
                Tool="Jenkins"
        }
        stages{
                stage('When_equals'){
                when{
                        equals expected:'Jenkins' , actual: "$Tool"
                }
                        steps{

                                        sh """
                                                echo " if when equal "
                                        """

                                }
                        }
                stage('When_no_equals'){
                        when{
                                not {
                                environment name:Tool , value:"Jenkins"
                                }

                        }
                        steps{

                                        sh """
                                                echo " if when not equal "
                                        """

                                }
                        }
                }
        }
```



Check previous build result and execute steps

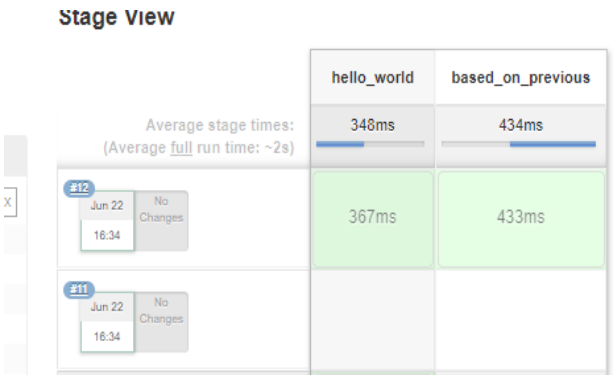
execute whe n previous build is success

```
pipeline{
        agent any
        environment{
                Tool="Jenkins"
        }
        stages{
                stage('hello_world'){
                        steps{

                                        sh """
                                                echo " hello_world "
                                        """

                                }
                        }
                stage('based_on_previous'){
                        when{
                                expression {
                                currentBuild.getPreviousBuild().result =='SUCCESS'
                                }

                }
                        steps{

                                        sh """
                                                echo " previous build is failled, now sucess"
                                        """

                                }
                        }
                }
        }
```



When previous build is failure

```
pipeline{
        agent any
        environment{
                Tool="Jenkins"
        }
        stages{
                stage('hello_world'){
                        steps{

                                        sh """
                                                echo " hello_world "
                                        """

                                }
                        }
                stage('based_on_previous'){
                        when{
                                expression {
                                currentBuild.getPreviousBuild().result =='FAILURE'
                                }

                }
                        steps{

                                        sh """
                                                echo " previous build is failled, now sucess"
                                        """

                                }
                        }
                }
        }
```



Steps based on commit messages (jenkinsfile : SCM)

when they commited with message build, it will get execute

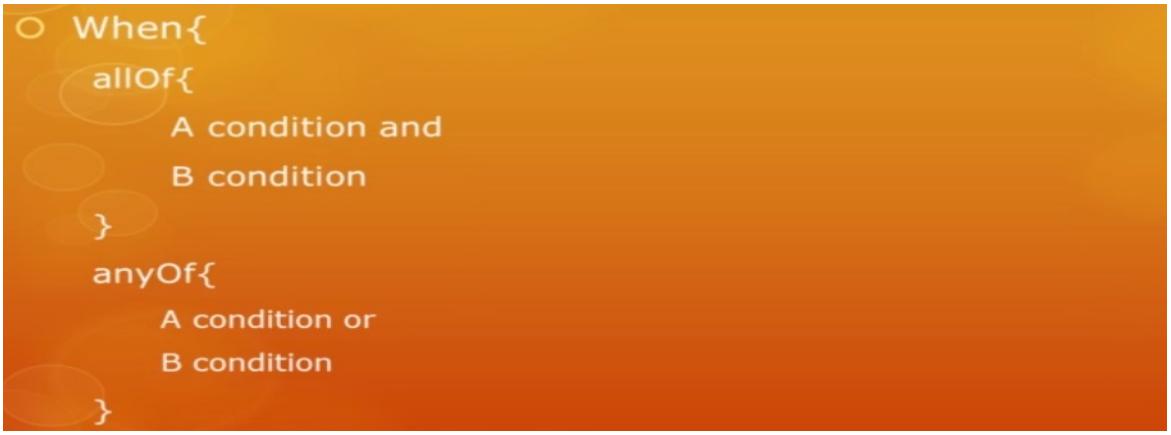
```
pipeline{
        agent any
        stages{
                stage('svn'){
                when{
                        changelog 'build'
                }
                        steps{

                                        sh """
                                                echo "found build keyword in the commit, so proceeding futher satge "
                                        """

                                }
                        }
                }
        }
}
```

Steps based on Committed files (jenkinsfile : SCM)

```
pipeline{
        agent any
        stages{
                stage('pythonfile'){
                when{
                        changeset '*.py'
                }
                        steps{

                                        sh """
                                                echo "commit files contains only for python file "
                                        """

                                }
                        }
```

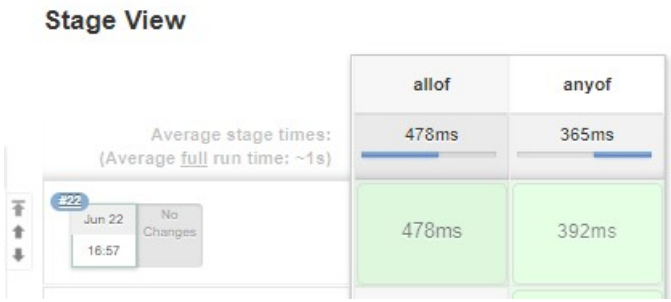
```
                stage('java_file'){
                when{
                        changeset '*.xml'
                        }
                steps{

                                sh """
                                        echo " commit files contains only for xml file"
                                """

                        }
                }
        }
    }
}
```

Allof, Anyof




```
pipeline{
        agent any
        environment{
                Tool="jenkins"
                envv="PRD"
        }
        stages{
                stage('allof'){
                when{
                 allOf{
                   equals expected: "jenkins" , actual: "$Tool"
                   equals expected: "PRD" , actual: "$envv"
                 }

                }

                        steps{

                                sh """
                                        echo " when both condition is pass "
                                """

                        }
                }
```

```
                stage('anyof'){
                when{
                anyOf{
                        equals expected: "jenkins" , actual: "$Tool"
                        equals expected: "STG" , actual: "$env"
                }
                }
                steps{

                        sh """
                                echo " when any one condition is pass"
                        """

                }
        }
    }
}
```



Execute stage if required string is matched in the file

```
pipeline{
        agent any

        stages{
                stage('string_in_file'){
                when{
                        expression{ return readFile('C:\\Users\\user\\Desktop\\cloudguru_sysops\\files.txt').contains('truncated')}
                }

                        steps{

                        sh """
                                echo " string in file "
                        """

                        }
                }
        }
}
```

Skip Stage always

if you want to skip the stage or skip for few days.

```
pipeline{
        agent any

        stages{
                stage('string_in_file'){
                when{
                    return false
                                        expression{ return readFile("C:\\Users\\user\\Desktop\\cloudguru_sysops\\files.txt").contains('truncated')}
                }

                        steps{

                                                sh """
                                                                echo " string in file "
                                                """

                        }
                }
        }
}
```

SHELL

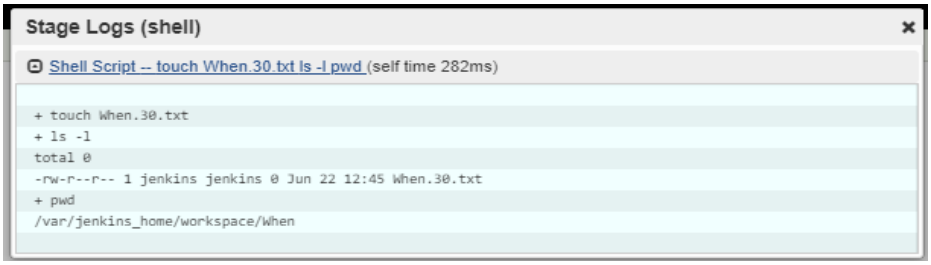
Shell Syntax and Commands

```
pipeline{
        agent any
        stages{
                stage('shell'){


                        steps{
                                                sh """           ls -l
                                                                 pwd
                                                """

                        }
                }
        }
}
```

Create file with Build Number and Build Name

```
pipeline{
```

```
    agent any
    stages{
        stage('shell'){
            steps{
                sh """
                    touch ${JOB_NAME}.${BUILD_NUMBER}.txt
                    ls -l
                    pwd
                """

            }
        }
    }
}
```



Sample Local Deployment

Create Html and Copy to the location.

```
pipeline{
    agent any

    stages{
        stage('html'){
            steps{
                bat """
                echo hello this is my HTML >> "D:\\"

                """
            }

            }
        stage('copy_location'){
            steps{
                bat """
                copy  *.html D:\\tomcat\\

                """
            }

            }
        }
    }
}
```