Jenkins Pipeline As Code

Topics

About Me

```
Lots of Engineers aspire to be full-stack, but I don't. I aspire to be a T-Shaped Engineer.
```

Hi there 👋

I am Kalaiarasan (GitHub), a DevOps Consultant from IN with 6+ years of experience focusing majorly on Continous Integration, Continous Deployment, Continous Monitoring and process development which includes code compilation, packaging , deployment/release and monitoring. 🎯

I mostly work with CI/CD Setup, Cloud Services, Developing Automation & SRE tools and AI/MLops. 🚀

Topics

- 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
-

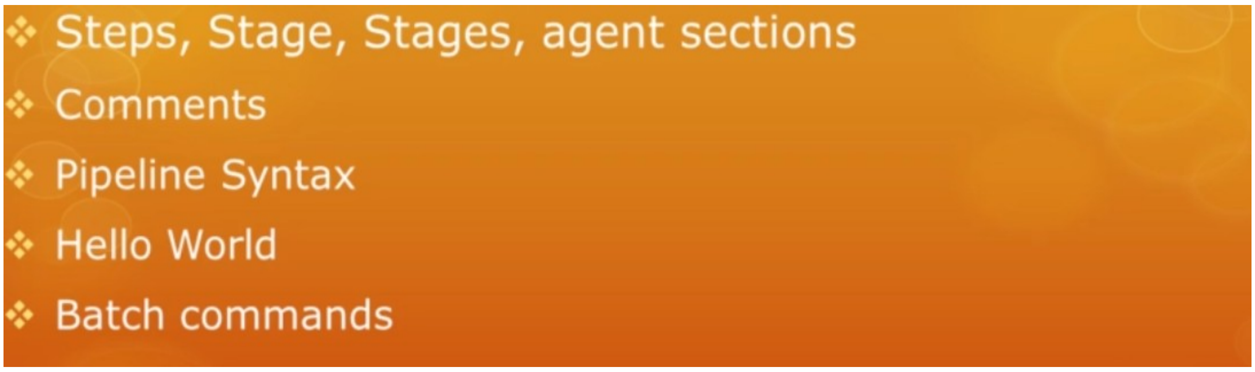
Declarative syntax

```
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, Stage, Stages, agent sections
- Comments
- Pipeline Syntax
- Hello World
- Batch commands

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

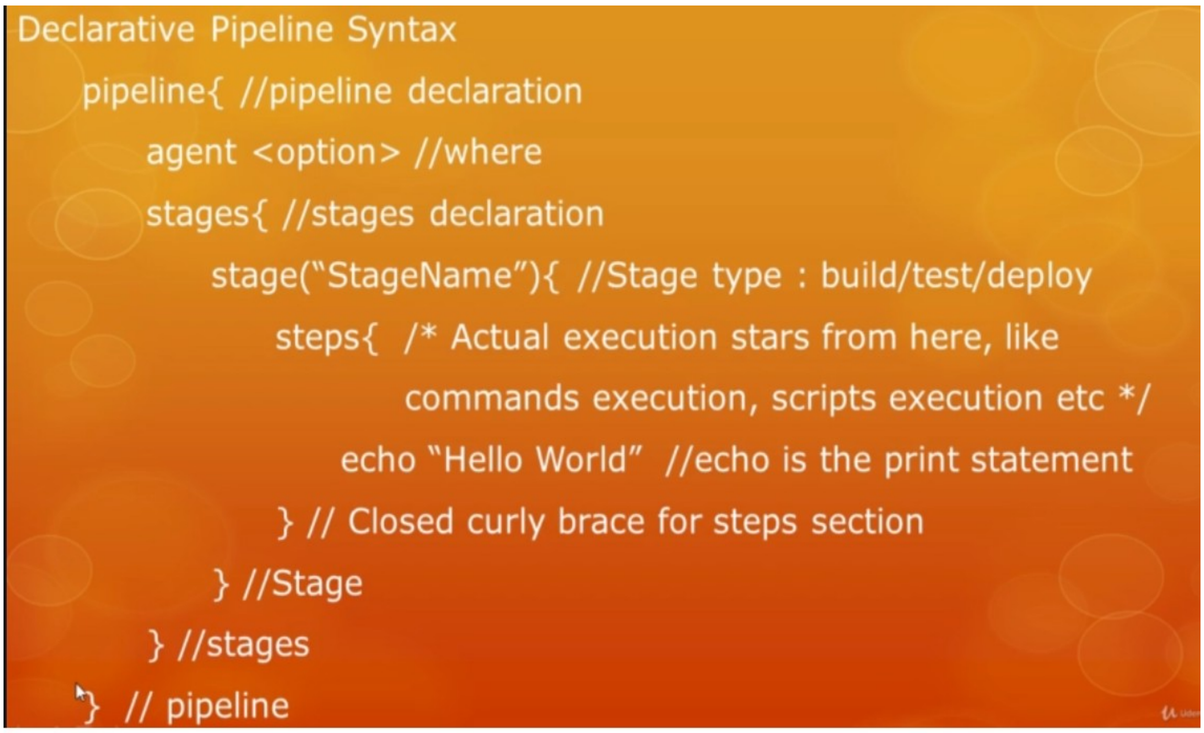
Stage colors

- 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**:** /*
```
                    ==============
                        */
```



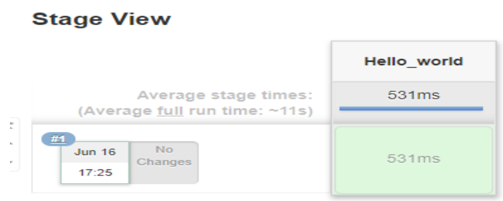
Declarative Pipeline Syntax

```
    pipeline{ //pipeline declaration
        agent <option> //where
        stages{ //stages declaration
            stage("StageName"){ //Stage type : build/test/deploy
                steps{  /* Actual execution stars from here, like
                        commands execution, scripts execution etc */
                    echo "Hello World"  //echo is the print statement
                } // Closed curly brace for steps section
            } //Stage
        } //stages
    } // pipeline
```

Simple Hello world pipeline:

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

O/P



Stage View

Hello_world

Average stage times:
(Average full run time: ~11s)

531ms

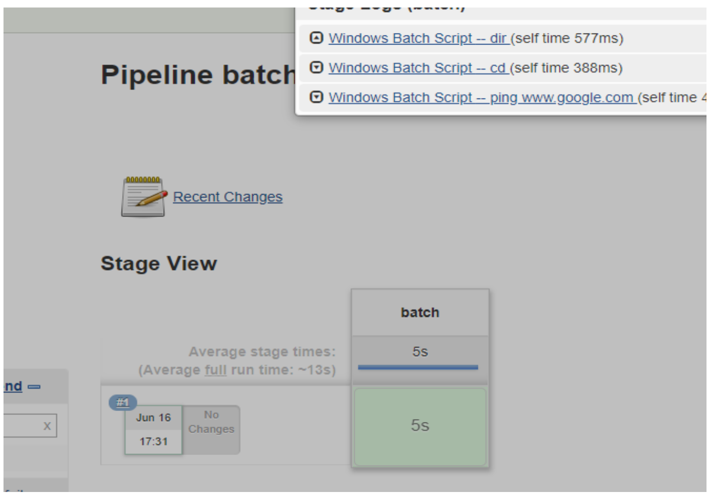
#1 Jun 16 17:25 No Changes

531ms

Batch commands

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

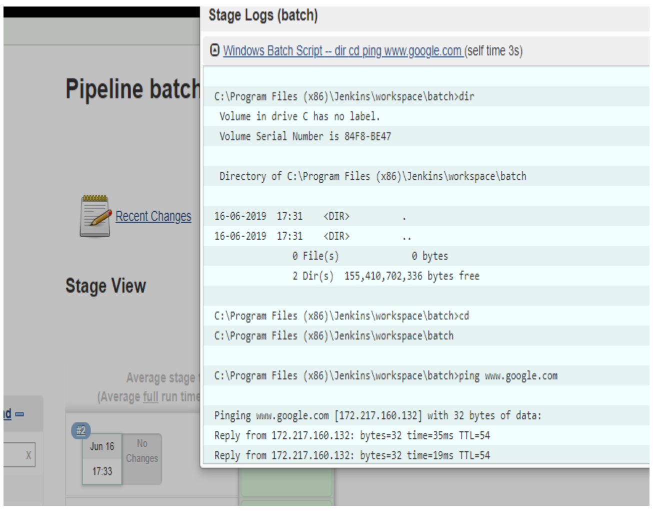
O/P


Windows Batch Script -- dir (self time 577ms)
Windows Batch Script -- cd (self time 388ms)
Windows Batch Script -- ping www.google.com (self time 4

Multiline bat command

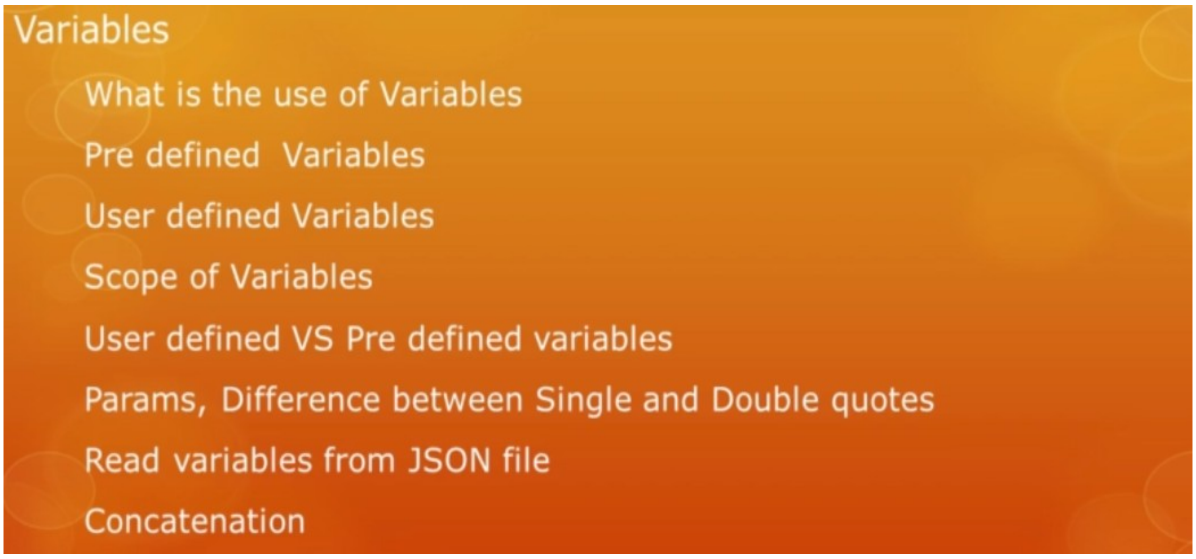
```
pipeline{
    agent any
    stages{
        stage('batch'){
            steps{
                bat """
                   dir
                   cd
                   ping www.google.com
                """

            }
        }
    }
}
```

O/P



```
C:\Program Files (x86)\Jenkins\workspace\batch>dir
 Volume in drive C has no label.
 Volume Serial Number is 84F8-BE47

 Directory of C:\Program Files (x86)\Jenkins\workspace\batch

16-06-2019  17:31    <DIR>          .
16-06-2019  17:31    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  155,410,702,336 bytes free

C:\Program Files (x86)\Jenkins\workspace\batch>cd
C:\Program Files (x86)\Jenkins\workspace\batch

C:\Program Files (x86)\Jenkins\workspace\batch>ping www.google.com

Pinging www.google.com [172.217.160.132] with 32 bytes of data:
Reply from 172.217.160.132: bytes=32 time=35ms TTL=54
Reply from 172.217.160.132: bytes=32 time=19ms TTL=54
```

VARIABLES



Variables

- What is the use of Variables
- Pre defined Variables
- User defined Variables
- Scope of Variables
- User defined VS Pre defined variables
- Params, Difference between Single and Double quotes
- Read variables from JSON file
- Concatenation

What is variable?

Variable is used to store the value.

<variable name> = <variable value>

Types

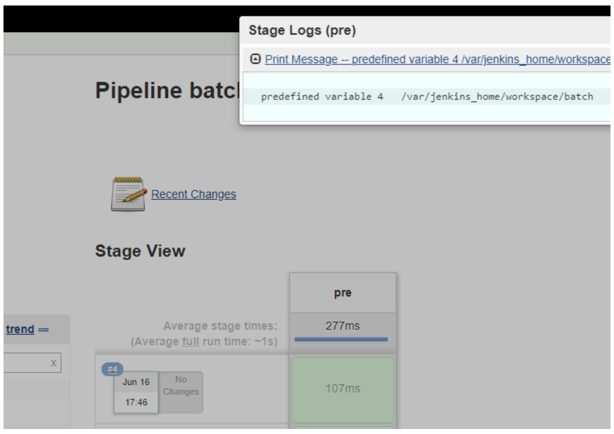
- Predefined variable
- User variable

Predefined:

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

Predefined

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



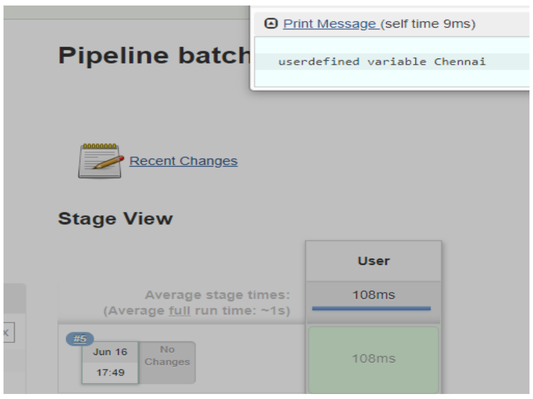
Stage Logs (pre)

Print Message -- predefined variable 4 /var/jenkins_home/workspace

```
predefined variable 4   /var/jenkins_home/workspace/batch
```

Userdefined : variable we can define in rootlevel or stage level

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



Print Message (self time 9ms)

```
userdefined variable Chennai
```

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   "

            }
        }
    }
}
```

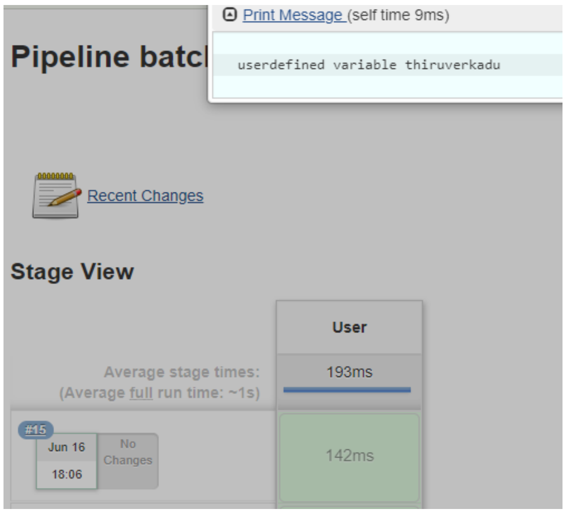
```
pipeline{
    agent any
    stages{

        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

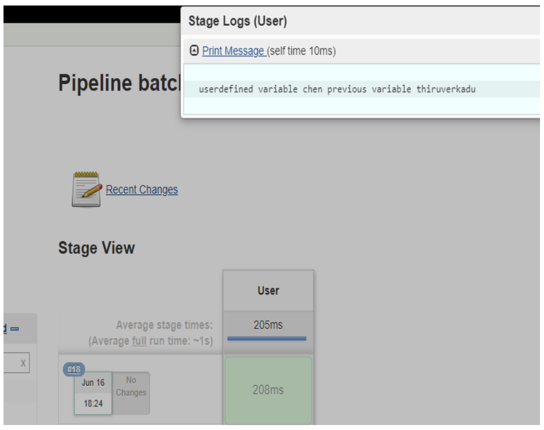


```
userdefined variable thiruverkadu
```

Predefined vs user defined values:

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

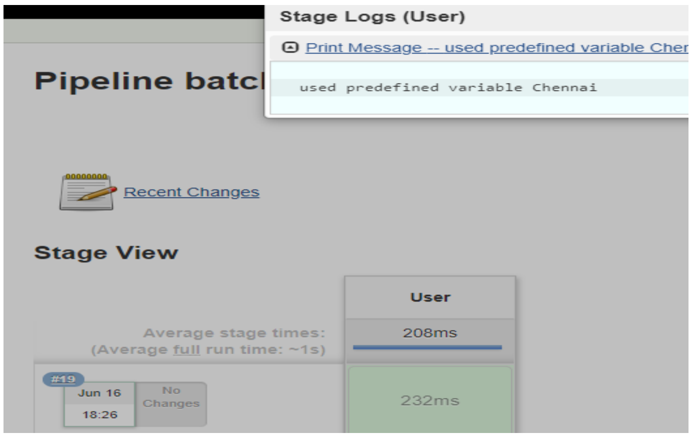
```
pipeline{
   agent any
   environment{
             MYHOME="Chennai"
            }
   stages{

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



```
userdefined variable chen previous variable thiruverkadu
```

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

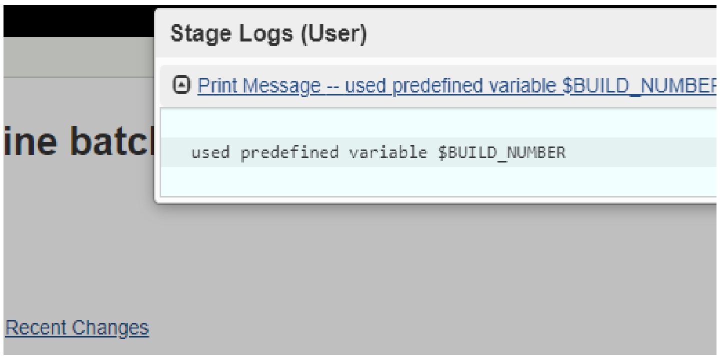
```
pipeline{
    agent any
    environment{
                BUILD_NUMBER="Chennai"
                }
    stages{

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



```
used predefined variable Chennai
```

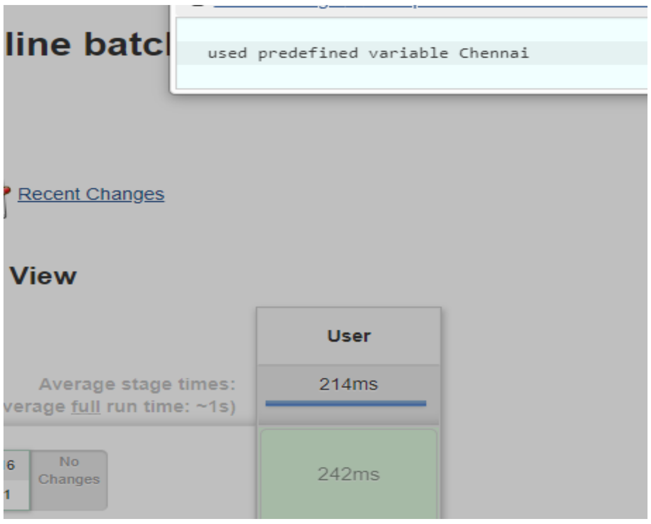
Diff B/W Single and Double quotes

if we defined in single quote it will take as string



```
used predefined variable $BUILD_NUMBER
```

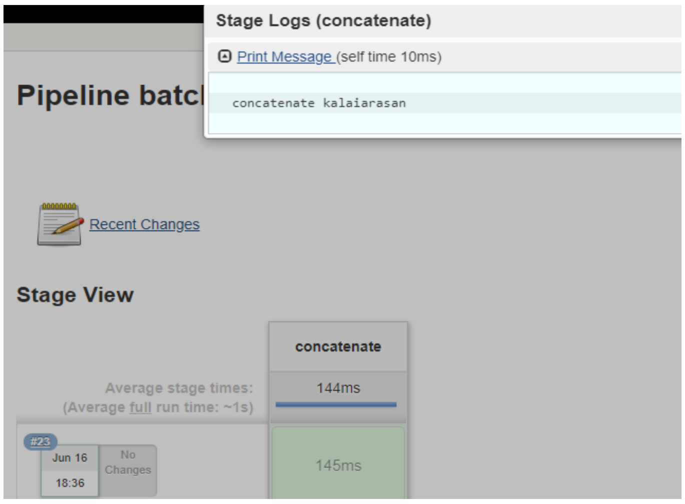
If we defined in double quotes, it will take as variable name



```
used predefined variable Chennai
```

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"
            }
        }
    }
}
```

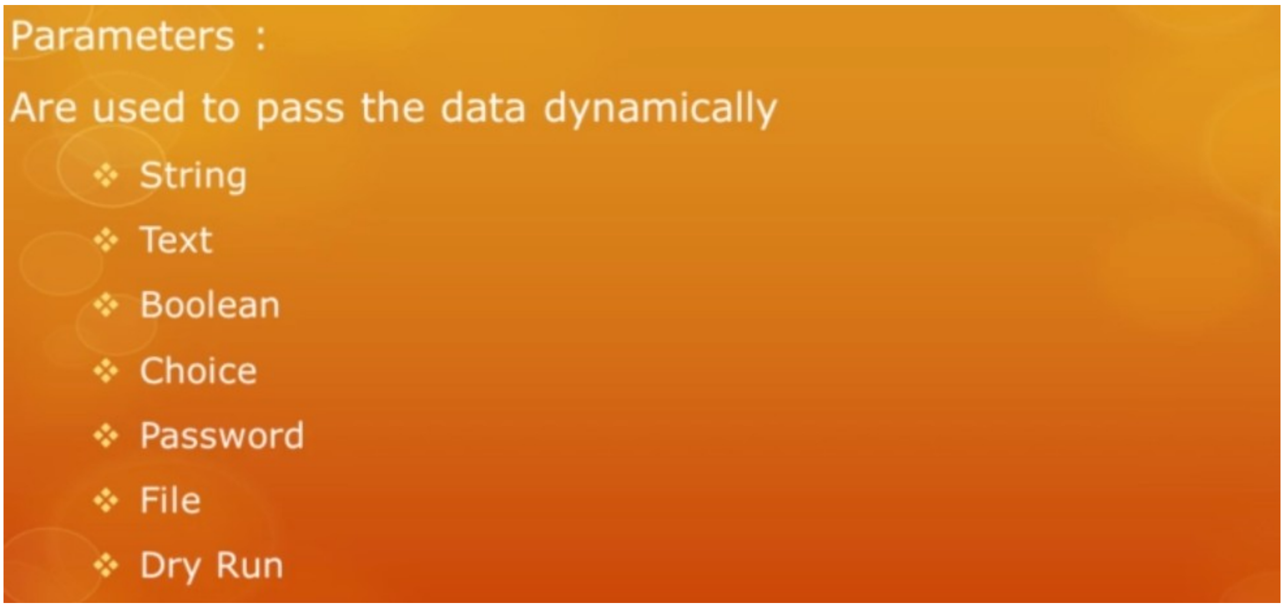


Stage Logs (concatenate)

Print Message (self time 10ms)

```
concatenate kalaiarasan
```

PARAMETERS


Parameters :

Are used to pass the data dynamically

- String
- Text
- Boolean
- Choice
- Password
- File
- Dry Run

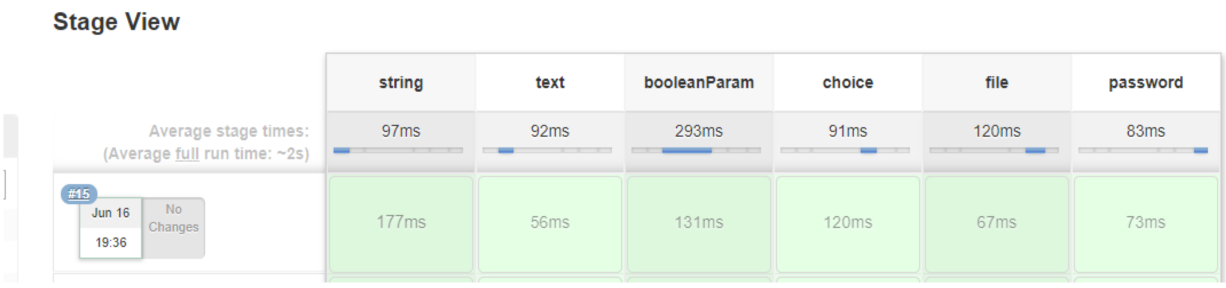
Syntax:

$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

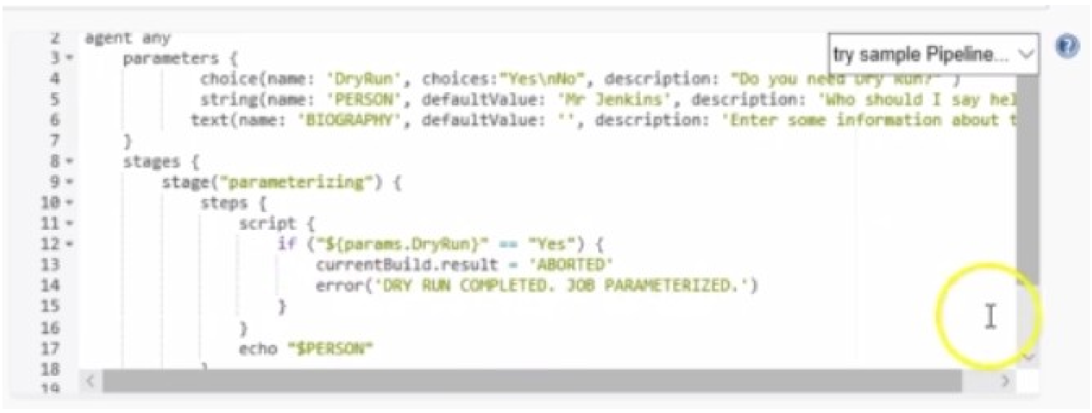


Stage View

	string	text	booleanParam	choice	file	password
Average stage times: (Average full run time: ~2s)	97ms	92ms	293ms	91ms	120ms	83ms
#15 Jun 16 19:36 No Changes	177ms	56ms	131ms	120ms	67ms	73ms

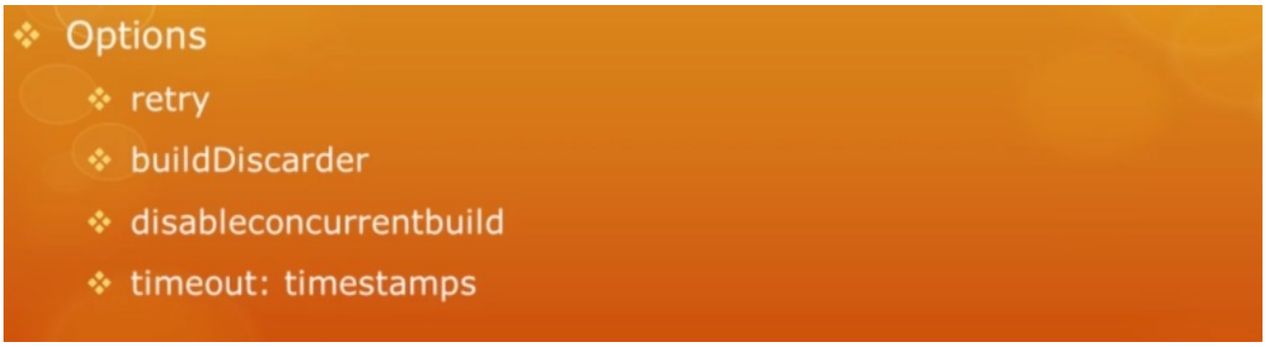
Dryrun

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



```
agent any
    parameters {
            choice(name: 'DryRun', choices:"Yes\nNo", description: "Do you need Dry Run?" )
            string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who should I say hel
           text(name: 'BIOGRAPHY', defaultValue: '', description: 'Enter some information about t
    }
    stages {
        stage("parameterizing") {
            steps {
                script {
                    if ("${params.DryRun}" == "Yes") {
                        currentBuild.result = 'ABORTED'
                        error('DRY RUN COMPLETED. JOB PARAMETERIZED.')
                    }
                }
                echo "$PERSON"
```

OPTION SET



- Options
  - retry
  - buildDiscarder
  - disableconcurrentbuild
  - timeout: timestamps

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)

         }
      }
   }
}
```

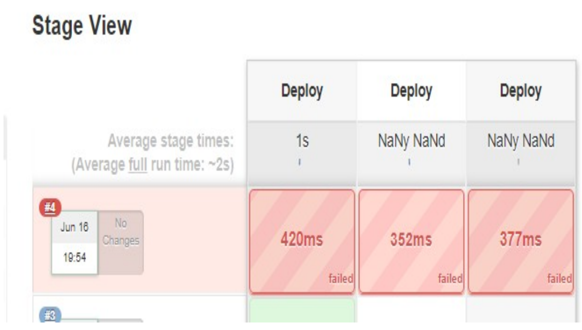
Retry: step based

```
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


Stage View

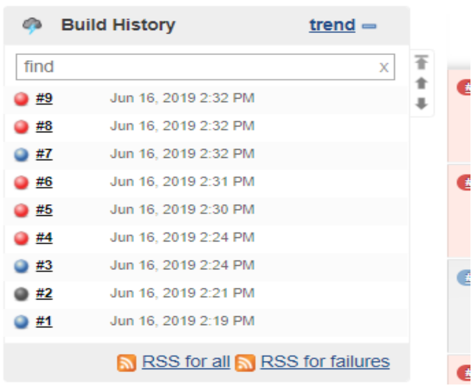
	Deploy	Deploy	Deploy
Average stage times: (Average full run time: ~2s)	1s	NaNy NaNd	NaNy NaNd
#4 Jun 16 19:54 No Changes	420ms failed	352ms failed	377ms failed
#3			

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



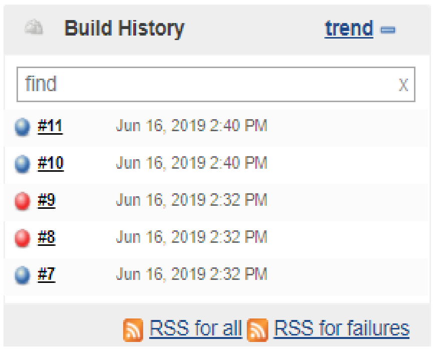
Build History trend

find

●	#9	Jun 16, 2019 2:32 PM
●	#8	Jun 16, 2019 2:32 PM
●	#7	Jun 16, 2019 2:32 PM
●	#6	Jun 16, 2019 2:31 PM
●	#5	Jun 16, 2019 2:30 PM
●	#4	Jun 16, 2019 2:24 PM
●	#3	Jun 16, 2019 2:24 PM
●	#2	Jun 16, 2019 2:21 PM
●	#1	Jun 16, 2019 2:19 PM

RSS for all RSS for failures

After : buildDiscarder execution



Build History trend

find

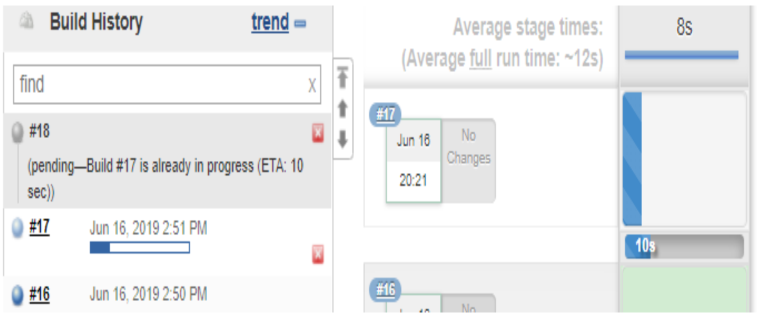
●	#11	Jun 16, 2019 2:40 PM
●	#10	Jun 16, 2019 2:40 PM
●	#9	Jun 16, 2019 2:32 PM
●	#8	Jun 16, 2019 2:32 PM
●	#7	Jun 16, 2019 2:32 PM

RSS for all RSS for failures

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)

            }
        }
    }
}
```



#18
(pending—Build #17 is already in progress (ETA: 10 sec))
#17 Jun 16, 2019 2:51 PM
#16 Jun 16, 2019 2:50 PM
#17 Jun 16 20:21
#16

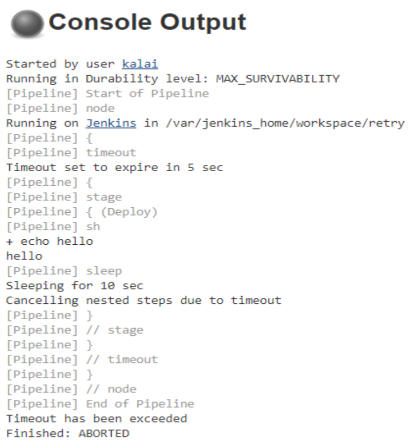
Timeout:

```
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



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/retry
[Pipeline] {
[Pipeline] timeout
Timeout set to expire in 5 sec
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] sh
+ echo hello
hello
[Pipeline] sleep
Sleeping for 10 sec
Cancelling nested steps due to timeout
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Timeout has been exceeded
Finished: ABORTED
```

Timeout Stage based:

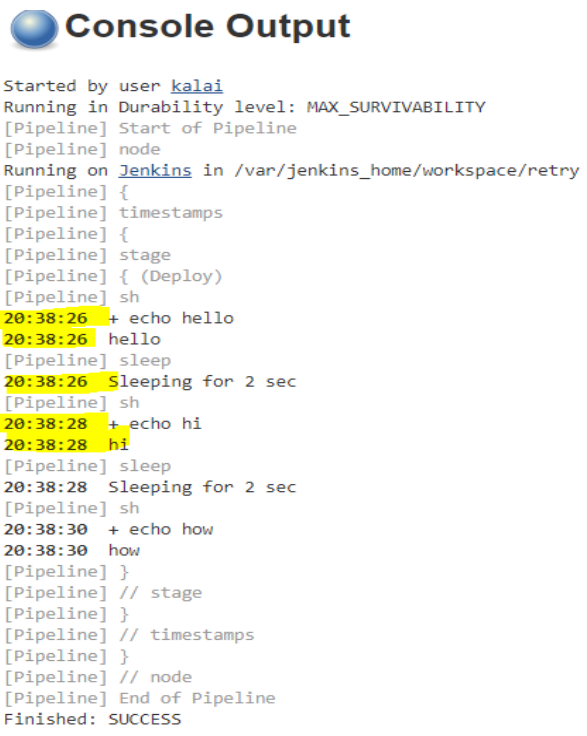
```
pipeline {
    agent any

    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'
            }
        }
    }
}
```

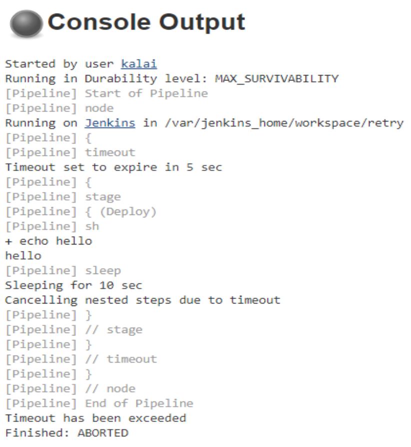
With timestamp



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/retry
[Pipeline] {
[Pipeline] timestamps
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] sh
20:38:26  + echo hello
20:38:26  hello
[Pipeline] sleep
20:38:26  Sleeping for 2 sec
[Pipeline] sh
20:38:28  + echo hi
20:38:28  hi
[Pipeline] sleep
20:38:28  Sleeping for 2 sec
[Pipeline] sh
20:38:30  + echo how
20:38:30  how
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timestamps
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

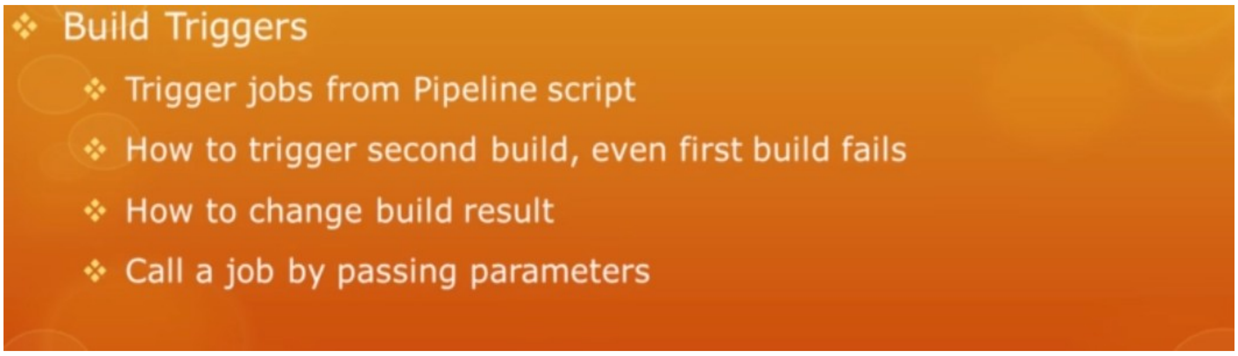
Without timestamp



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/retry
[Pipeline] {
[Pipeline] timeout
Timeout set to expire in 5 sec
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] sh
+ echo hello
hello
[Pipeline] sleep
Sleeping for 10 sec
Cancelling nested steps due to timeout
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Timeout has been exceeded
Finished: ABORTED
```

TRIGGER BUILDS



- Build Triggers
  - Trigger jobs from Pipeline script
  - How to trigger second build, even first build fails
  - How to change build result
  - Call a job by passing parameters

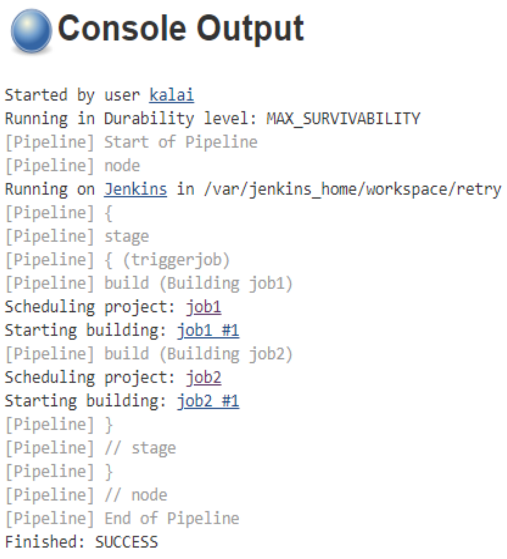
Trigger Other Jobs

- we used build('jobname') option

syntax

```
pipeline {
   agent any

   stages {
      stage('triggerjob') {
         steps {

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

O/P



Console Output

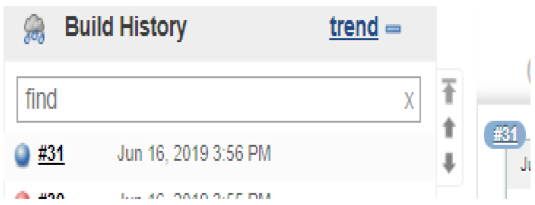
```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/retry
[Pipeline] {
[Pipeline] stage
[Pipeline] { (triggerjob)
[Pipeline] build (Building job1)
Scheduling project: job1
Starting building: job1 #1
[Pipeline] build (Building job2)
Scheduling project: job2
Starting building: job2 #1
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

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

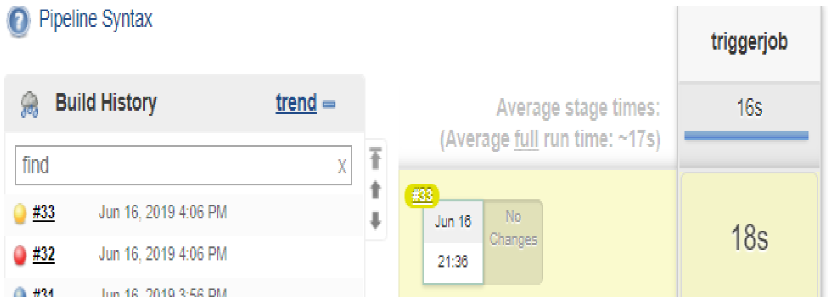
while using the below function, it will store the status in jobresult, now eventhough job failed, it will run triffer both job, but it will show unstable result status

```
 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



```
#33    Jun 16, 2019 4:06 PM
#32    Jun 16, 2019 4:06 PM
```

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

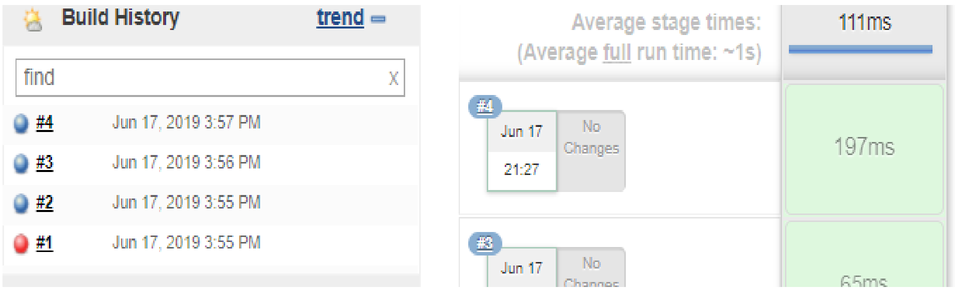
Schedule Jobs

- Cron
- Poll SCM

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



Build History trend

find

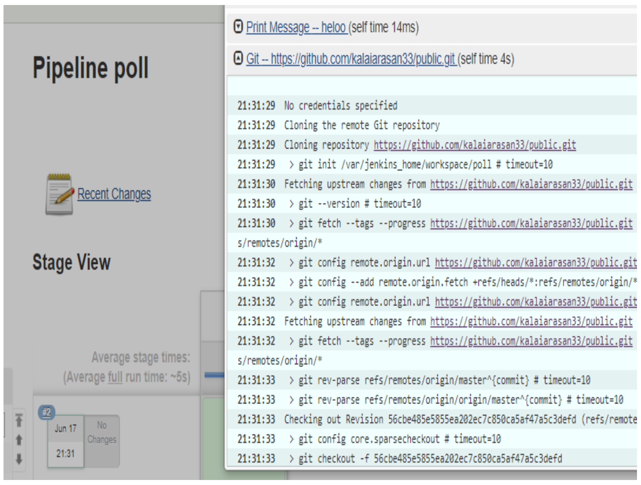
#4	Jun 17, 2019 3:57 PM
#3	Jun 17, 2019 3:56 PM
#2	Jun 17, 2019 3:55 PM
#1	Jun 17, 2019 3:55 PM

Average stage times:
(Average full run time: ~1s)

		111ms
#4 Jun 17 21:27	No Changes	197ms
#3 Jun 17	No Changes	65ms

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"
              }
         }
      }
   }
```



Pipeline poll

Recent Changes

Stage View

Average stage times:
(Average full run time: ~5s)

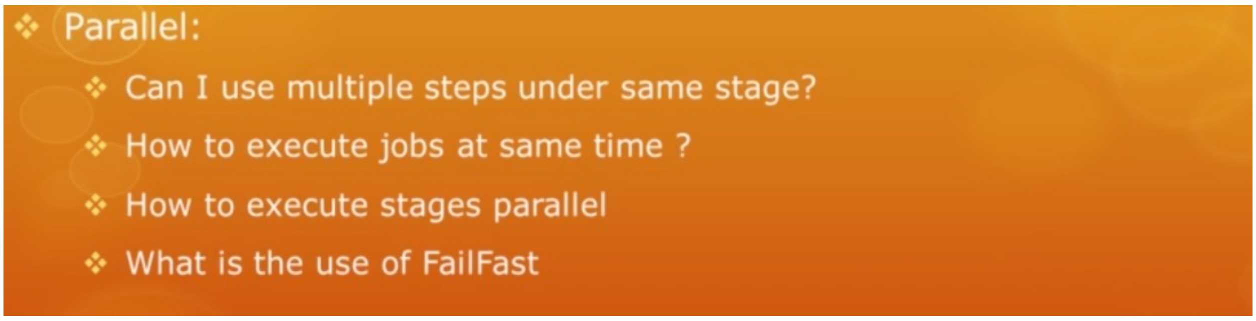
#2 Jun 17 21:31 No Changes

Print Message -- heloo (self time 14ms)

Git -- https://github.com/kalaiarasan33/public.git (self time 4s)

```
21:31:29  No credentials specified
21:31:29  Cloning the remote Git repository
21:31:29  Cloning repository https://github.com/kalaiarasan33/public.git
21:31:29   > git init /var/jenkins_home/workspace/poll # timeout=10
21:31:30  Fetching upstream changes from https://github.com/kalaiarasan33/public.git
21:31:30   > git --version # timeout=10
21:31:30   > git fetch --tags --progress https://github.com/kalaiarasan33/public.git
s/remotes/origin/*
21:31:32   > git config remote.origin.url https://github.com/kalaiarasan33/public.git
21:31:32   > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/*
21:31:32   > git config remote.origin.url https://github.com/kalaiarasan33/public.git
21:31:32  Fetching upstream changes from https://github.com/kalaiarasan33/public.git
21:31:32   > git fetch --tags --progress https://github.com/kalaiarasan33/public.git
s/remotes/origin/*
21:31:33   > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
21:31:33   > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
21:31:33  Checking out Revision 56cbe485e5855ea202ec7c850ca5af47a5c3defd (refs/remote
21:31:33   > git config core.sparsecheckout # timeout=10
21:31:33   > git checkout -f 56cbe485e5855ea202ec7c850ca5af47a5c3defd
```

Parallel



❖ Parallel:
- ❖ Can I use multiple steps under same stage?
- ❖ How to execute jobs at same time ?
- ❖ How to execute stages parallel
- ❖ What is the use of FailFast

Multiple steps sections under same stage

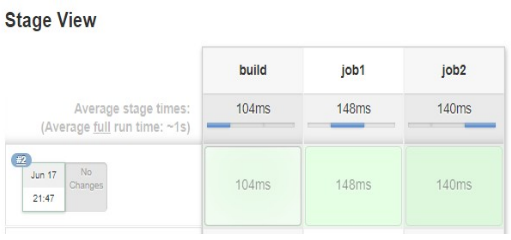
No we cant use multiple steps in same stage.like below

```
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"
                                       }
                                  }

          }
        }
    }
}
```

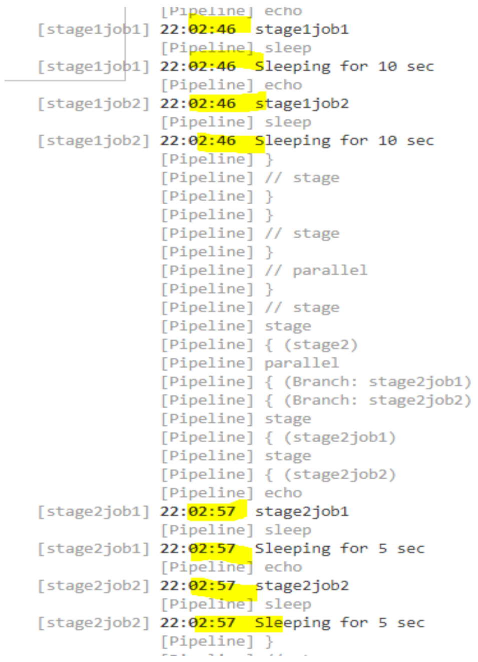
build Job is triggering parallely



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



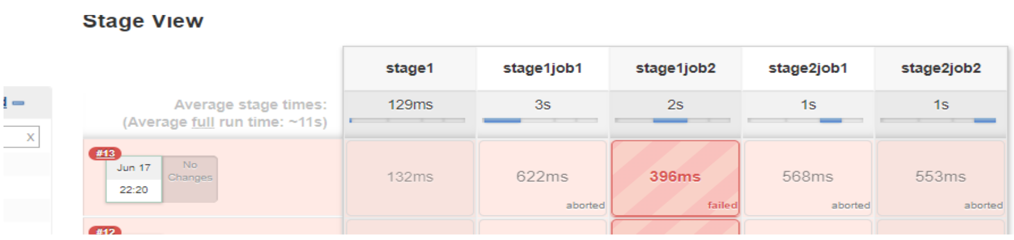
```
               [Pipeline] echo
[stage1job1] 22:02:46  stage1job1
             [Pipeline] sleep
[stage1job1] 22:02:46  Sleeping for 10 sec
             [Pipeline] echo
[stage1job2] 22:02:46  stage1job2
             [Pipeline] sleep
[stage1job2] 22:02:46  Sleeping for 10 sec
             [Pipeline] }
             [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{
```

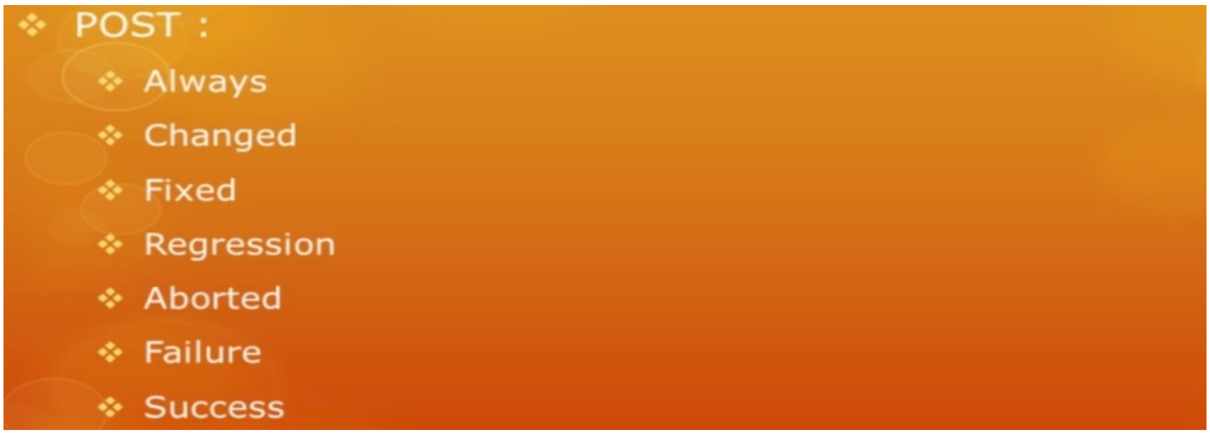
```
                                        echo "stage2job2"
                                        sleep(5)
                                               }
                                      }
                       }
                    }

                   }
                }
```

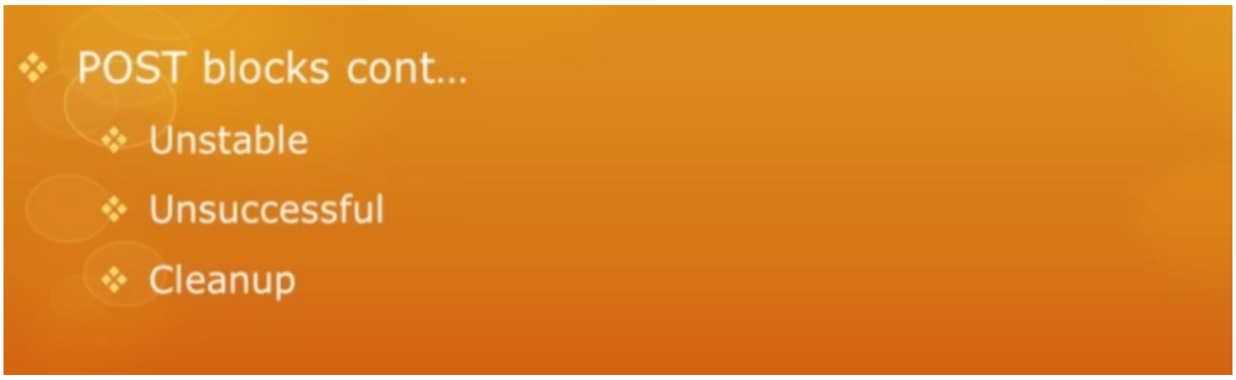

Stage View

	stage1	stage1job1	stage1job2	stage2job1	stage2job2
Average stage times: (Average full run time: ~11s)	129ms	3s	2s	1s	1s
#13 Jun 17 22:20 No Changes	132ms	622ms aborted	396ms failed	568ms aborted	553ms aborted
#12					

POST JOBS

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


- POST :
  - Always
  - Changed
  - Fixed
  - Regression
  - Aborted
  - Failure
  - Success


- POST blocks cont…
  - Unstable
  - Unsuccessful
  - Cleanup

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

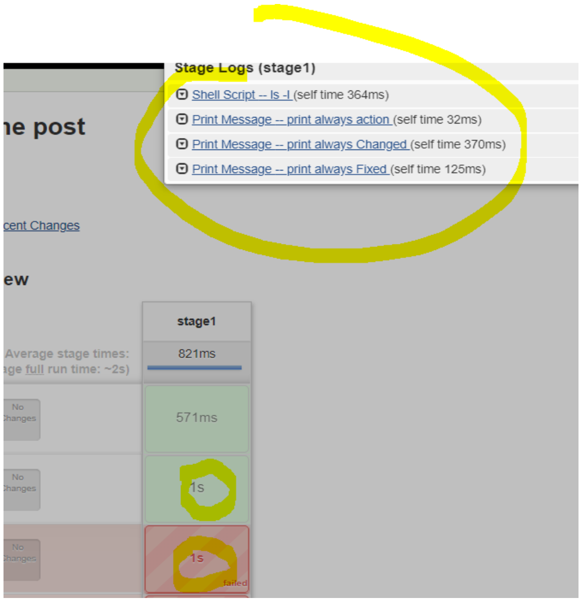
Unstable : current build is unstable

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

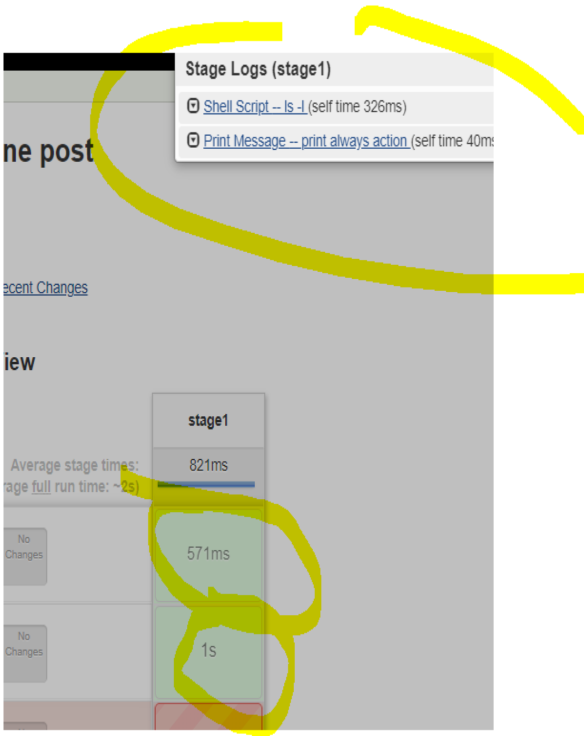


Stage Logs (stage1)

- Shell Script -- ls -l (self time 364ms)
- Print Message -- print always action (self time 32ms)
- Print Message -- print always Changed (self time 370ms)
- Print Message -- print always Fixed (self time 125ms)

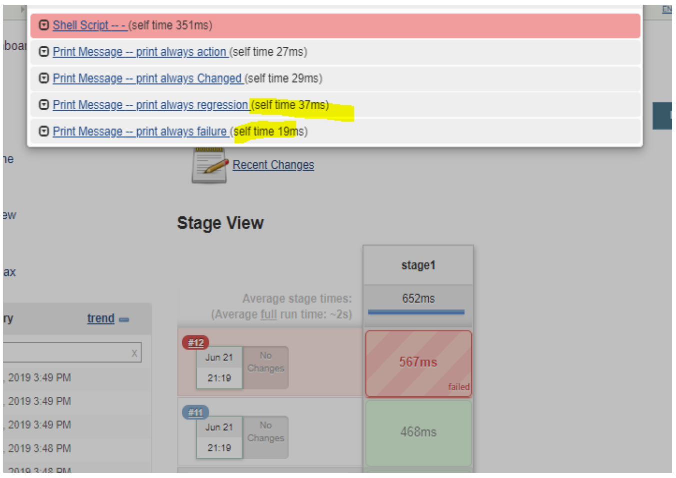
Previous build is success O/P

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

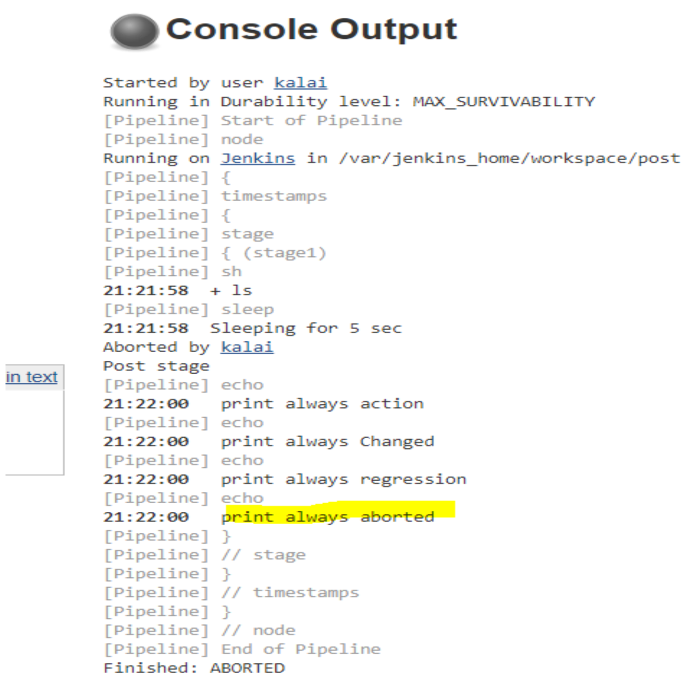


Shell Script -- - (self time 351ms)
Print Message -- print always action (self time 27ms)
Print Message -- print always Changed (self time 29ms)
Print Message -- print always regression (self time 37ms)
Print Message -- print always failure (self time 19ms)

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



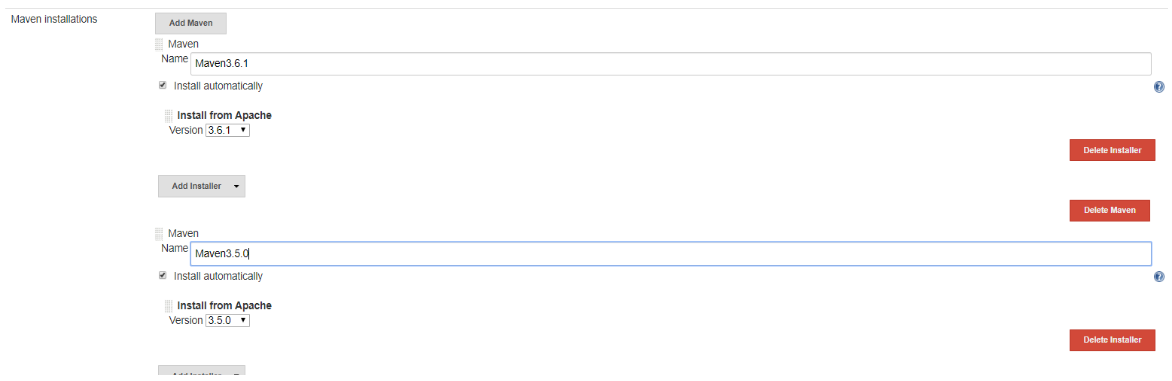
Console Output

```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/post
[Pipeline] {
[Pipeline] timestamps
[Pipeline] {
[Pipeline] stage
[Pipeline] { (stage1)
[Pipeline] sh
21:21:58  + ls
[Pipeline] sleep
21:21:58  Sleeping for 5 sec
Aborted by kalai
Post stage
[Pipeline] echo
21:22:00   print always action
[Pipeline] echo
21:22:00   print always Changed
[Pipeline] echo
21:22:00   print always regression
[Pipeline] echo
21:22:00   print always aborted
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timestamps
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: ABORTED
```

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



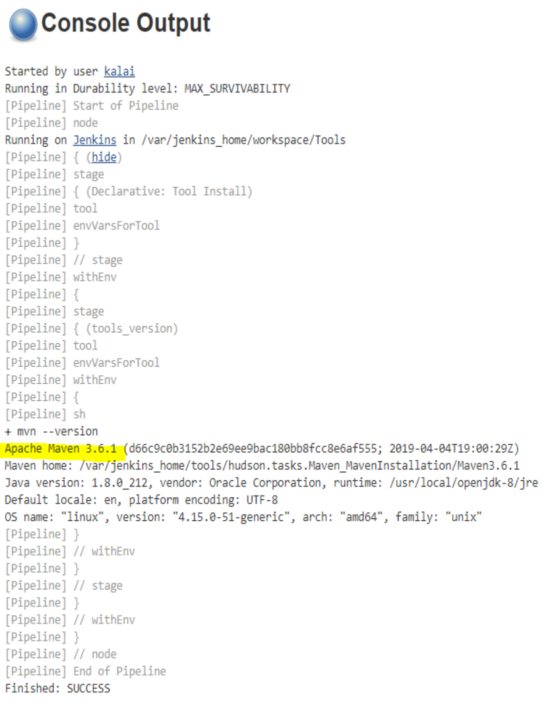
Maven installations

Maven
Name: Maven3.6.1
Install automatically [✓]
Install from Apache
Version: 3.6.1

Maven
Name: Maven3.5.0
Install automatically [✓]
Install from Apache
Version: 3.5.0

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

O/P

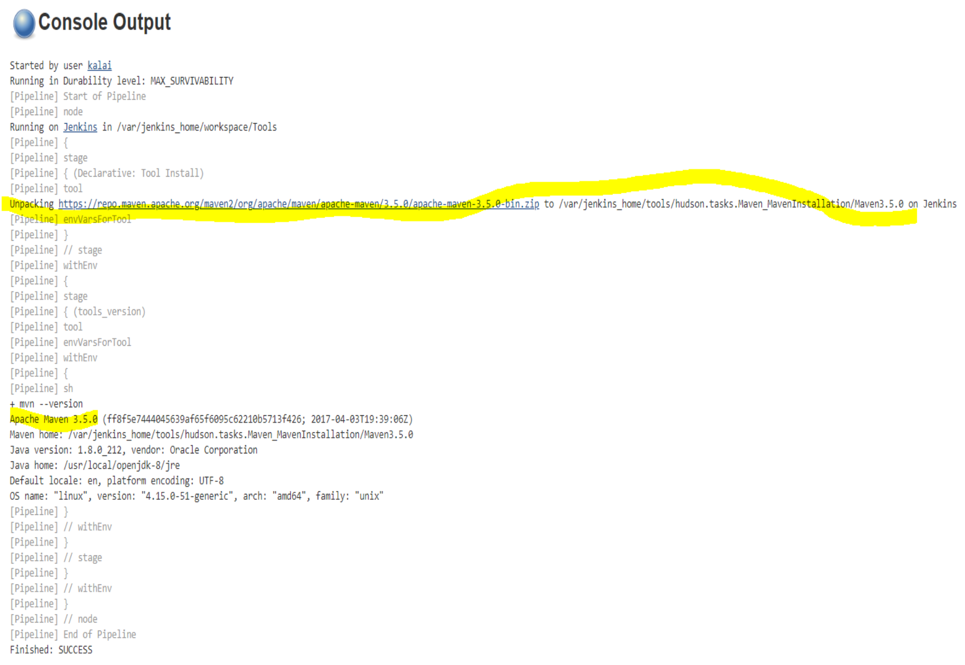


Console Output

```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Tools
[Pipeline] { (hide)
[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
[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] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Different version maven in job

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

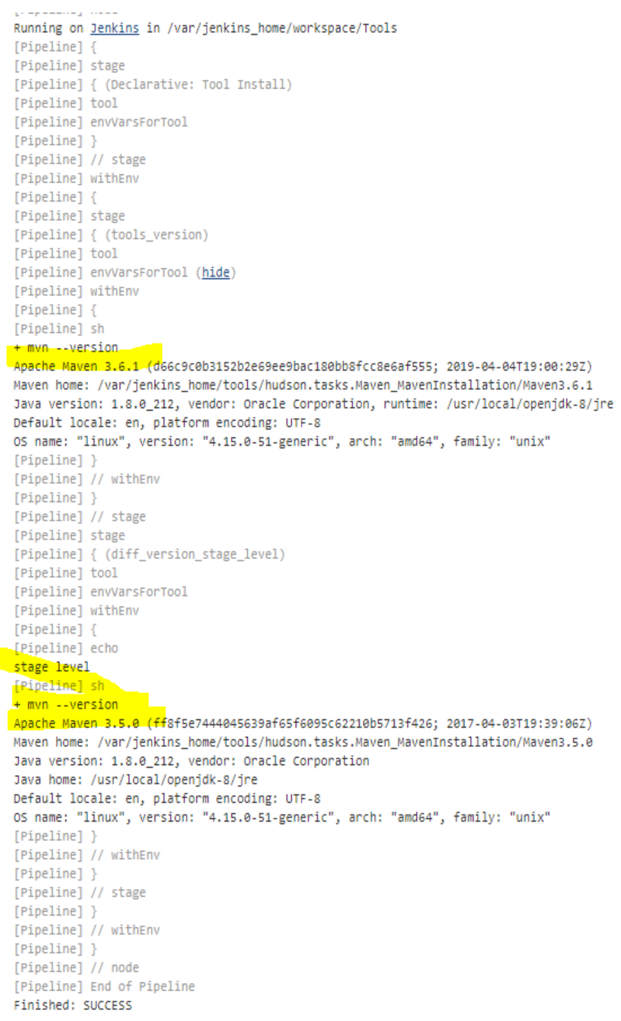


Console Output

```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Tools
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
Unpacking https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.5.0/apache-maven-3.5.0-bin.zip to /var/jenkins_home/tools/hudson.tasks.Maven_MavenInstallation/Maven3.5.0 on Jenkins
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (tools_version)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[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
```

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
```

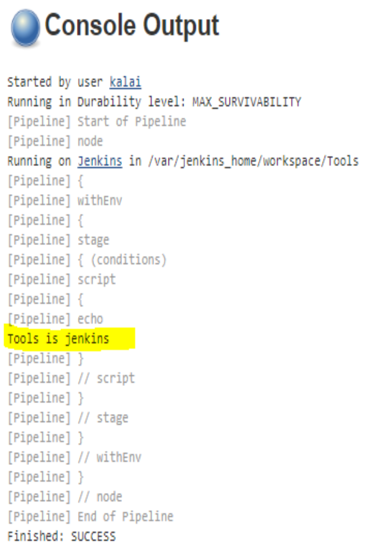
Conditional and Loop Statements

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 '
                                                       }
                                              }
         }
      }

   }
```

}

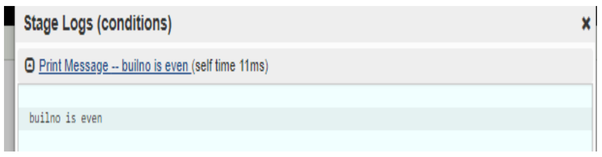


Console Output

```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Tools
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (conditions)
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Tools is jenkins
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

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'
                                                        }
                                              }
            }
        }

    }
}
```



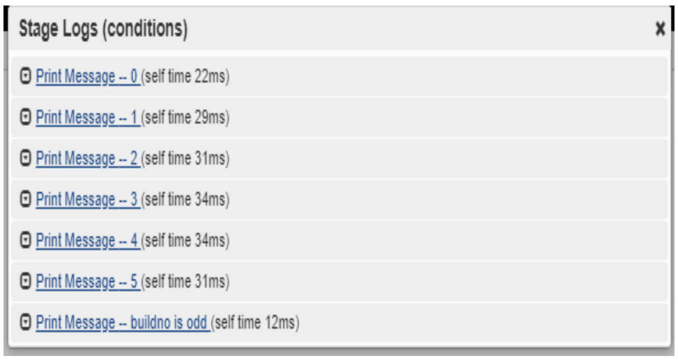
Stage Logs (conditions)

Print Message -- builno is even (self time 11ms)

```
builno is even
```

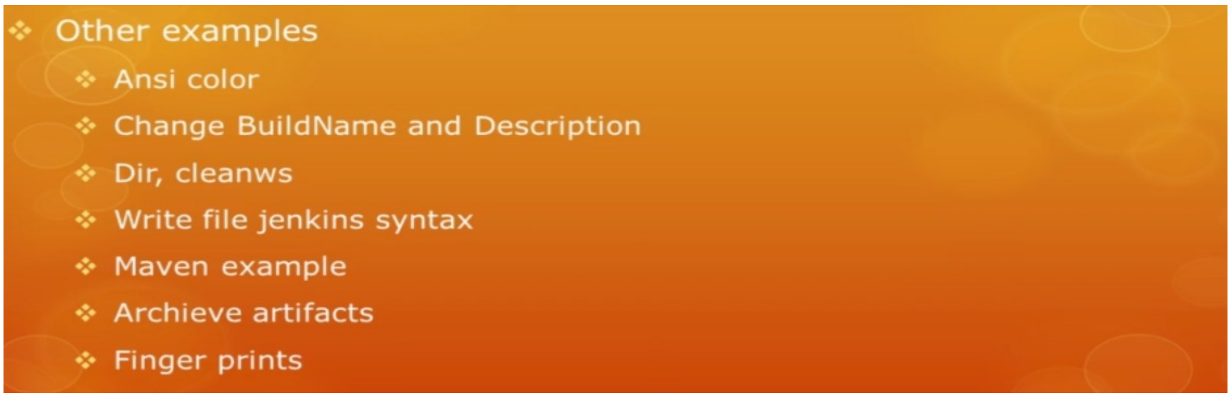
Demo: For loop

```
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'
                                                               }
                                                    }
            }
        }

    }
}
```



```
Stage Logs (conditions)
Print Message -- 0 (self time 22ms)
Print Message -- 1 (self time 29ms)
Print Message -- 2 (self time 31ms)
Print Message -- 3 (self time 34ms)
Print Message -- 4 (self time 34ms)
Print Message -- 5 (self time 31ms)
Print Message -- buildno is odd (self time 12ms)
```

Other Example



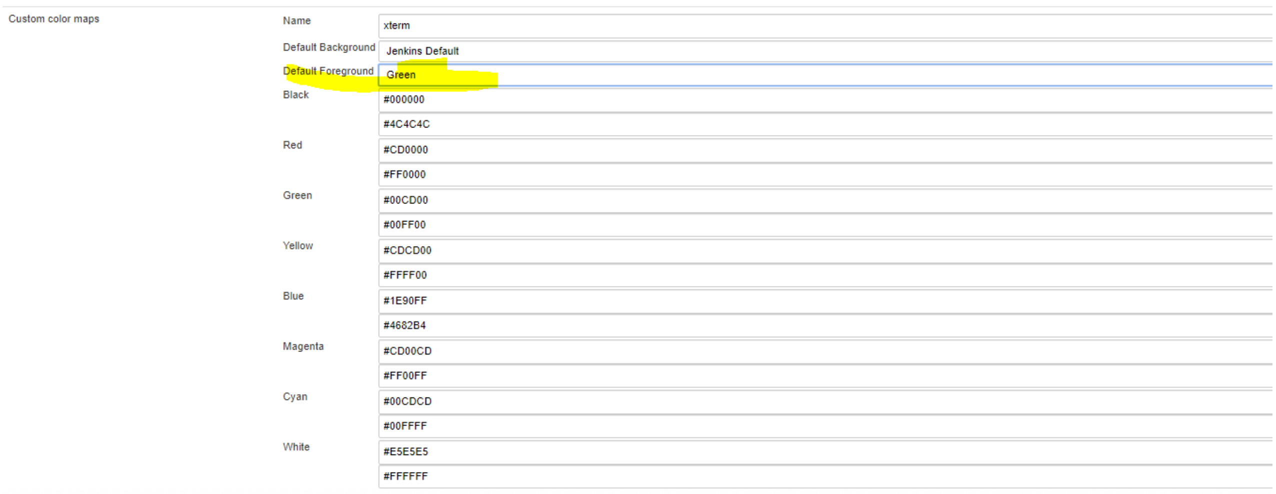
- Other examples
  - Ansi color
  - Change BuildName and Description
  - Dir, cleanws
  - Write file jenkins syntax
  - Maven example
  - Archieve artifacts
  - Finger prints

- Other examples cont…
  - Credentials
  - Check OS type
  - Trim string

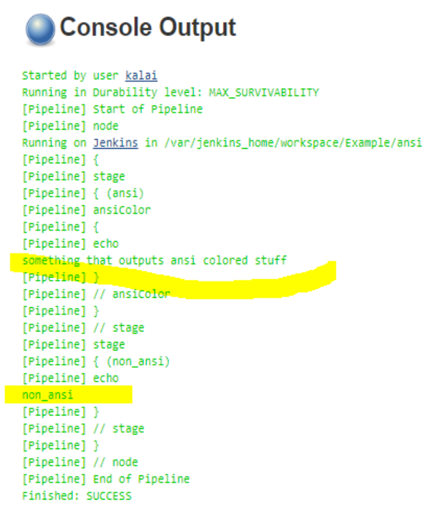
Ansicolor:

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'
            }

        }
    }
}
```


Custom color maps

Name	xterm
Default Background	Jenkins Default
Default Foreground	Green
Black	#000000
	#4C4C4C
Red	#CD0000
	#FF0000
Green	#00CD00
	#00FF00
Yellow	#CDCD00
	#FFFF00
Blue	#1E90FF
	#4682B4
Magenta	#CD00CD
	#FF00FF
Cyan	#00CDCD
	#00FFFF
White	#E5E5E5
	#FFFFFF



Console Output

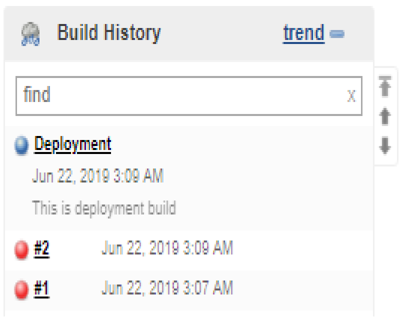
```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Example/ansi
[Pipeline] {
[Pipeline] stage
[Pipeline] { (ansi)
[Pipeline] ansiColor
[Pipeline] {
[Pipeline] echo
something that outputs ansi colored stuff
[Pipeline] }
[Pipeline] // ansiColor
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (non_ansi)
[Pipeline] echo
non_ansi
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

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



Deployment
Jun 22, 2019 3:09 AM
This is deployment build
#2 Jun 22, 2019 3:09 AM
#1 Jun 22, 2019 3:07 AM

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

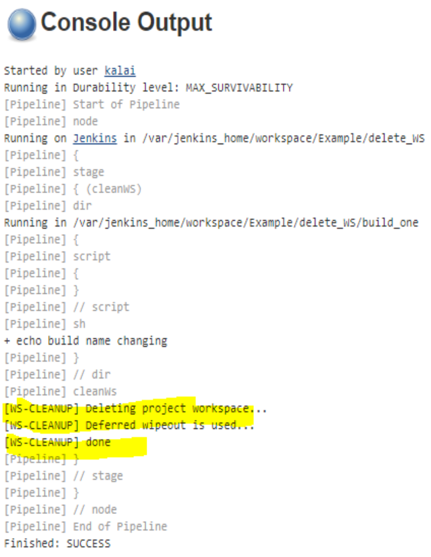
```
pipeline{
    agent any
    stages{
        stage('cleanWS'){
            steps{
                                                    dir('build_one'){
                                        script{
                                            currentBuild.displayName = "Deployment"
                                                    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()

                                }

        }
    }
```

}



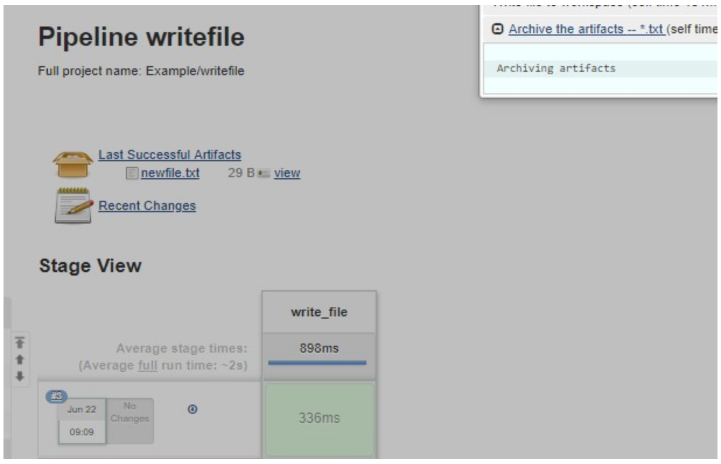
Console Output

```
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/delete_WS
[Pipeline] {
[Pipeline] stage
[Pipeline] { (cleanWS)
[Pipeline] dir
Running in /var/jenkins_home/workspace/Example/delete_WS/build_one
[Pipeline] {
[Pipeline] script
[Pipeline] {
[Pipeline] }
[Pipeline] // script
[Pipeline] sh
+ echo build name changing
[Pipeline] }
[Pipeline] // dir
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

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'


                            }

        }
    }
}
```



Archive the artifacts -- *.txt (self time

```
Archiving artifacts
```

Pipeline writefile

Full project name: Example/writefile

Last Successful Artifacts
newfile.txt 29 B view
Recent Changes

Stage View

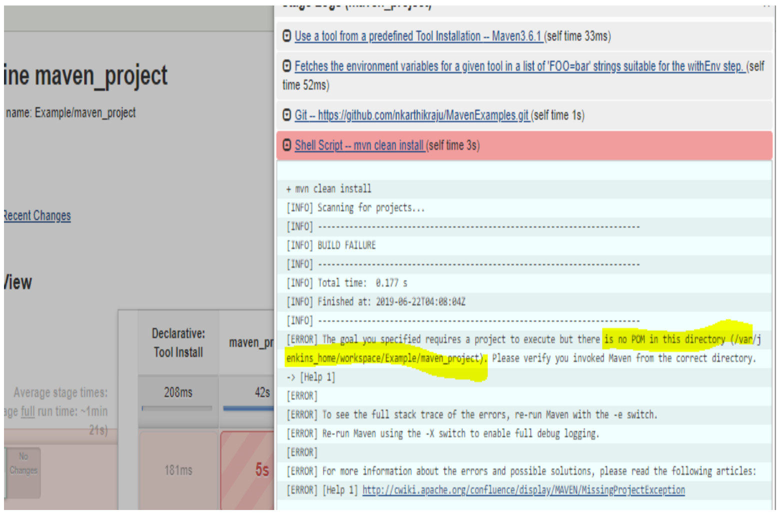
write_file
Average stage times:
(Average full run time: ~2s)
898ms
#3
Jun 22
09:09
No Changes
336ms

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



Use a tool from a predefined Tool Installation -- Maven3.6.1 (self time 33ms)

Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 52ms)

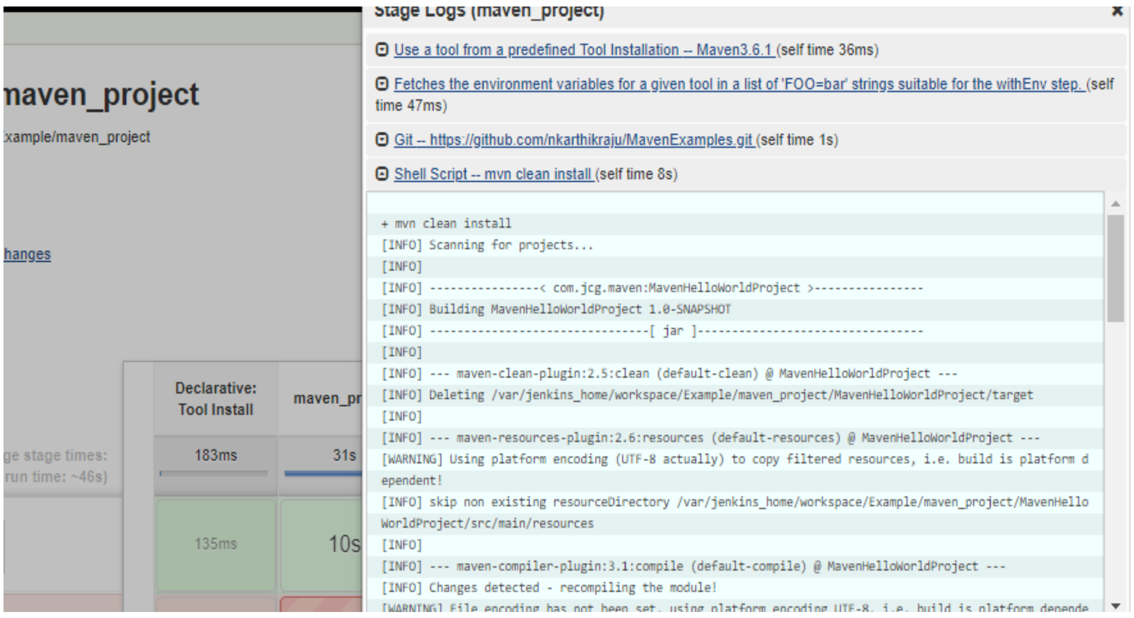
Git -- https://github.com/nkarthikraju/MavenExamples.git (self time 1s)

Shell Script -- mvn clean install (self time 3s)

```
+ mvn clean install
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.177 s
[INFO] Finished at: 2019-06-22T04:08:04Z
[INFO] ------------------------------------------------------------------------
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (/var/j
enkins_home/workspace/Example/maven_project). Please verify you invoked Maven from the correct directory.
-> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MissingProjectException
```

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"
         }
                   }

      }
   }
}
```



Stage Logs (maven_project)

Use a tool from a predefined Tool Installation -- Maven3.6.1 (self time 36ms)

Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. (self time 47ms)

Git -- https://github.com/nkarthikraju/MavenExamples.git (self time 1s)

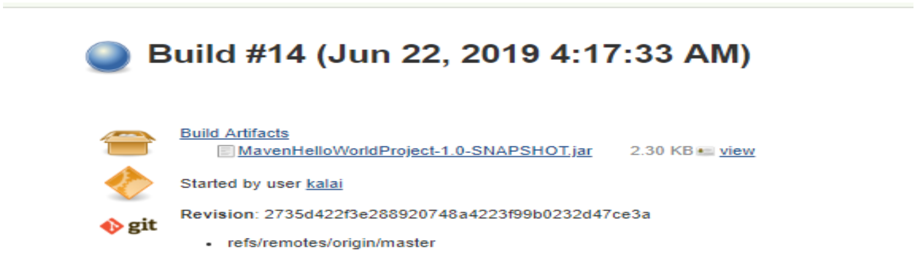
Shell Script -- mvn clean install (self time 8s)

```
+ mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------< com.jcg.maven:MavenHelloWorldProject >---------------
[INFO] Building MavenHelloWorldProject 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ MavenHelloWorldProject ---
[INFO] Deleting /var/jenkins_home/workspace/Example/maven_project/MavenHelloWorldProject/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ MavenHelloWorldProject ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform d
ependent!
[INFO] skip non existing resourceDirectory /var/jenkins_home/workspace/Example/maven_project/MavenHello
WorldProject/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ MavenHelloWorldProject ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform depende
```

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"
                        }
                    }
      }
    }
}
```



Build #14 (Jun 22, 2019 4:17:33 AM)

Build Artifacts
MavenHelloWorldProject-1.0-SNAPSHOT.jar 2.30 KB view

Started by user kalai

Revision: 2735d422f3e288920748a4223f99b0232d47ce3a

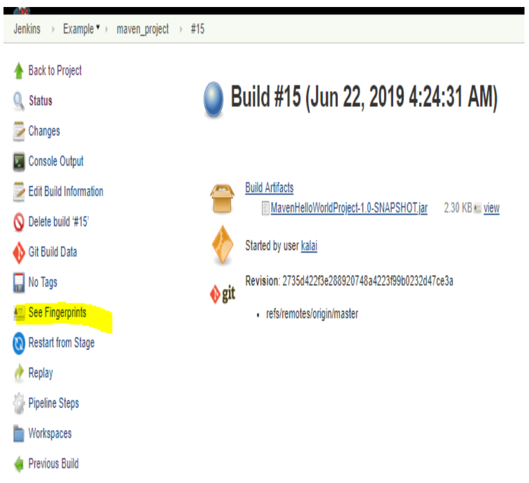
- refs/remotes/origin/master

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



Jenkins › Example › maven_project › #15

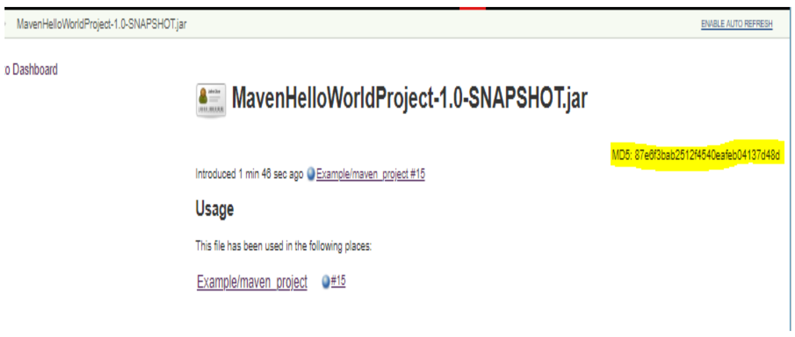
- Back to Project
- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#15'
- Git Build Data
- No Tags
- See Fingerprints
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces
- Previous Build

Build #15 (Jun 22, 2019 4:24:31 AM)

Build Artifacts
MavenHelloWorldProject-1.0-SNAPSHOT.jar 2.30 KB view

Started by user kalai

Revision: 2735d422f3e288920748a4223f99b0232d47ce3a
- refs/remotes/origin/master



MavenHelloWorldProject-1.0-SNAPSHOT.jar ENABLE AUTO REFRESH

o Dashboard

MavenHelloWorldProject-1.0-SNAPSHOT.jar

MD5: 87e6f3bab2512f4540eafeb04137d48d

Introduced 1 min 46 sec ago Example/maven_project #15

Usage

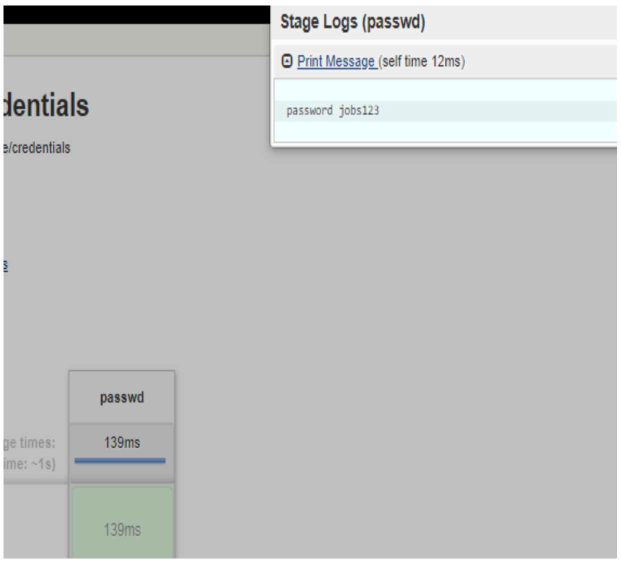
This file has been used in the following places:

Example/maven_project #15

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"
            }
        }
    }
}
```

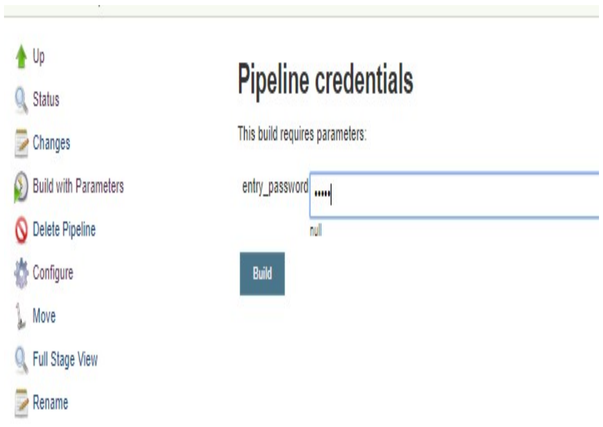


```
password jobs123
```

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

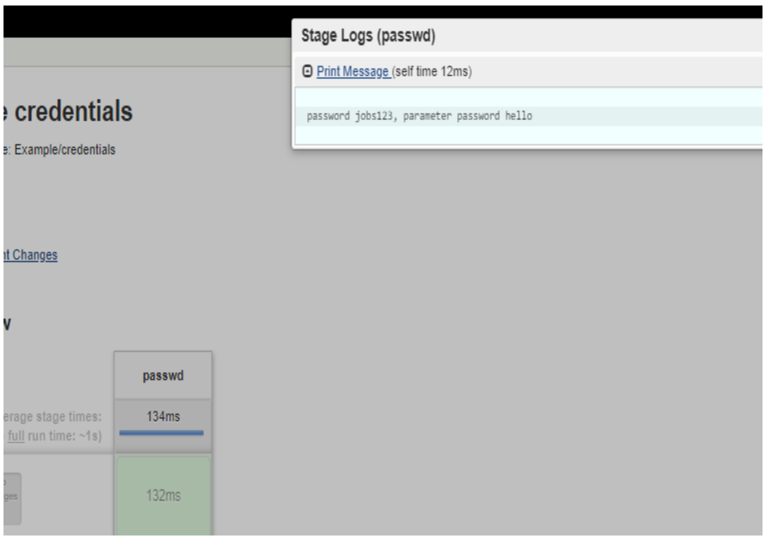


Pipeline credentials

This build requires parameters:

entry_password •••••

null



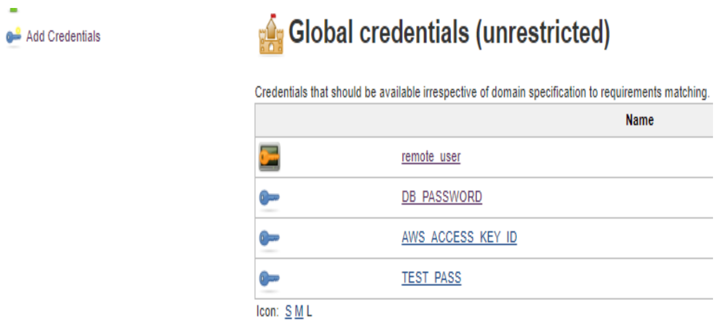
Stage Logs (passwd)

Print Message (self time 12ms)

```
password jobs123, parameter password hello
```

Now password with credential function

create sceret text in credentials in jenkins



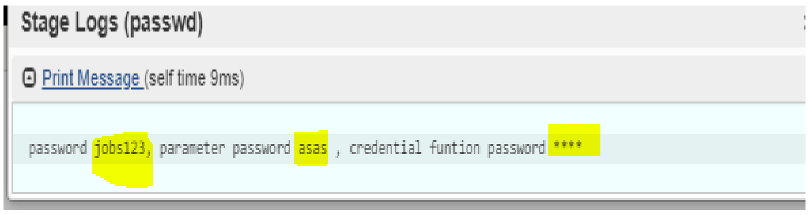
Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

	Name
	remote_user
	DB_PASSWORD
	AWS_ACCESS_KEY_ID
	TEST_PASS

Icon: S M L

```
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"
            }
        }
    }
}
```



Stage Logs (passwd)

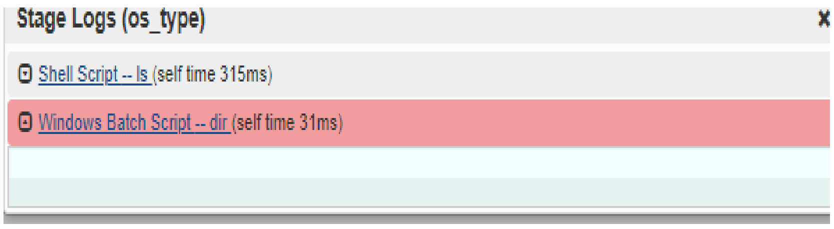
Print Message (self time 9ms)

```
password jobs123, parameter password asas , credential funtion password ****
```

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"
            }
        }
    }
}
```



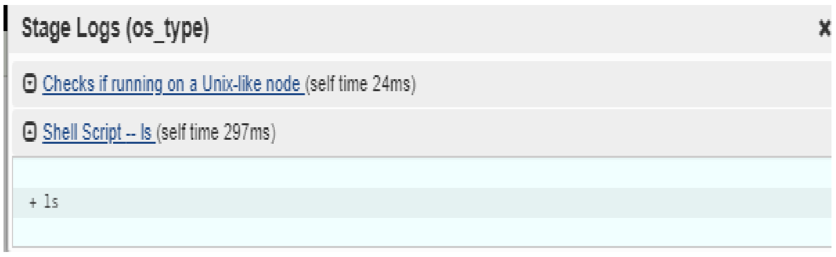
Stage Logs (os_type)

Shell Script -- ls (self time 315ms)

Windows Batch Script -- dir (self time 31ms) 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


Stage Logs (os_type)

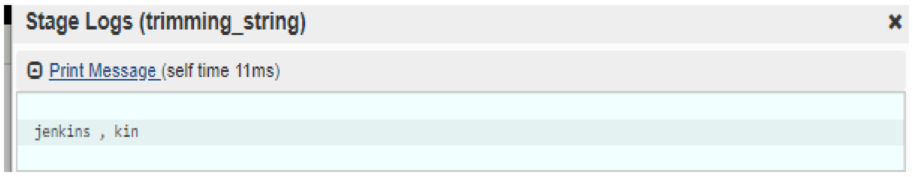
Checks if running on a Unix-like node (self time 24ms)

Shell Script -- ls (self time 297ms)

```
+ ls
```

Trim

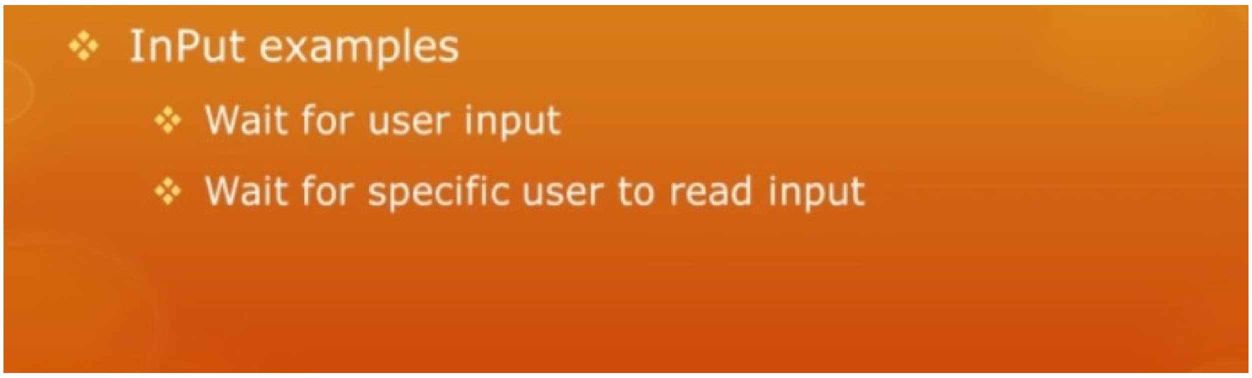
```
pipeline{
    agent any
    environment{
        tools='jenkins'
        }
    stages{
        stage('trimming_string'){
            steps{
                script{
                     t1=tools[0..6] //jenkins
                     t2=tools[3..5] //kin
                     echo "$t1 , $t2"
      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

            }

        }
    }
}
```


Stage Logs (trimming_string)

Print Message (self time 11ms)

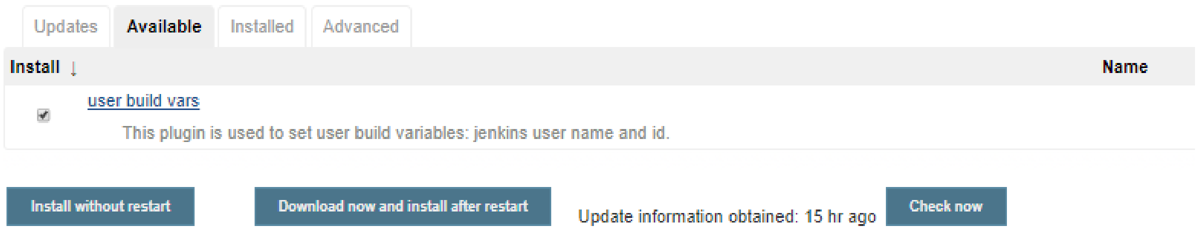
```
jenkins , kin
```

InPut Section



- InPut examples
  - Wait for user input
  - Wait for specific user to read input

Install the plugin



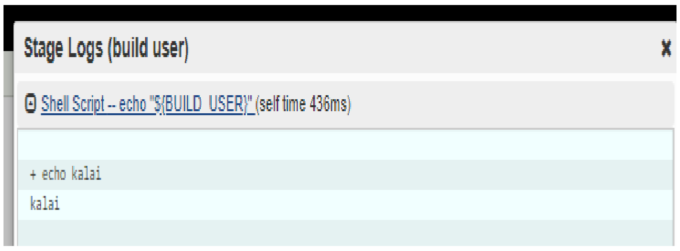
user build vars

This plugin is used to set user build variables: jenkins user name and id.

Update information obtained: 15 hr ago

```
pipeline {
  agent any

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

it will pull the user name



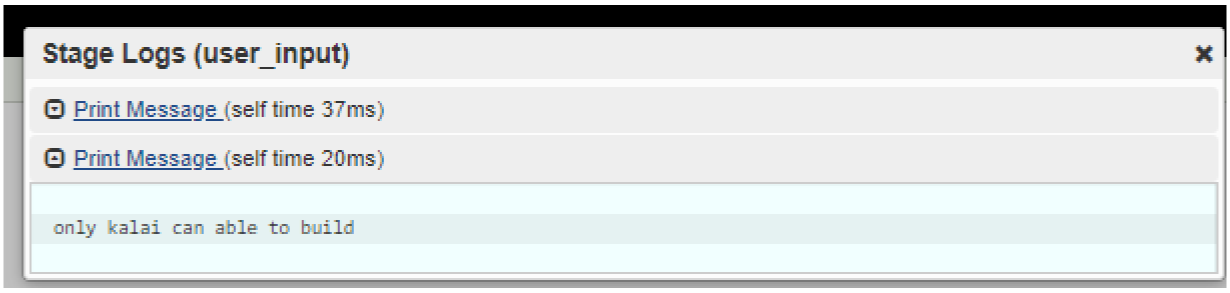
Stage Logs (build user)

Shell Script -- echo "${BUILD_USER}" (self time 436ms)

```
+ echo kalai
kalai
```

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"
                                                             }
                                                           }
                                                }
                                    }
                        }
            }
}
```



Stage Logs (user_input)

Print Message (self time 37ms)

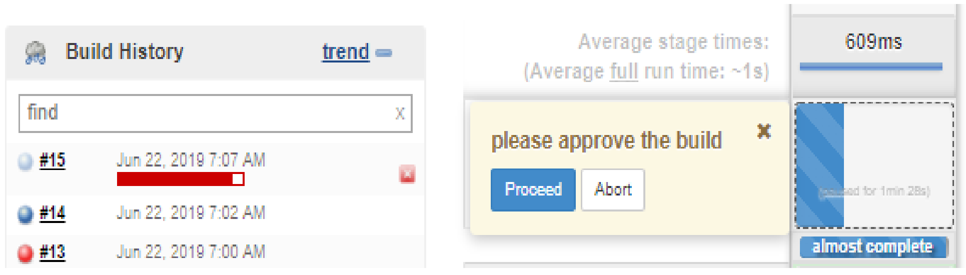
Print Message (self time 20ms)

```
only kalai can able to build
```

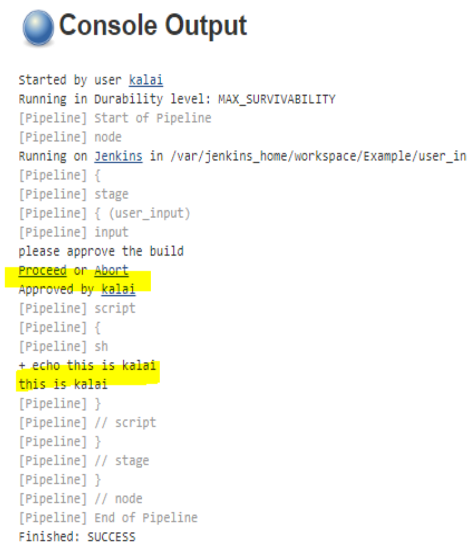
User input procced or abort

```
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



Console Output

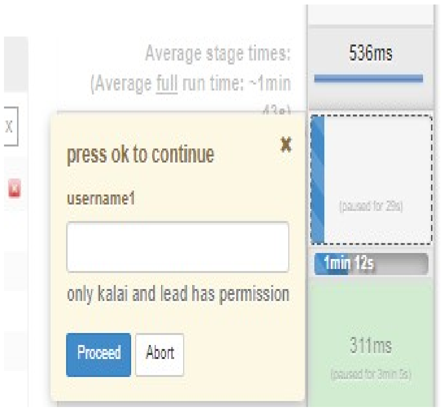
```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Example/user_in
[Pipeline] {
[Pipeline] stage
[Pipeline] { (user_input)
[Pipeline] input
please approve the build
Proceed or Abort
Approved by kalai
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ echo this is kalai
this is kalai
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Read Input From Specific user:

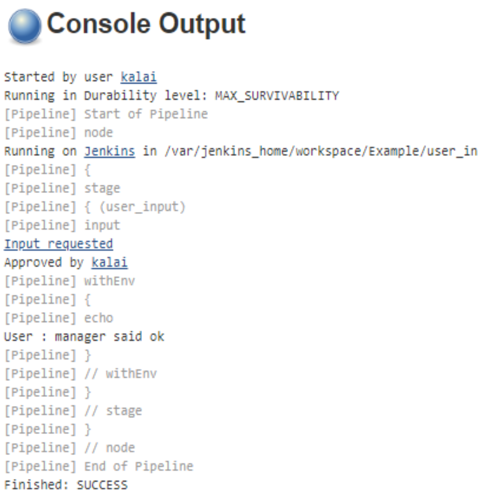
we can give list of user permission to proceed, other cant give proceed and get specific string from submitter.

```
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:


Average stage times:
(Average full run time: ~1min 43s)
536ms
press ok to continue
username1
only kalai and lead has permission
Proceed
Abort
(paused for 29s)
1min 12s
311ms
(paused for 3min 5s)

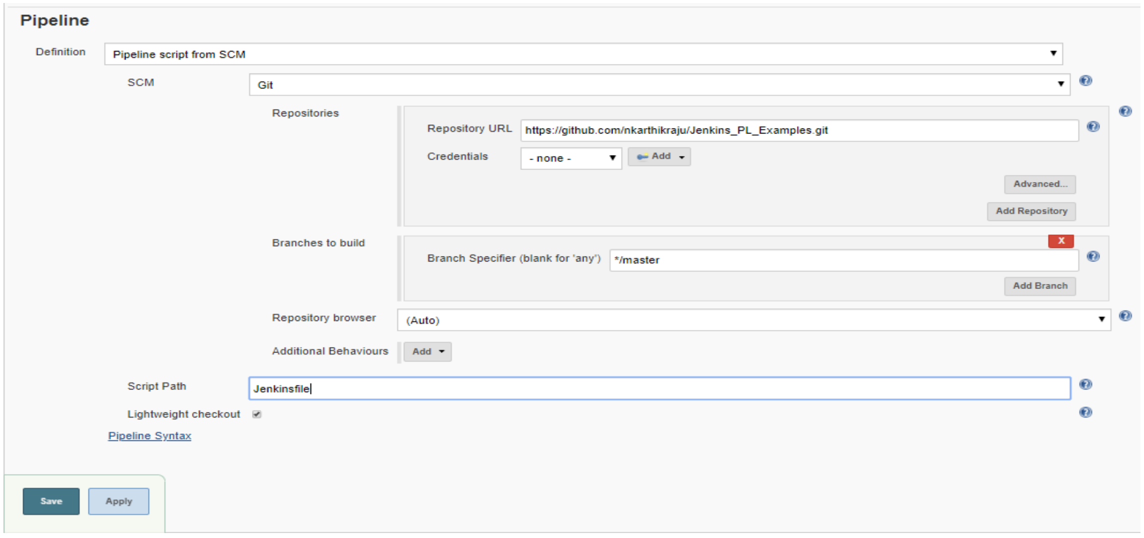
here is said manager


Console Output

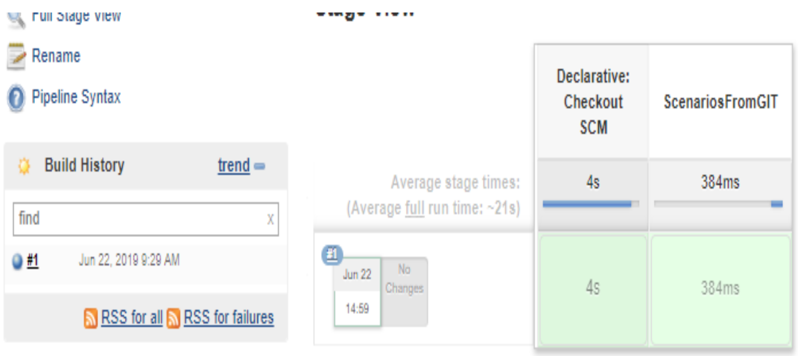
```
Started by user kalai
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Example/user_in
[Pipeline] {
[Pipeline] stage
[Pipeline] { (user_input)
[Pipeline] input
Input requested
Approved by kalai
[Pipeline] 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.

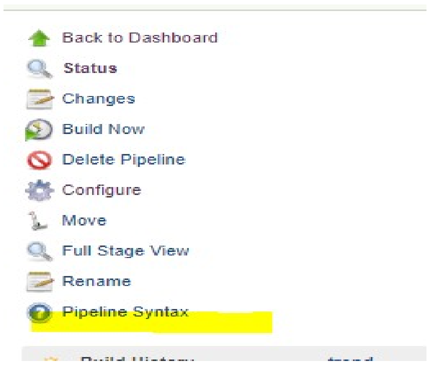

Pipeline

Definition: Pipeline script from SCM
SCM: Git
Repositories
Repository URL: https://github.com/nkarthikraju/Jenkins_PL_Examples.git
Credentials: - none -
Branches to build
Branch Specifier (blank for 'any'): */master
Repository browser: (Auto)
Additional Behaviours
Script Path: Jenkinsfile
Lightweight checkout: ✓
Pipeline Syntax

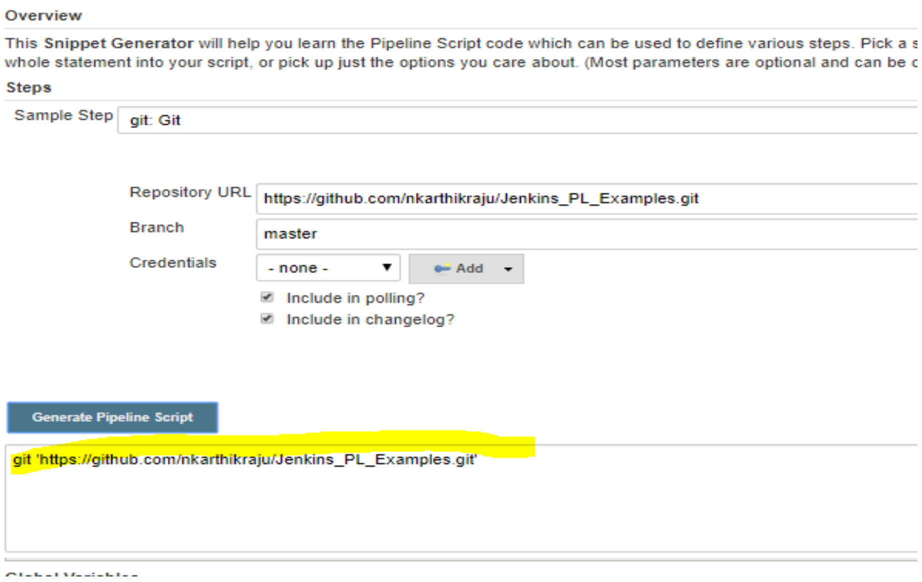


GIT checkout with the help of Pipeline syntax option

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


Overview

This Snippet Generator will help you learn the Pipeline Script code which can be used to define various steps. Pick a s
whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be c

Steps

Sample Step	git: Git
Repository URL	https://github.com/nkarthikraju/Jenkins_PL_Examples.git
Branch	master
Credentials	- none -
	✓ Include in polling?
	✓ Include in changelog?

```
git 'https://github.com/nkarthikraju/Jenkins_PL_Examples.git'
```

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.

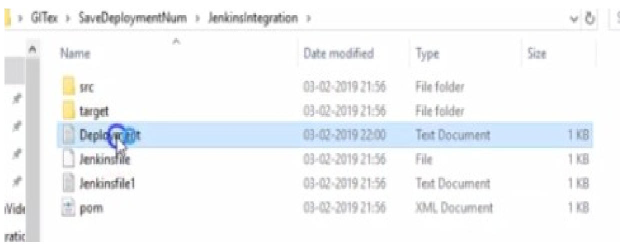
```
pipeline{
        agent any
        stages{
                stage('commitcode'){
                        steps{
                                git 'https://github.com/kalaiarasan33/jenkins_commit.git'
                                dir('tag_jenkins'){

                                        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

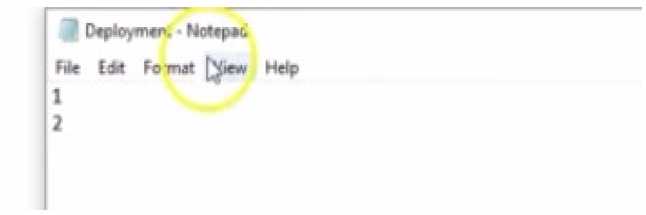
```
pipeline{
        agent any
        stages{
                stage('builnum'){
                        steps{
                                git 'https://github.com/kalaiarasan33/jenkins_commit.git'
                                sh "echo $BUILD_NUMBER >> Deployment.txt"
                       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

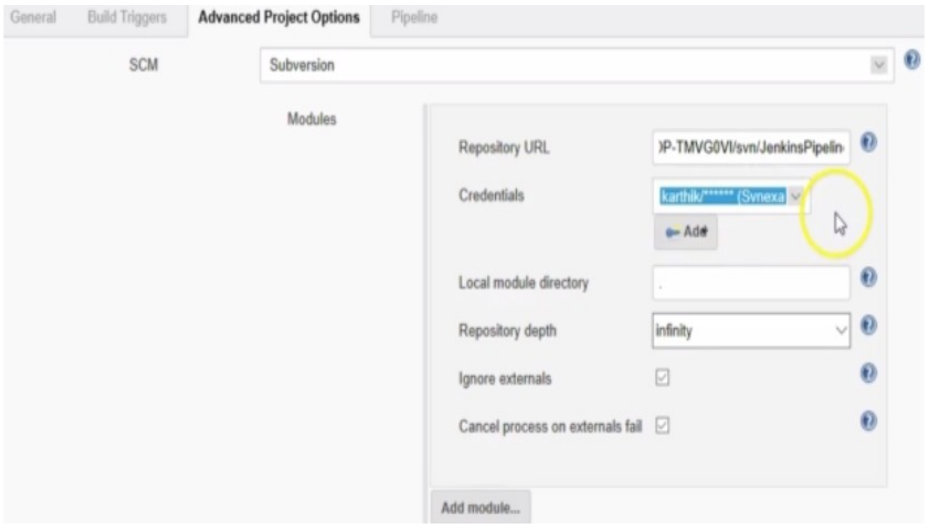


```
GITex > SaveDeploymentNum > JenkinsIntegration >

Name            Date modified       Type            Size
src             03-02-2019 21:56    File folder
target          03-02-2019 21:56    File folder
Deployment      03-02-2019 22:00    Text Document   1 KB
Jenkinsfile     03-02-2019 21:56    File            1 KB
Jenkinsfile1    03-02-2019 21:56    Text Document   1 KB
pom             03-02-2019 21:56    XML Document    1 KB
```



```
Deployment - Notepad
File  Edit  Format  View  Help
1
2
```

SVN examples



SCM: Subversion

Modules

Repository URL:)P-TMVG0VI/svn/JenkinsPipelin

Credentials: karthik/****** (Svnexa

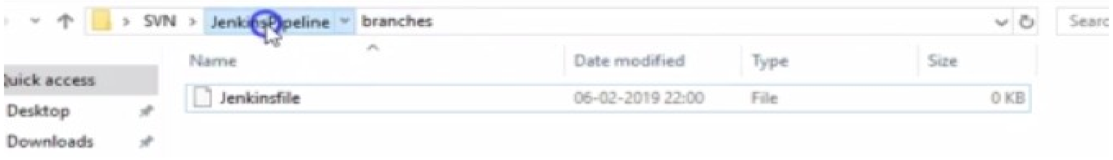
Local module directory: .

Repository depth: infinity

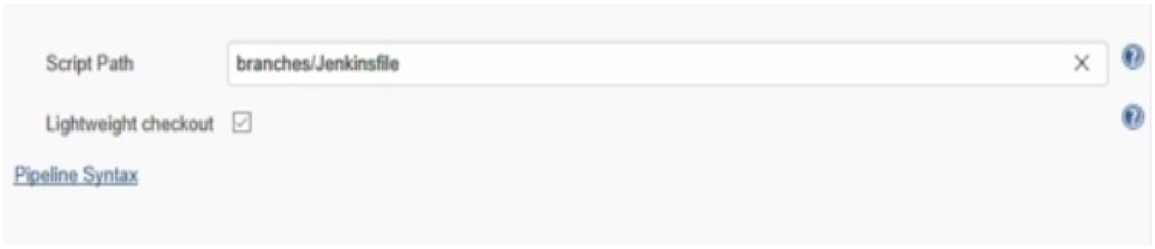
Ignore externals ☑

Cancel process on externals fail ☑

jenkins file in brancher folder



Name	Date modified	Type	Size
Jenkinsfile	06-02-2019 22:00	File	0 KB



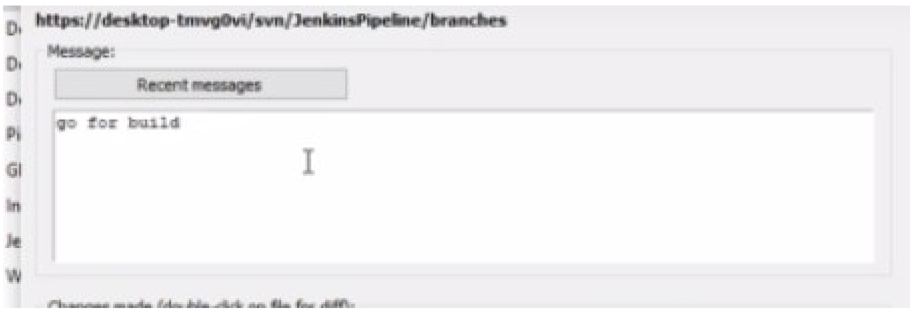
Script Path: branches/Jenkinsfile

Lightweight checkout ☑

Stages based on When condition

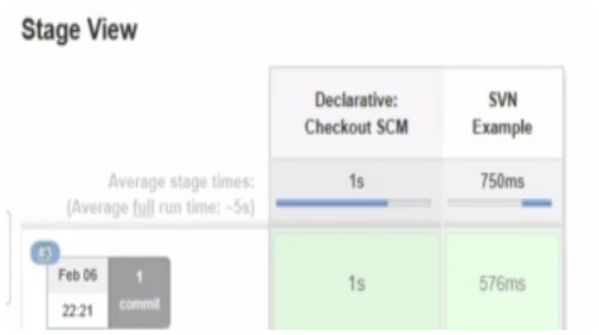
```
pipeline{
        agent any
        stages{
                stage('svn'){
                when{
                        changelog 'build'
                }
                        steps{

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

                                }
                        }
                }
        }
}
```

If you commit with build message, then it will build


https://desktop-tmvg0vi/svn/JenkinsPipeline/branches

Message:

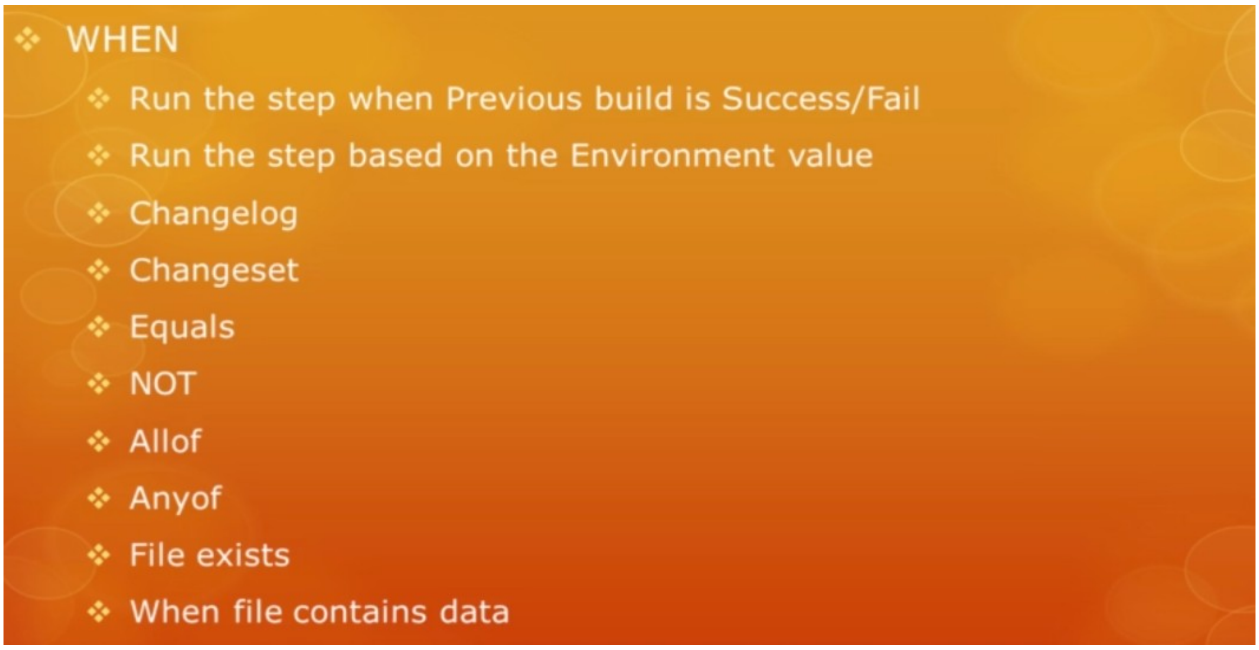
```
go for build
```


Stage View

	Declarative: Checkout SCM	SVN Example
Average stage times: (Average full run time: ~5s)	1s	750ms
#3 Feb 06 22:21 1 commit	1s	576ms

WHEN

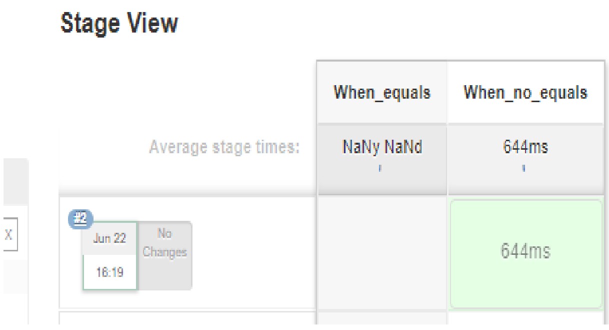
When is similar to If condition, but its more adavanced with in-built condition


- WHEN
  - Run the step when Previous build is Success/Fail
  - Run the step based on the Environment value
  - Changelog
  - Changeset
  - Equals
  - NOT
  - Allof
  - Anyof
  - File exists
  - When file contains data

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

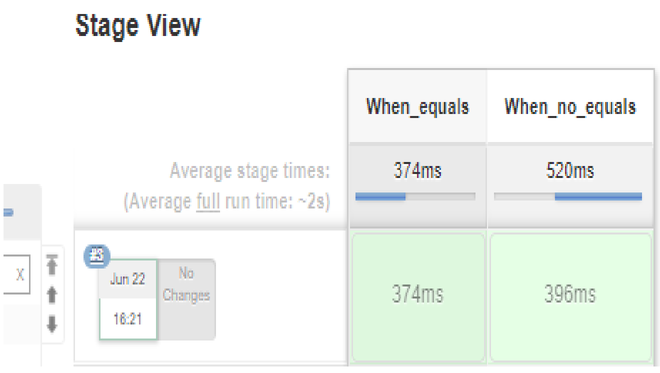


Stage View

	When_equals	When_no_equals
Average stage times:	NaNy NaNd	644ms
#2 Jun 22 16:19 No Changes		644ms

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

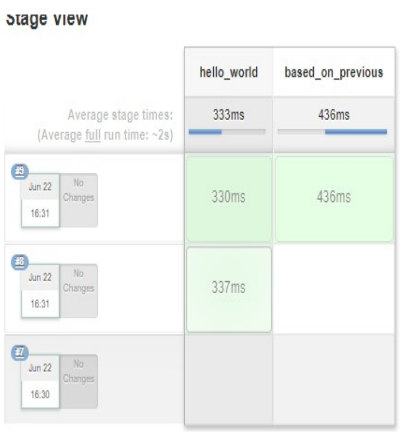
```
pipeline{
        agent any
        environment{
                Tool="Jenkins"
        }
        stages{
                stage('When_equals'){
                when{
                        equals expected:'Jenkins' , actual: "$Tool"
                }
                        steps{

                                        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"
                                        """


                                }
                        }
                }
        }
```

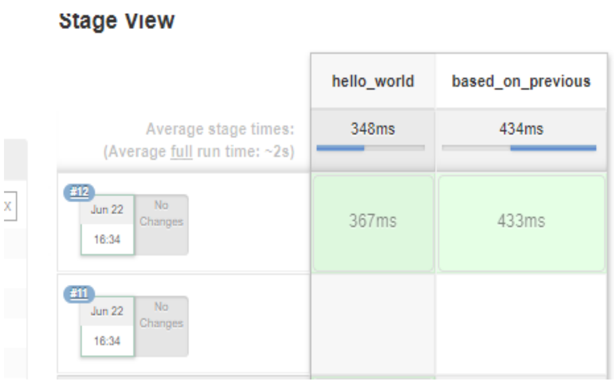


Stage View

	hello_world	based_on_previous
Average stage times: (Average full run time: ~2s)	333ms	436ms
#9 Jun 22 16:31 No Changes	330ms	436ms
#8 Jun 22 16:31 No Changes	337ms	
#7 Jun 22 16:30 No Changes		

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)

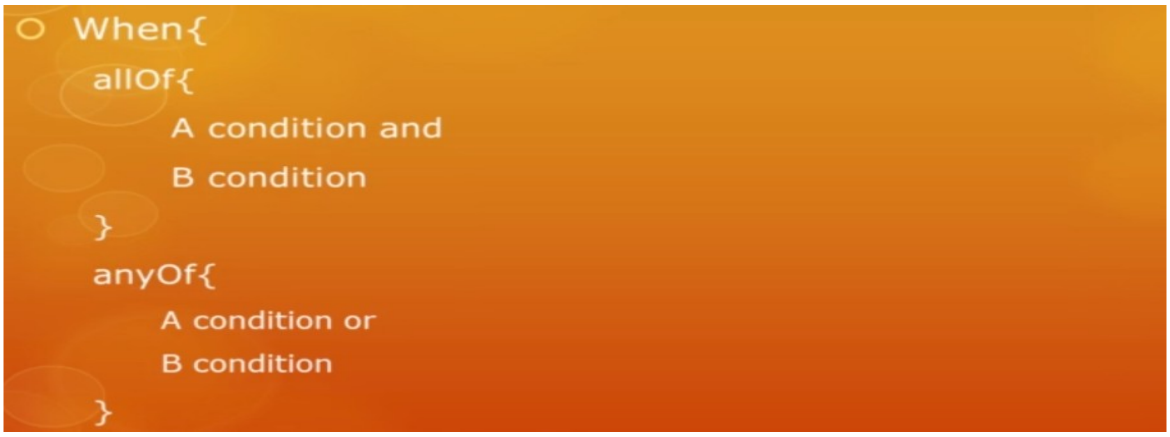
when they commited with message build, it will get execute

```
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)

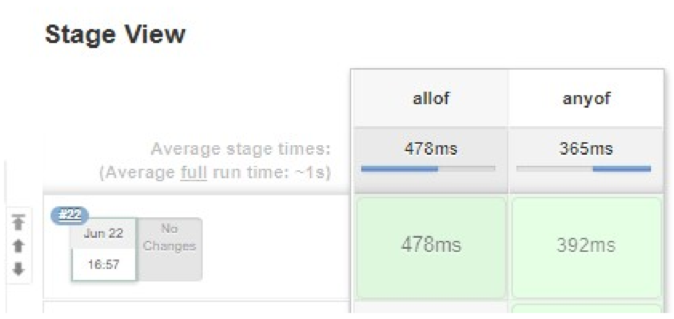
```
pipeline{
          agent any
          stages{
                    stage('pythonfile'){
                    when{
                              changeset '*.py'
                    }
                              steps{

                                                  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



```
○ When{
    allOf{
        A condition and
        B condition
    }
    anyOf{
        A condition or
        B condition
    }
```

```
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

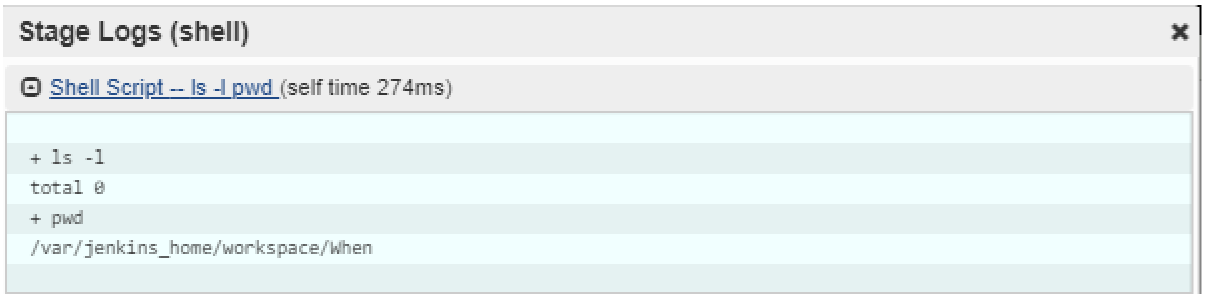
if you want to skip the stage or skip for few days.

```
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
                                """


                        }
                }
        }
}
```



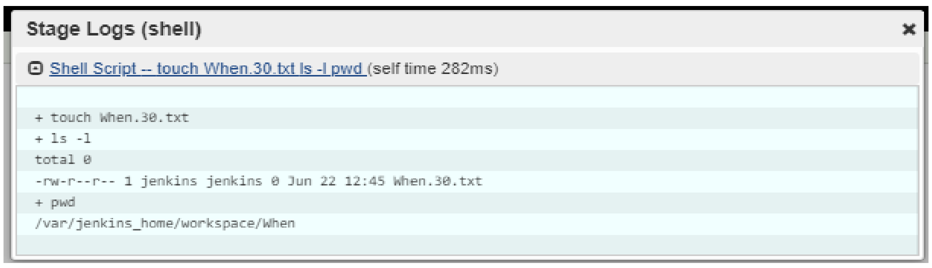
Stage Logs (shell)

Shell Script -- ls -l pwd (self time 274ms)

```
+ ls -l
total 0
+ pwd
/var/jenkins_home/workspace/When
```

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
                                """


                        }
                }
        }
}
```



Stage Logs (shell)

Shell Script -- touch When.30.txt ls -l pwd (self time 282ms)

```
+ touch When.30.txt
+ ls -l
total 0
-rw-r--r-- 1 jenkins jenkins 0 Jun 22 12:45 When.30.txt
+ pwd
/var/jenkins_home/workspace/When
```

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\\

                                """
                        }

                        }
                }
        }
}
```