DVS Technologies Aws & Devops

Compiled and Scrutinized by

Mr. Shaan Shaik

(Senior DevOps Lead)

Words To The Students

Though we have taken utmost efforts to present you this book error free, but still it may contain some errors or mistakes. Students are encouraged to bring, if there are any mistakes or errors in this document to our notice. So that it may be rectified in the next edition of this document.

"Suppressing your doubts is Hindering your growth".

We urge you to work hard and make use of the facilities we are providing to you, because there is no substitute for hard work. We wish you all the best for your future.

"The grass isn't greener on the other side; the grass is greener where you water it."

You and your suggestions are valuable to us; Help us to serve you better. In case of any suggestions, grievance, or complaints, please feel free to write us your suggestions, grievance and feedback on the following

Dvs.training@gmail.com

DVS Technologies Aws & Devops

1. Introduction

What is Jenkins Pipeline:

A Pipeline which provides an extensible set of tools for modeling simple-to-complex delivery of application. Pipelines are defined in Domain Specific Language (DSL) syntax . Typically, the definition of a Jenkins Pipeline is written into a text file (called a Jenkinsfile) which in turn is checked into a project's source control repository. This is the foundation of "Pipeline-as-Code"
which is stored in the git and helps us to version and review it like any other code.

Creating a Jenkinsfile provides a number of immediate benefits:

- **Automatically create Pipelines for all Branches and Pull Requests**
- **Code review/iteration on the Pipeline**
- **Audit trail for the Pipeline**
- **Single source of truth**

Note: generally considered best practice to define the Pipeline in a Jenkinsfile and check that in to source control.

There are two ways of pipeline scripting they are as follows.

1. Declarative
2. scripted

Examples :

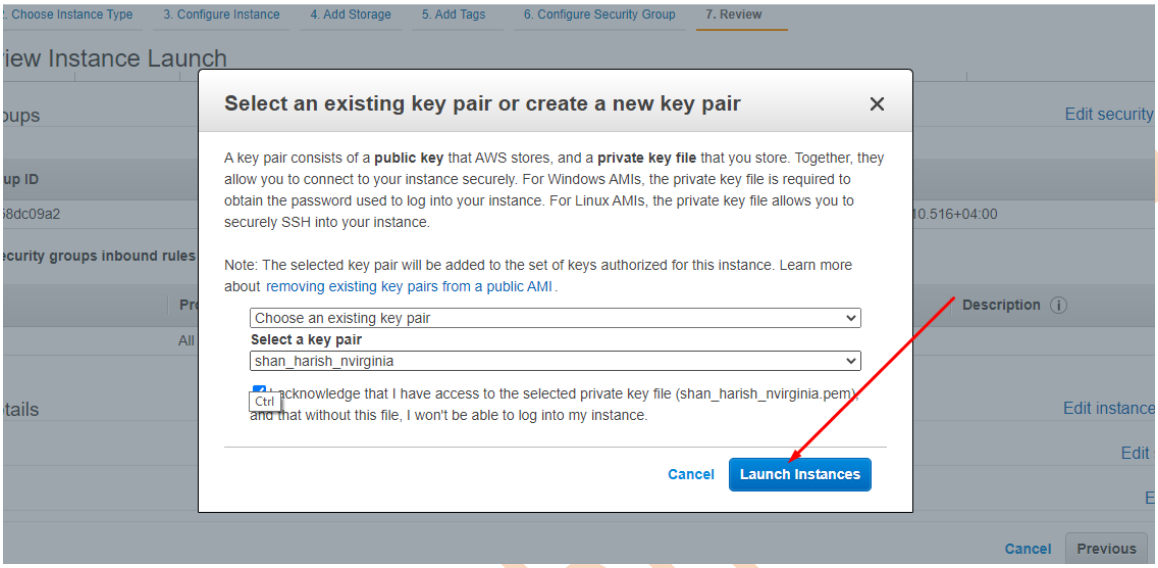
// Declarative //

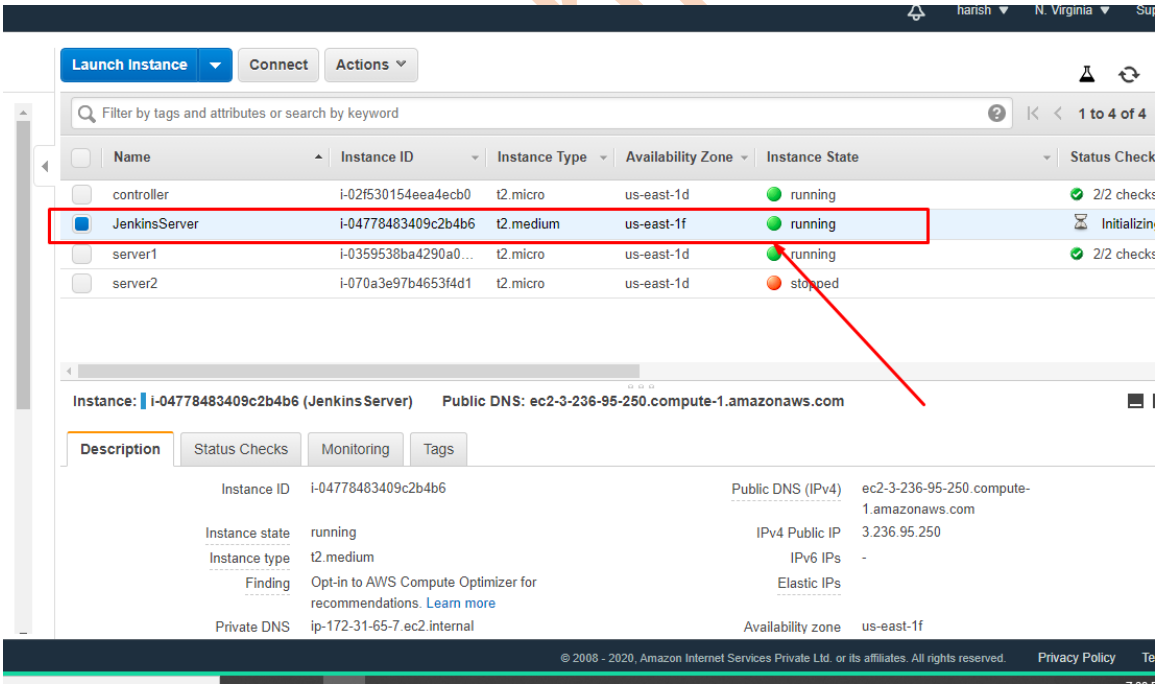
```
pipeline {
agent any ①
stages {
  stage('Build') { ②
  steps { ③
  sh 'make' ④
  }
  }
  stage('Test'){
  steps {
  sh 'make check'
  junit 'reports/**/*.xml' ⑤
```

```
  }
  }
  stage('Deploy') {
  steps {
  sh 'make publish'
  }
  }
  }
}

// Script //

node {
  stage('Build') {
  sh 'make'
  }
  stage('Test') {
  sh 'make check'
  junit 'reports/**/*.xml'
  }
  stage('Deploy') {
  sh 'make publish'
  }
}
```

2. Installation & Configuration

Launch one instance





Jenkins Installation :

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
   https://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo yum clean all
sleep 5
sudo yum install jenkins java-1.8.0-openjdk-devel -y
service jenkins start
chkconfig jenkins on
```

Jenkins Plugins Installation :

Script :

vim installplugins.sh

```
#!/bin/bash

set -e

plugin_dir=/var/lib/jenkins/plugins
file_owner=jenkins.jenkins

mkdir -p /var/lib/jenkins/plugins

installPlugin() {
  if [ -f ${plugin_dir}/${1}.hpi -o -f ${plugin_dir}/${1}.jpi ]; then
    if [ "$2" == "1" ]; then
      return 1
    fi
    echo "Skipped: $1 (already installed)"
    return 0
  else
    echo "Installing: $1"
    curl -L --silent --output ${plugin_dir}/${1}.hpi  https://updates.jenkins-
ci.org/latest/${1}.hpi
    return 0
  fi
}

while read -r plugin
```

```
do
   installPlugin "$plugin"
done < "/tmp/plugins.txt"

changed=1
maxloops=100

while [ "$changed"  == "1" ]; do
  echo "Check for missing dependecies ..."
  if  [ $maxloops -lt 1 ] ; then
    echo "Max loop count reached - probably a bug in this script: $0"
    exit 1
  fi
  ((maxloops--))
  changed=0
  for f in ${plugin_dir}/*.hpi ; do
    deps=$( unzip -p ${f} META-INF/MANIFEST.MF | tr -d '\r' | sed -e ':a;N;$!ba;s/\n //g' |
grep -e "^Plugin-Dependencies: " | awk '{ print $2 }' | tr ',' '\n' | awk -F ':' '{ print $1 }' | tr
'\n' ' ' )
    for plugin in $deps; do
      installPlugin "$plugin" 1 && changed=1
    done
  done
done

echo "fixing permissions"

chown ${file_owner} ${plugin_dir} -R

echo "all done"
```

Plugins list :

vim /tmp/plugins.txt

```
ace-editor
amazon-ecr
ant
antisamy-markup-formatter
apache-httpcomponents-client-4-api
authentication-tokens
aws-credentials
aws-java-sdk
```

blueocean
blueocean-autofavorite
blueocean-bitbucket-pipeline
blueocean-commons
blueocean-dashboard
blueocean-display-url
blueocean-events
blueocean-git-pipeline
blueocean-github-pipeline
blueocean-i18n
blueocean-jira
blueocean-jwt
blueocean-personalization
blueocean-pipeline-api-impl
blueocean-pipeline-editor
blueocean-pipeline-scm-api
blueocean-rest
blueocean-rest-impl
blueocean-web
bouncycastle-api
branch-api
build-pipeline-plugin
cloudbees-bitbucket-branch-source
cloudbees-folder
command-launcher
conditional-buildstep
config-file-provider
credentials
credentials-binding
display-url-api
docker-commons
docker-workflow
durable-task
email-ext
embeddable-build-status
external-monitor-job
favorite
git
git-client
git-server
github
github-api
github-branch-source
github-pullrequest

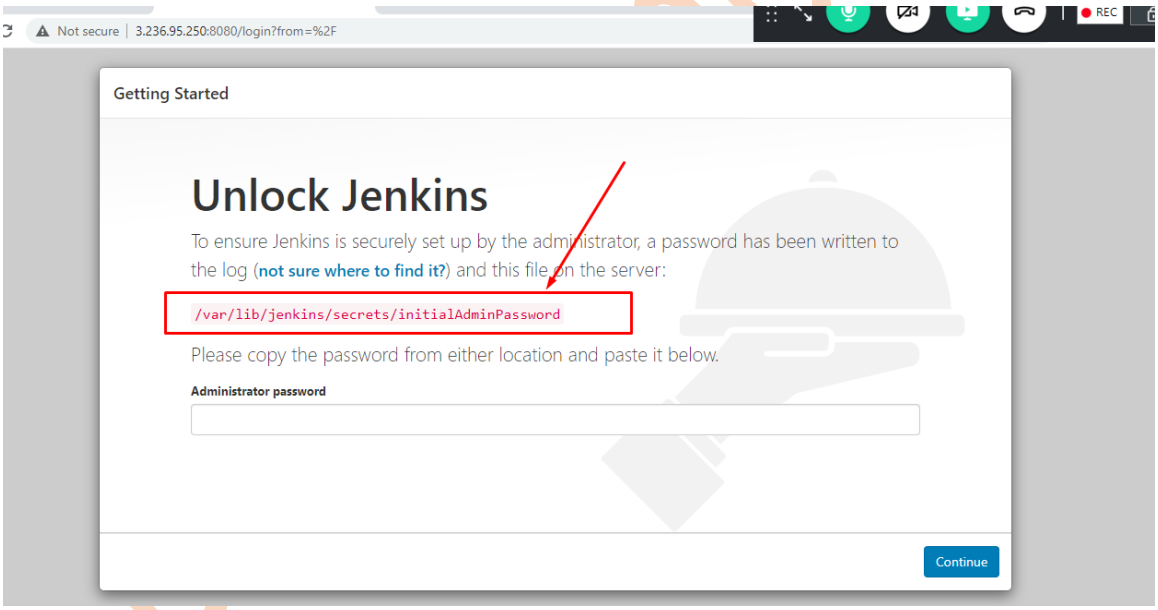
handy-uri-templates-2-api
htmlpublisher
jackson2-api
javadoc
jdk-tool
jenkins-design-language
jira
jquery
jquery-detached
jsch
junit
ldap
mailer
matrix-auth
matrix-project
maven-plugin
mercurial
momentjs
pam-auth
workflow-aggregator
workflow-api
workflow-basic-steps
workflow-cps
workflow-cps-global-lib
workflow-cps-global-lib-http
workflow-durable-task-step
workflow-job
workflow-multibranch
workflow-remote-loader
workflow-scm-step
workflow-step-api
workflow-support
parameterized-trigger
pipeline-aggregator-view
pipeline-aws
pipeline-bamboo
pipeline-build-step
pipeline-config-history
pipeline-cps-http
pipeline-dependency-walker
pipeline-deploymon
pipeline-giphy-api
pipeline-github
pipeline-github-lib

pipeline-githubnotify-step
pipeline-gitstatuswrapper
pipeline-graph-analysis
pipeline-input-step
pipeline-maven
pipeline-milestone-step
pipeline-model-api
pipeline-model-declarative-agent
pipeline-model-definition
pipeline-model-extensions
pipeline-multibranch-defaults
pipeline-npm
pipeline-rest-api
pipeline-restful-api
pipeline-stage-step
pipeline-stage-tags-metadata
pipeline-stage-view
pipeline-timeline
pipeline-utility-steps
plain-credentials
pubsub-light
run-condition
scm-api
script-security
slack
ssh
ssh-agent
ssh-credentials
ssh-slaves
structs
token-macro
variant
windows-slaves

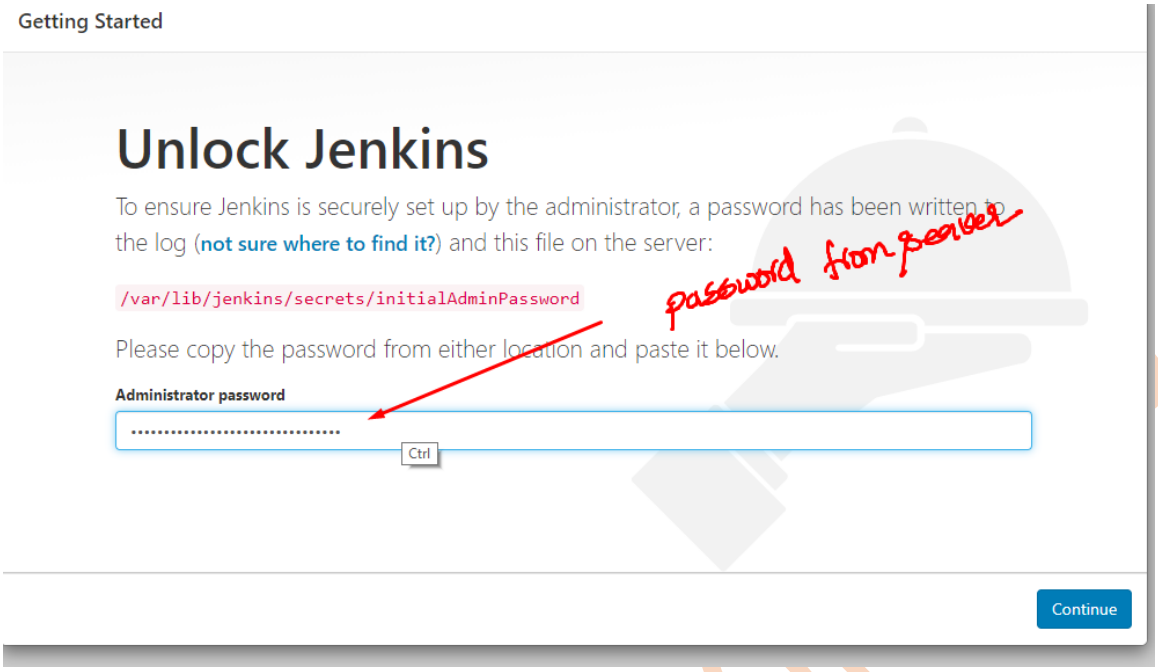
Installation:

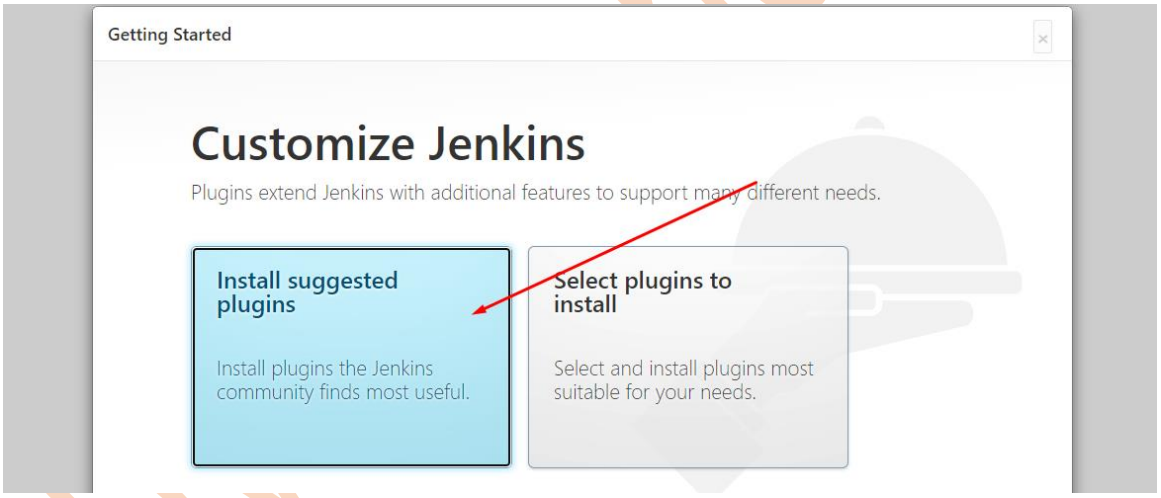
```
-rw-r--r-- 1 root root 1203 Oct 12 15:12 installplugins.sh
[root@jenkins ~]# bash installplugins.sh
Installing: ace-editor
Installing: amazon-ecr
Installing: ant
Installing: antisamy-markup-formatter
Installing: apache-httpcomponents-client-4-api
Installing: authentication-tokens
Installing: aws-credentials
Installing: aws-java-sdk
```

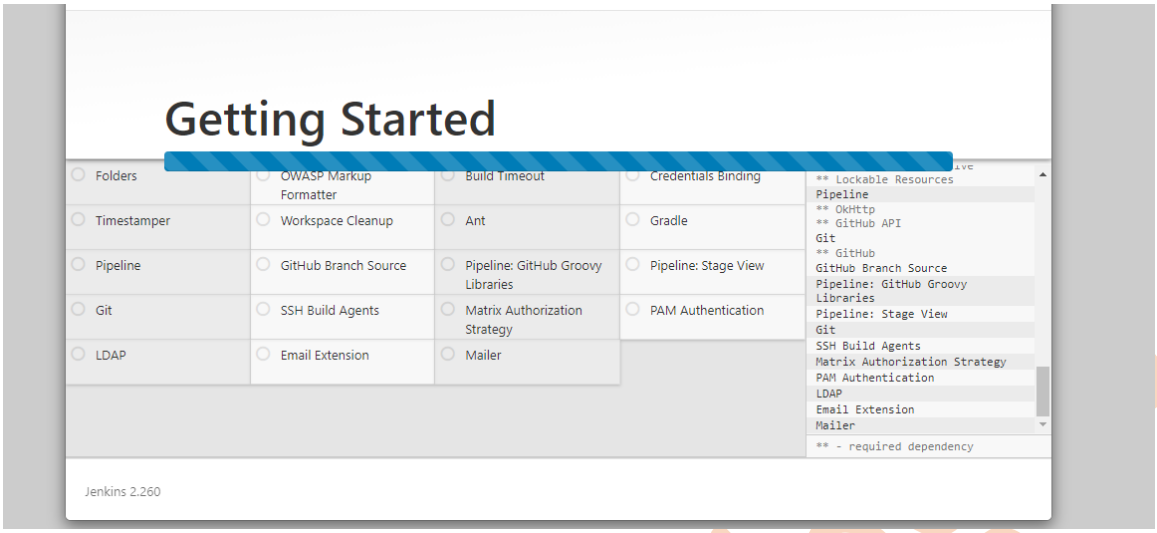
```
Installing: metrics
Check for missing dependecies ...
Installing: node-iterator-api
Check for missing dependecies ...
fixing permissions
all done
[root@jenkins ~]#
```

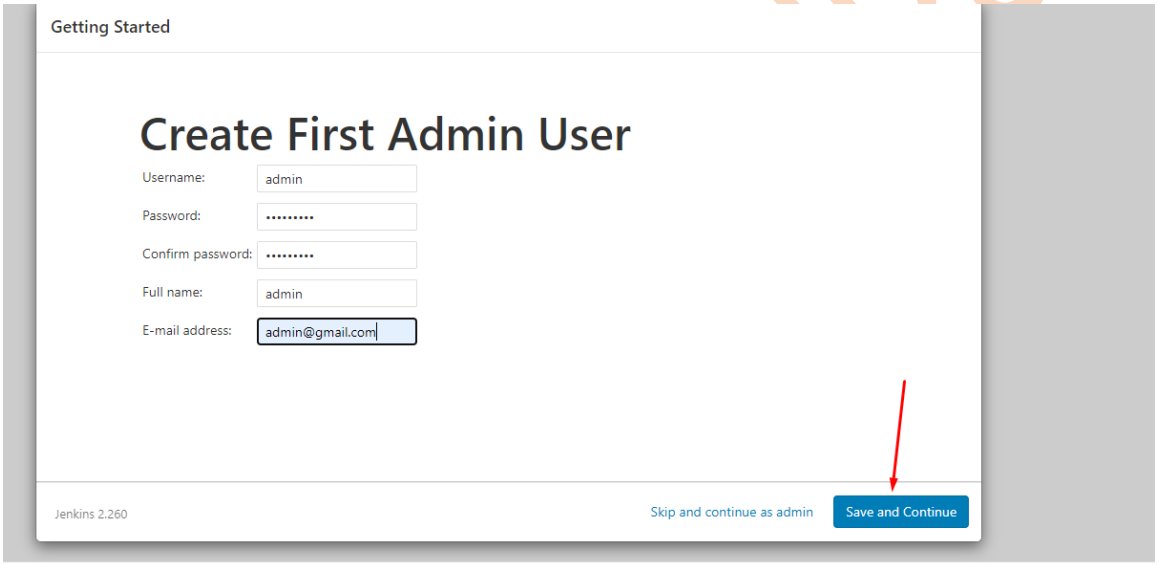


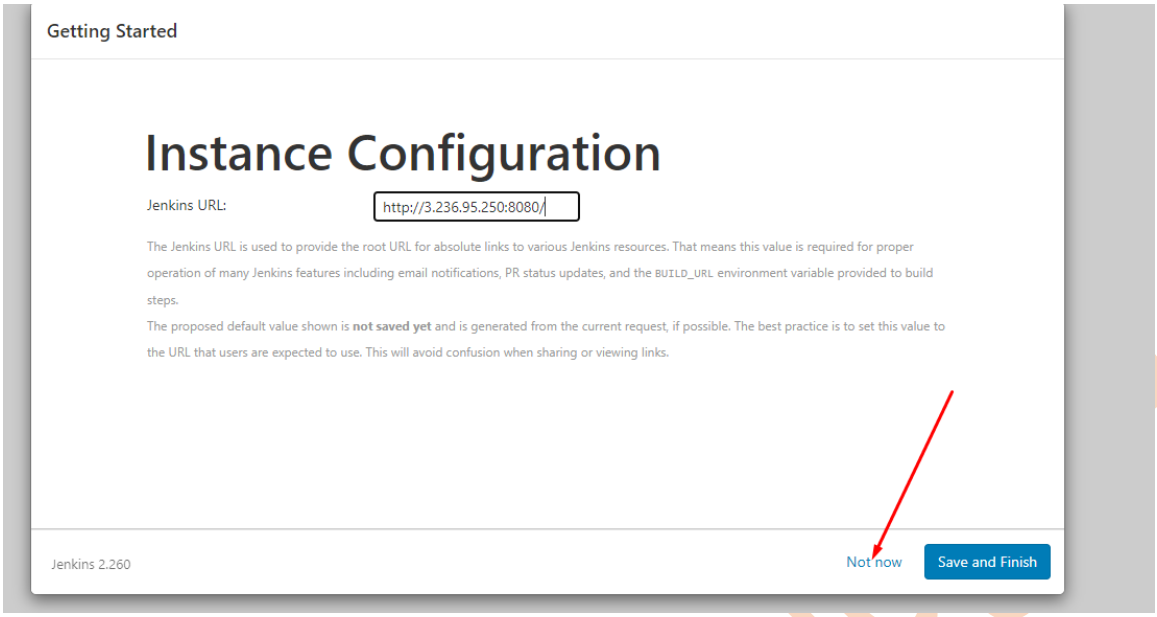

```
[root@jenkins ~]# cat /var/lib/jenkins/secrets/initialAdminPassword
8eb6276a54c8484ba6961b41120bc44d
[root@jenkins ~]#
```

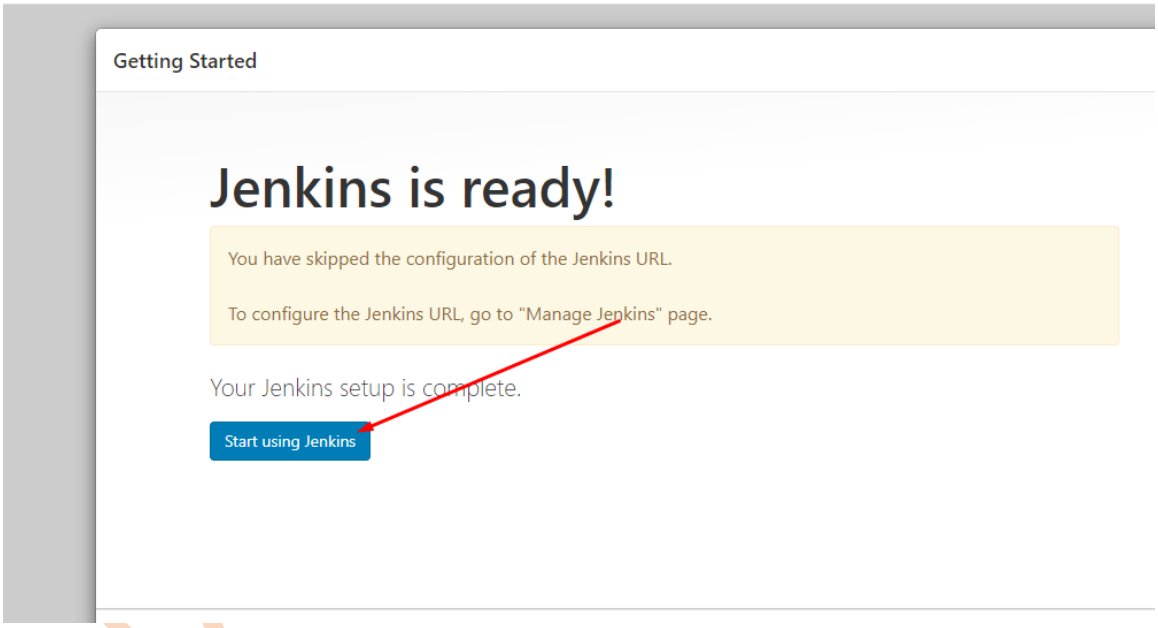

Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/lib/jenkins/secrets/initialAdminPassword
password from server
Please copy the password from either location and paste it below.
Administrator password
Ctrl
Continue

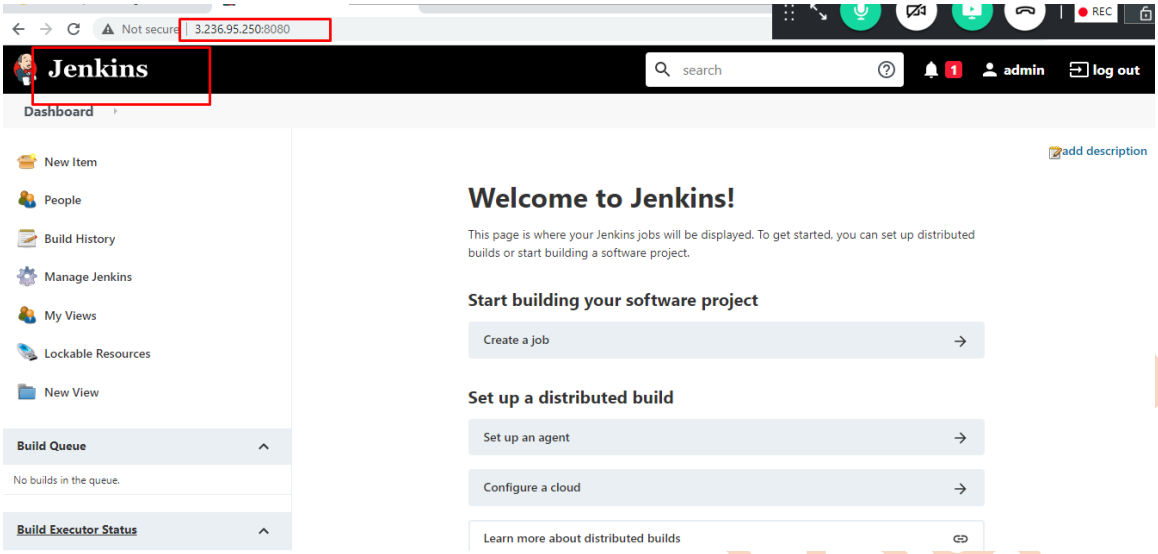

Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.


Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
** Lockable Resources
Pipeline
** OkHttp
** GitHub API
Git
** GitHub
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
** - required dependency
Jenkins 2.260


Getting Started
Create First Admin User
Username:
admin
Password:
Confirm password:
Full name:
admin
E-mail address:
admin@gmail.com
Jenkins 2.260
Skip and continue as admin
Save and Continue

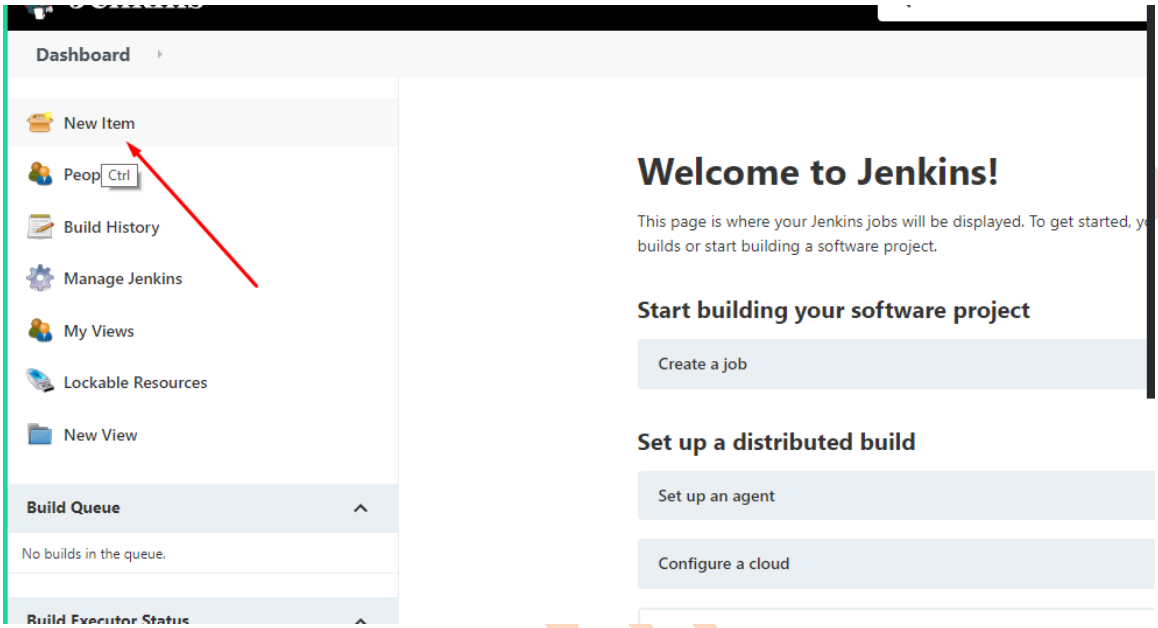

Getting Started
Instance Configuration
Jenkins URL:
http://3.236.95.250:8080/
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.
Jenkins 2.260
Not now
Save and Finish

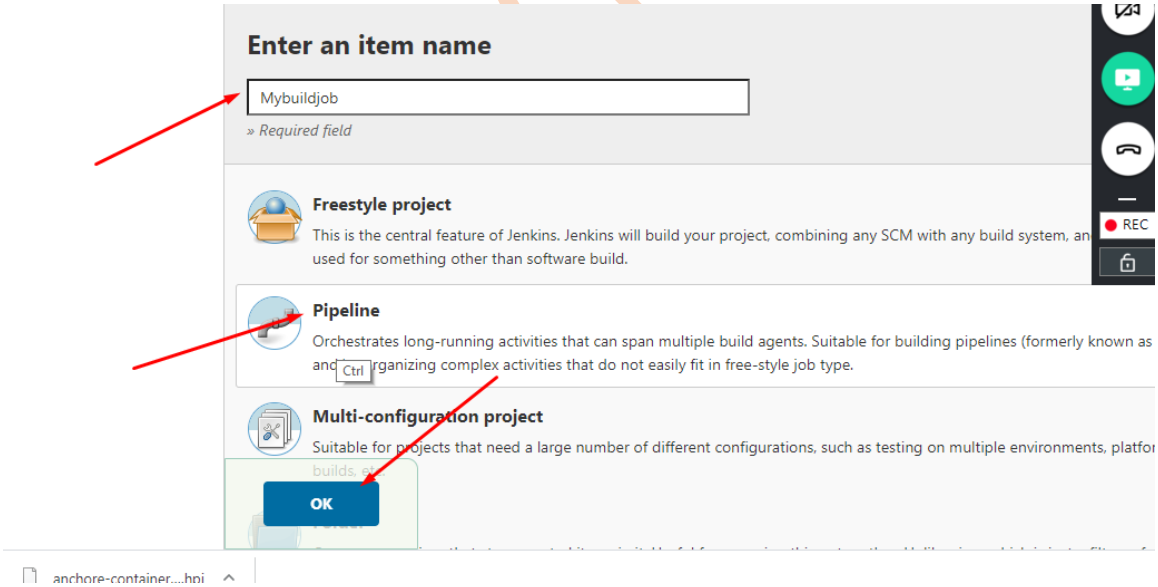

Getting Started
Jenkins is ready!
You have skipped the configuration of the Jenkins URL.
To configure the Jenkins URL, go to "Manage Jenkins" page.
Your Jenkins setup is complete.
Start using Jenkins

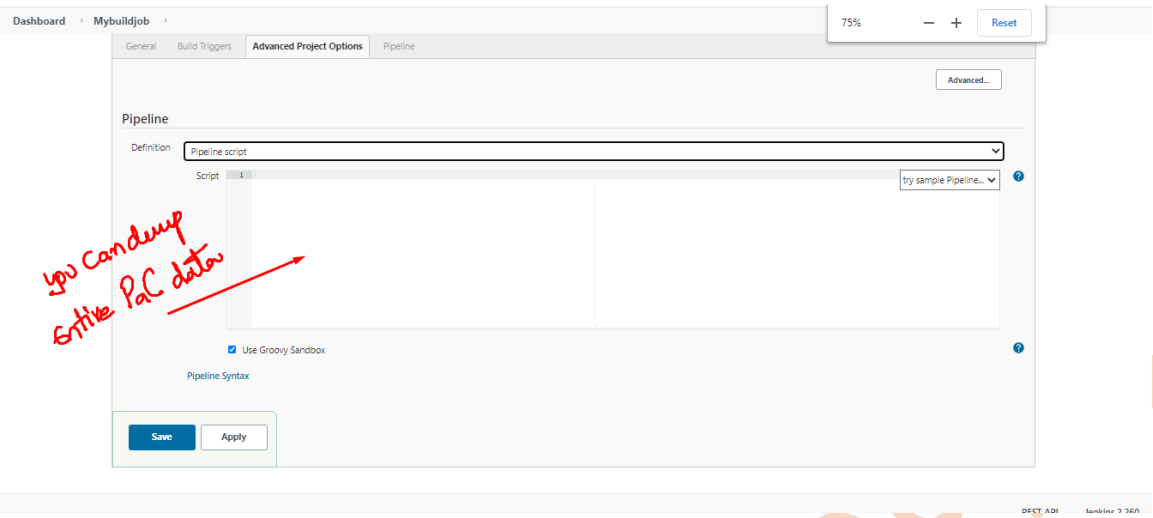


3. Working with Jenkins with PaC

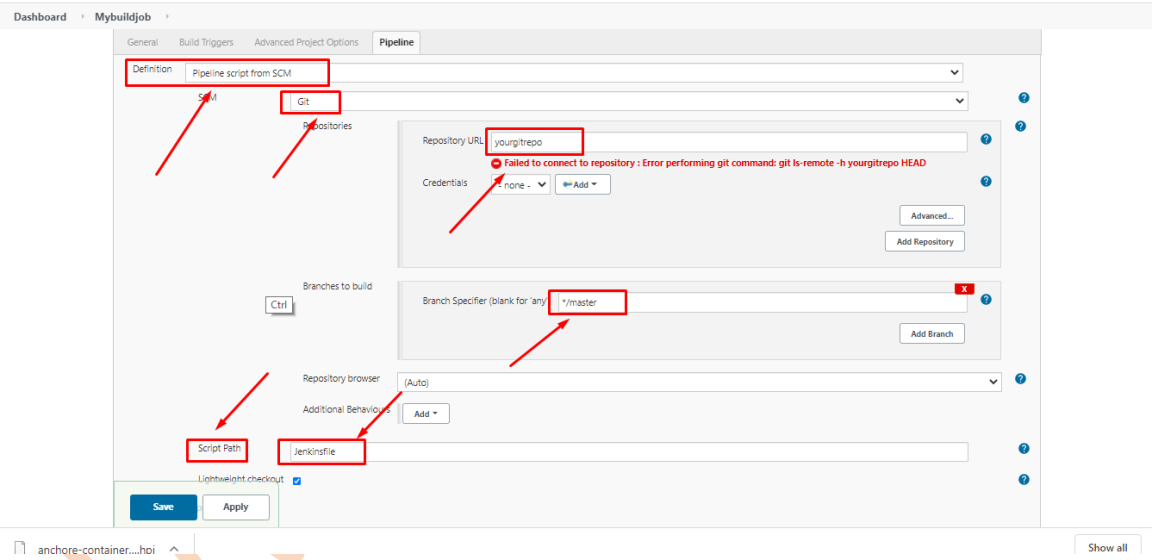
Executing a build job via Pipeline as code



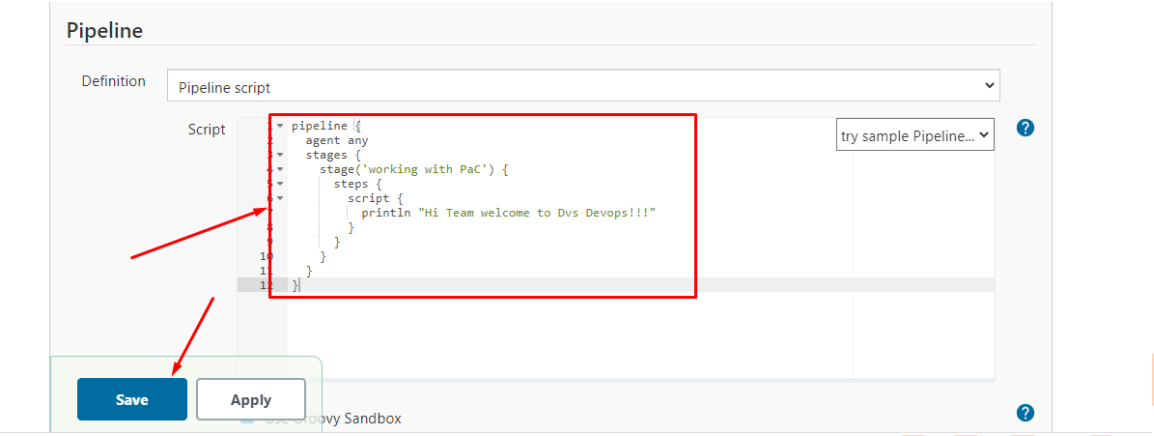


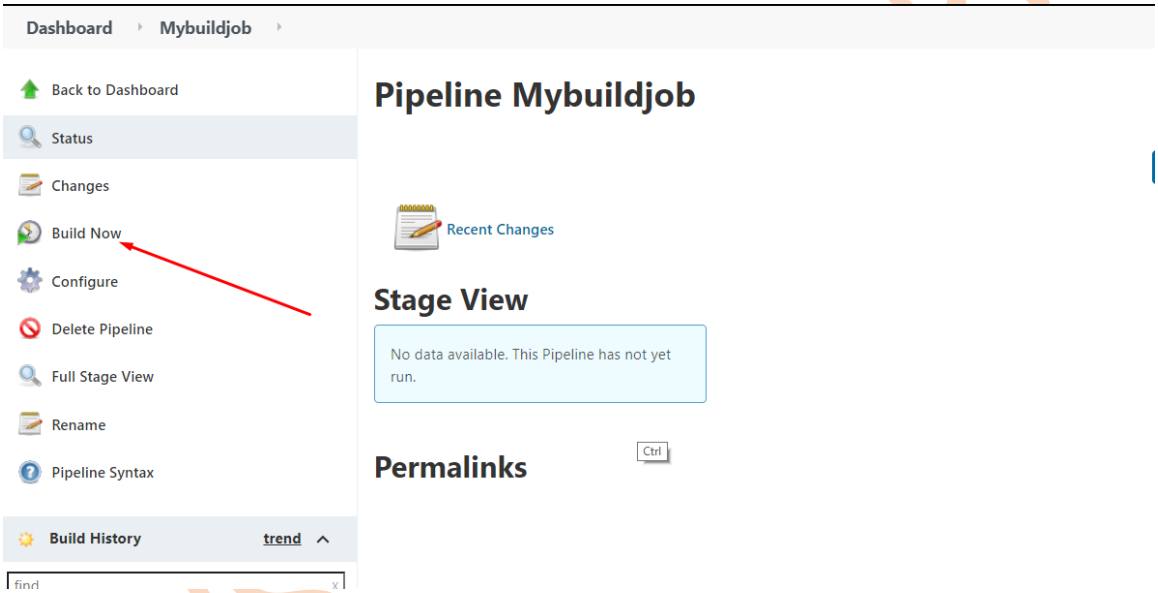


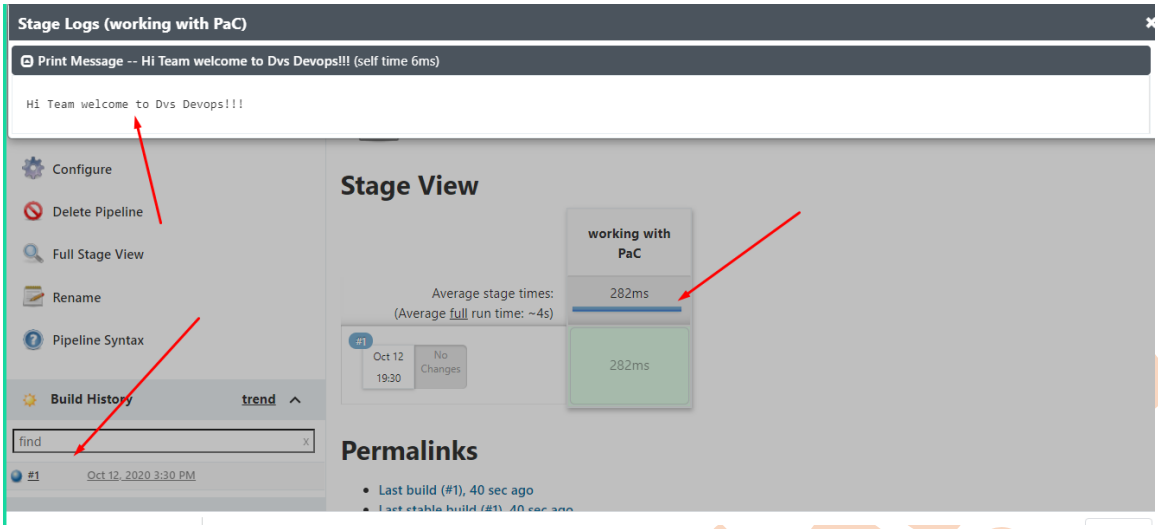
You can choose the above one or below one for your build job configuration but it always safe to store the data in GIT



Let's configure our build job with below data & try to run the build job:


Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
pipeline {
agent any
stages {
stage('working with PaC') {
steps {
script {
println "Hi Team welcome to Dvs Devops!!!"
}
}
}
}
}
Save
Apply
Sandbox

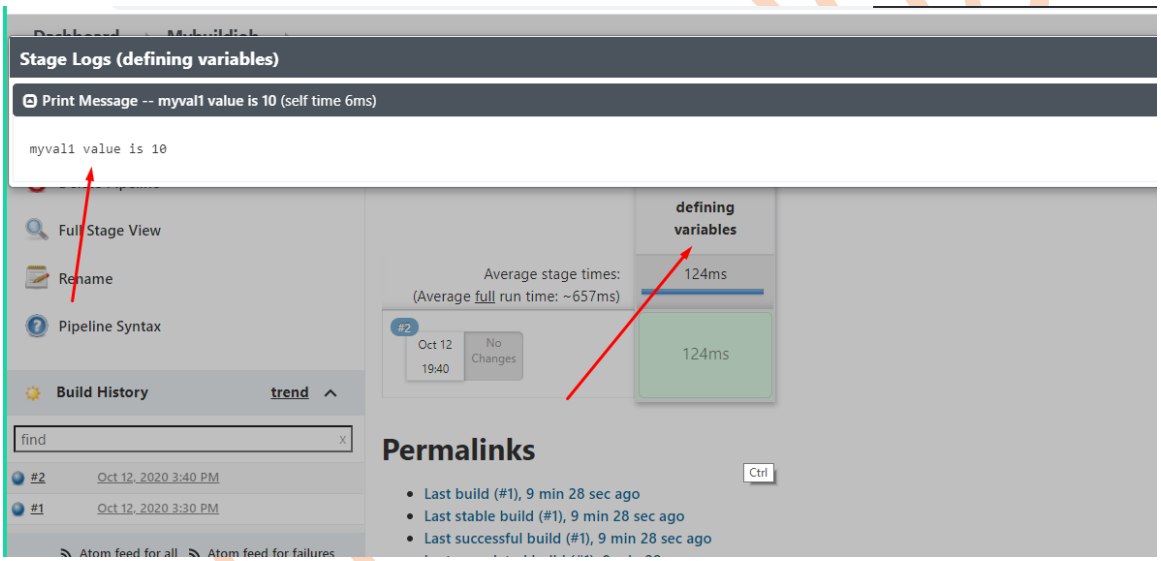

Dashboard
Mybuildjob
Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
find
Pipeline Mybuildjob
Recent Changes
Stage View
No data available. This Pipeline has not yet run.
Permalinks
Ctrl



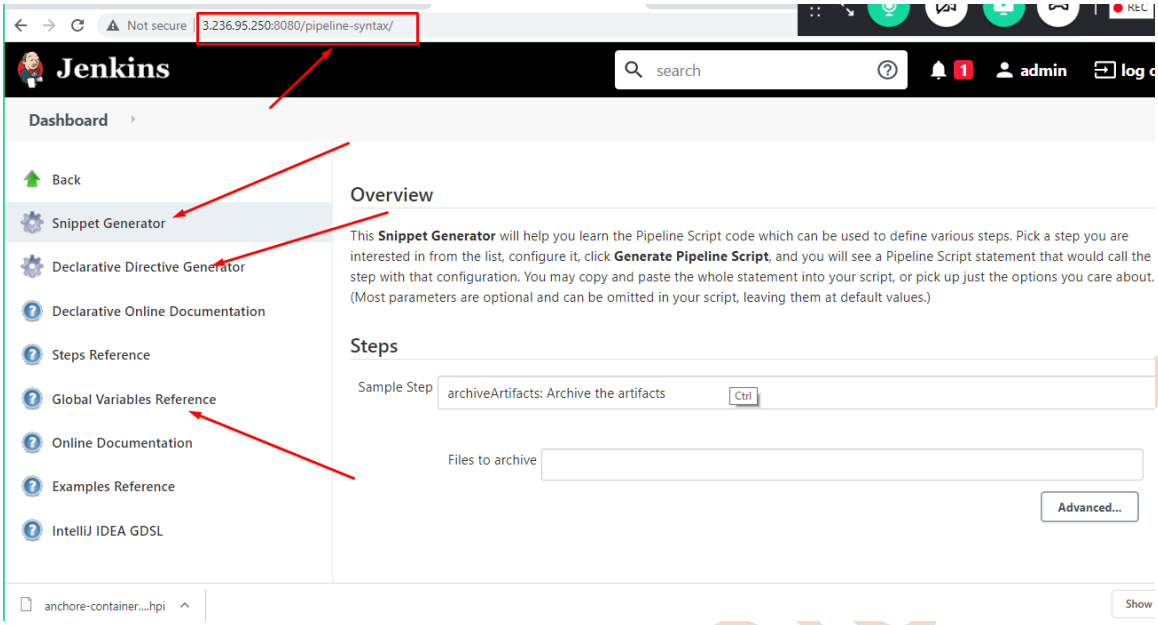
4. Variables

Variable initialization & accessing variable

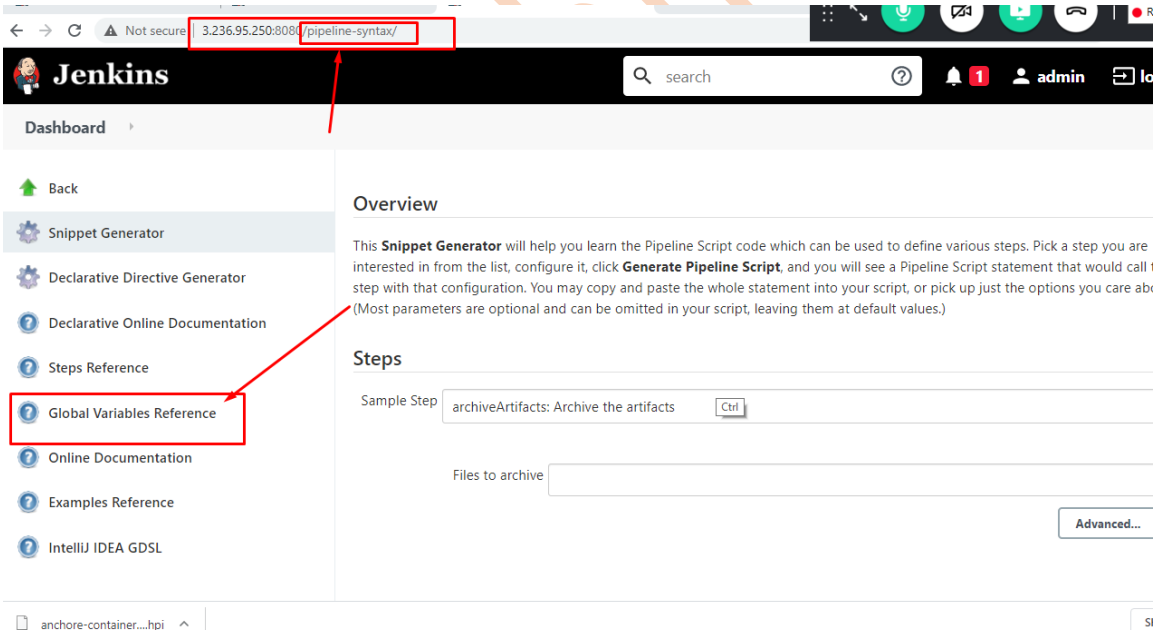
```
pipeline {
  agent any
  stages {
    stage('defining variables') {
      steps{
        script {
          def myval1=10
          println "myval1 value is ${myval1}"
        }
      }
    }
  }
}
```

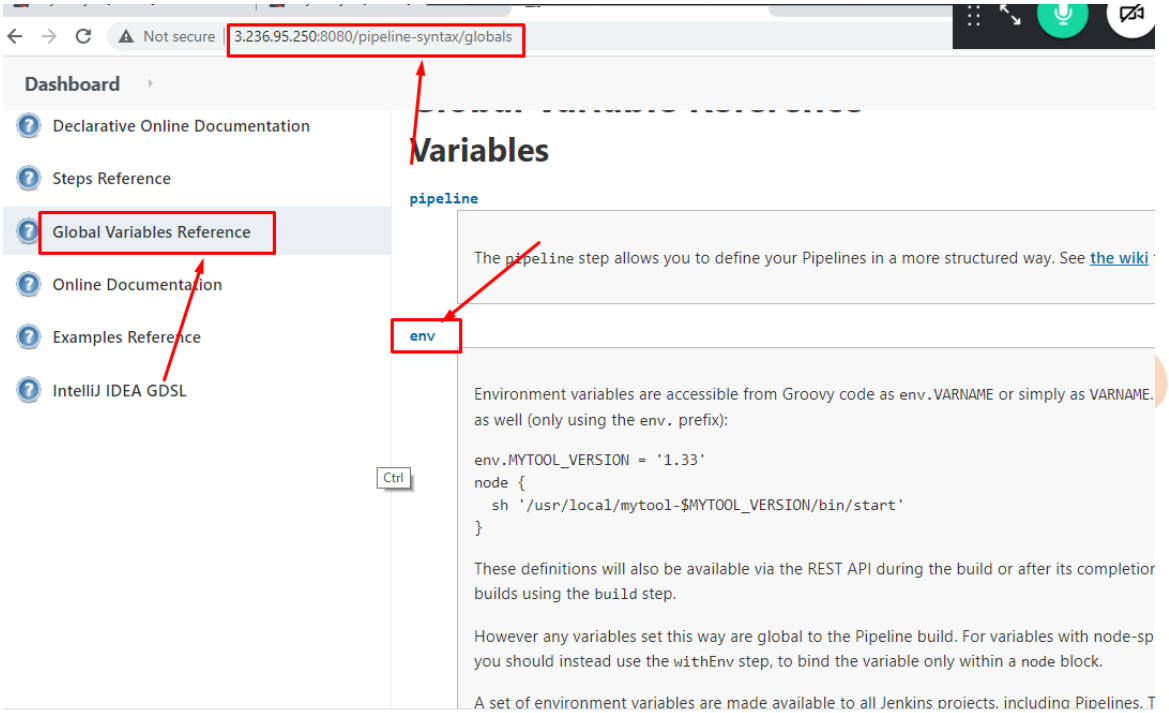


Syntax Generators:

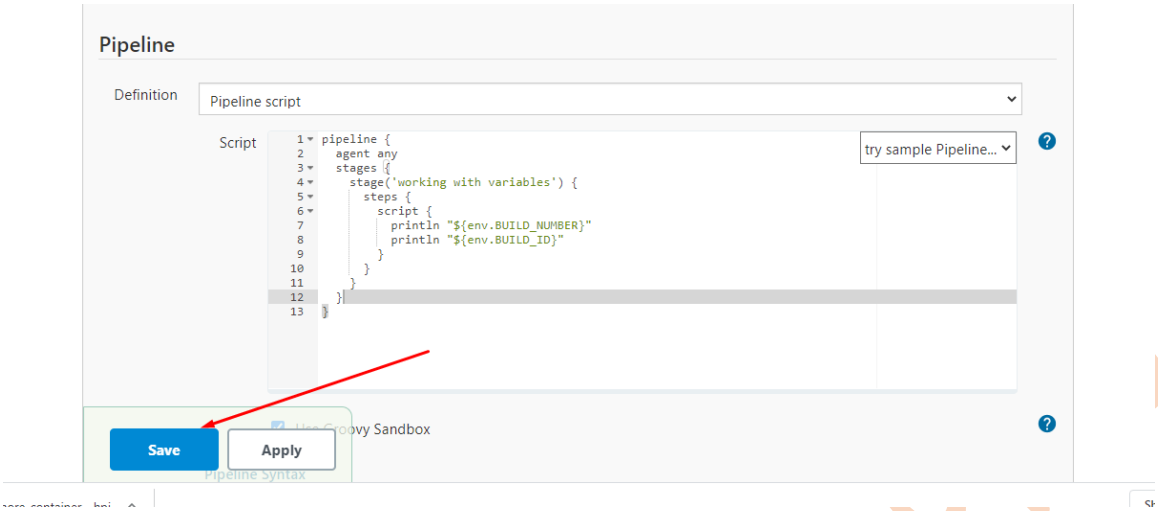


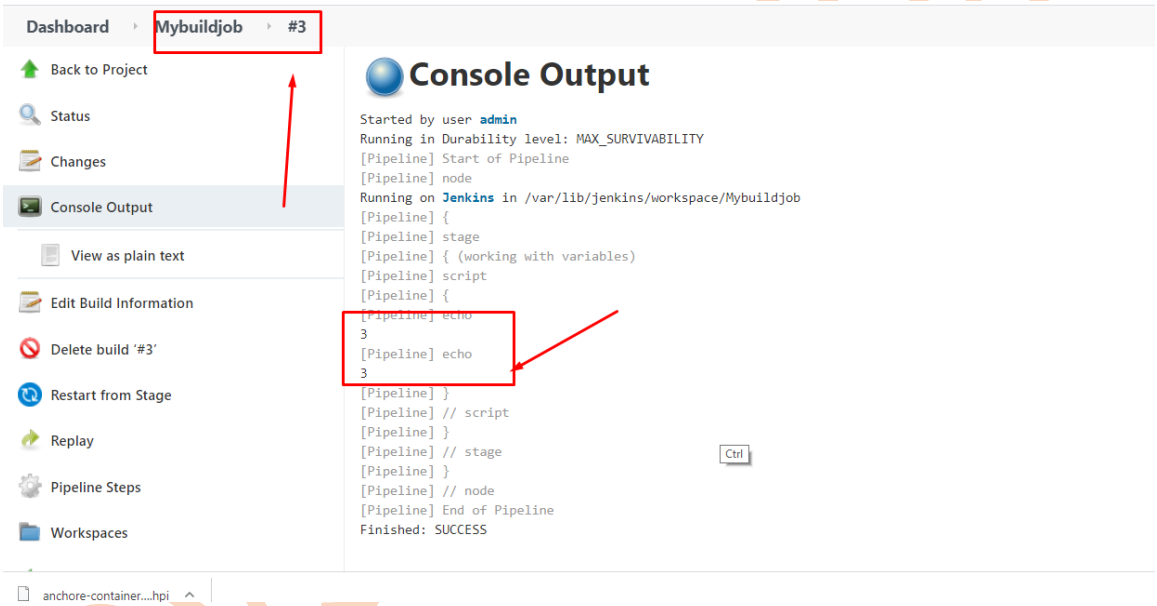
Global Variables:






```
pipeline {
  agent any
  stages {
    stage('working with variables') {
      steps {
        script {
          println "${env.BUILD_NUMBER}"
          println "${env.BUILD_ID}"
        }
      }
    }
  }
}
```

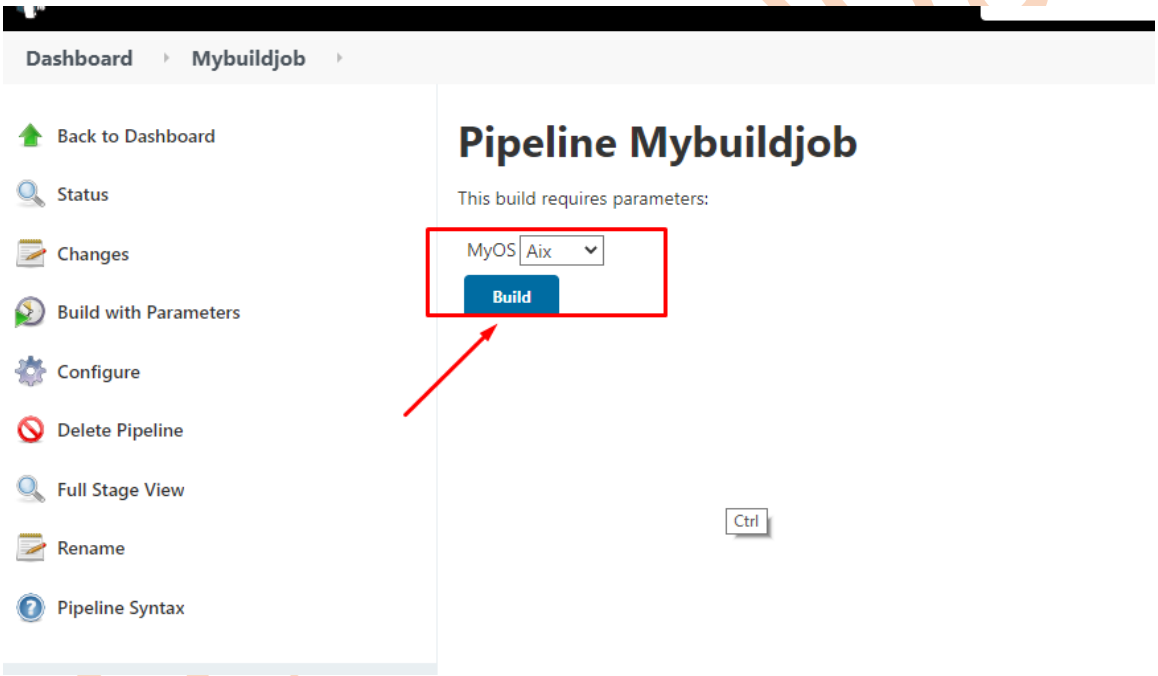


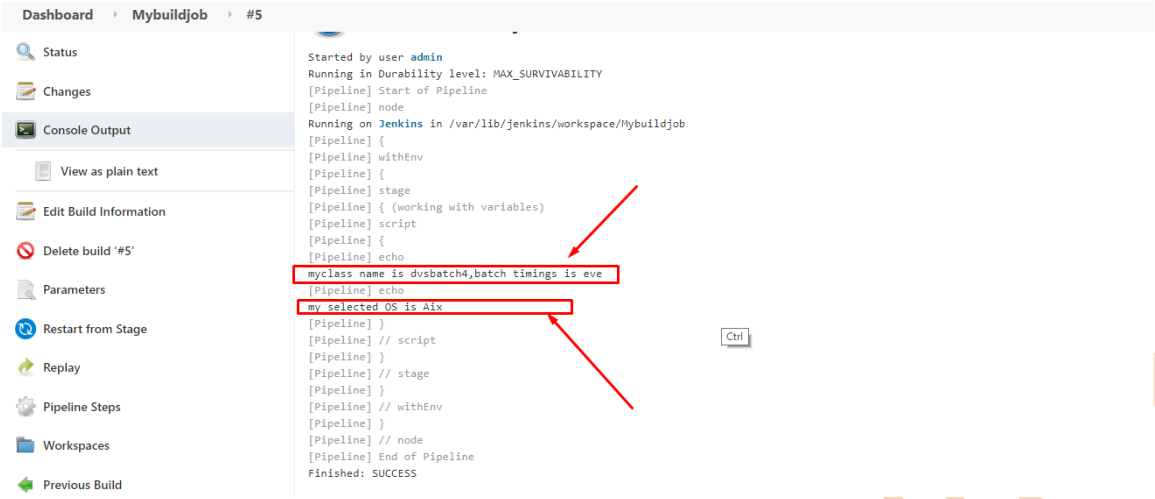


Defining Variables using environment && Defining Variables using parameters:

```
pipeline {
  agent any
  environment {
    myclass = "dvsbatch4"
    timings = "eve"
  }
  parameters {
  choice choices: ['Linux', 'Aix', 'Hp-Ux'], description: '', name: 'MyOS'
```

```
}
 stages {
  stage('working with variables') {
   steps {
    script {
     // My environment variables are
     println "myclass name is ${env.myclass},batch timings is ${env.timings}"
     /*  */
     println "my selected OS is ${params.MyOS}"
    }
   }
  }
 }
}
```





5. Using Decision Making

Jenkins supports multiple operators like

arithmetic operators:
+,-,/,%,*

unary operators:
+,-

Assignment arithmetic operators:
+=,-=,*=,%=,/=

Relational operators:
==,!=,<,<=,>,>=

Logical operators:

&&,||,!

Bitwise operators:
&,|,^,~

Conditional operators:
not operator - !

Example:

```
pipeline {
  agent any
  stages {
    stage('working with operators') {
      steps {
        script {
            def x=10
            def y=20
            println "add of x & y is "+ (x+y)
            println "sub of x & y is "+ (x-y)
            println "div of x & y is "+ (x/y)
            println "mod of x & y is "+ (x%y)
            println "mul of x & y is "+ (x*y)

        }
      }
    }
  }
}
```

Using Decision Making:

Example1:

If-else:

```
pipeline {
  agent any
  stages {
    stage('working with conditions') {
      steps {
        script {
          a=10
          b=20
          if ( "${a}" > "${b}" ){
            println "$a  is big"
          }
          else {
            println "$b is big"
          }
```

```
      }
    }
  }
 }
}
```

Example2:

```
pipeline {
  agent any
  stages {
    stage('working with if conditions') {
      steps {
        script {
          def myval1 = input message: 'enter myval1 value', parameters: [string(defaultValue:
'20', description: '', name: 'myval1', trim: false)]
          if(myval1.toInteger() == 10) {
            println "myval1 value is $mval1"
          }
          else if (myval1.toInteger() == 20)
          {
             println "myval1 value is $myval1"
          }
          else {
             println "$myval1 not satisfies if condition"
          }
        }
      }
    }
  }
}
```

Switch:

```
pipeline {
  agent any
  stages {
    stage('working with switch statement') {
      steps {
      script {
        def myval1 = input message: 'enter val1 value', parameters: [string(defaultValue: '20',
description: '', name: 'val1', trim: false)]
        switch(myval1.toInteger()) {
          case 1:
```

```
        println "I am 1"
        break
      case 2:
        println "I am 2"
                break
      default:
        println "Print either 1 or 2"
    }
  }
  }
 }
}
```

6. Using Loops

while loop:

```
pipeline {
  agent any
  stages {
    stage('working with switch statemetn') {
      steps {
      script {
        def myval = 10
        while(myval > 0)
        {
          println myval
          myval = myval - 1
        }
        }
      }
      }
    }
  }
```

for loop:

```
pipeline {
  agent any
  stages {
```

```
  stage('working with switch statemetn') {
    steps {
    script {
      def myval = 10
      for(i=0;i<=5;i++){
          println "${i}"
      }
      }
    }
    }
  }
}
```

for-in loop:

```
pipeline {
  agent any
  stages {
    stage('working with switch statemetn') {
      steps {
      script {
        def myval = 10
                // for(i in [1,2,3,4,5]) {}
        for(i in 1..5){
            print i
        }
        }
      }
      }
    }
  }
```

break statement:

```
pipeline {
  agent any
  stages {
    stage('working with switch statemetn') {
      steps {
      script {
        def myval = 10
        for(i in [1,2,3,4]){
```

```
            if (i == 3){
                break
            }
            println i
        }
        }
      }
      }
    }
  }
```

Continue Statement:

```
pipeline {
  agent any
  stages {
    stage('working with switch statemetn') {
      steps {
      script {
        def myval = 10
        for(i in [1,2,3,4]){

            if (i == 3){
                continue
            }
            println i
        }
        }
      }
      }
    }
  }
```

7. Using Methods

simple method:

```
def callingFn(){
  println "hello all welcome to dvs devops"
}
pipeline {
  agent any
  stages {
    stage('working with functions') {
    steps {
      script {
        callingFn()
      }
      }
    }
  }
}
```

method with arguments/parameters:

```
def sum(int a,int b){
    output=a+b
    println "add of a & b is "+output
}
pipeline {
  agent any
  stages {
    stage('working with functions') {
    steps {
      script {
        sum(20,30)
      }
      }
    }
  }
}
```

method with default parameters/arguments:

```
def sum(int a=10,int b)
{
  result = a+b
  println "add of $a and $b is "+result
}

pipeline {
  agent any
  stages {
    stage('working with fn with default values') {
      steps {
        script {
          sum(20,30)
        }
      }
    }
  }
}
```

method with return values:

```
def sum(int a=10,int b)
{
  result = a+b
  return (result)
}

pipeline {
  agent any
  stages {
    stage('working with fn with default values') {
      steps {
        script {
          println sum(20,30)
        }
      }
    }
  }
}
```

8 Using I/O operations on files

Reading files:

```
pipeline {
  agent any
  stages {
    stage('working with files') {
      steps {
        script {
          File file = new File("/tmp/test.txt")
          def lines = file.readLines()
          println "Lines\n ${lines}"
          for(line in lines)
          {
            println line
          }
        }
      }
    }
  }
}
```

write a new file:

```
pipeline {
  agent any
  stages {
    stage('working with file IO') {
      steps {
        script {
          File myfile = new File("/tmp/newfile.txt")
          myfile.write("Hello Shahan")
          println "content is ${myfile.text}"
        }
      }
    }
```

```
  }
}
```

Note: If already file exists then the content will be lost.

append the text to a file:

```
pipeline {
  agent any
  stages {
    stage('working with IO operations') {
      steps {
        script {
          File myfile = new File("/tmp/test.txt")
          myfile.append("Lets test the file append")
          println "Printing the content post changes "
          println "file content is \n ${myfile.text}"
        }
      }
    }
  }
}
```

Other fn()'s related to the file operations:

- isFile()
- isDirectory()
- length()

```
pipeline {
  agent any
  stages {
    stage('working with IO') {
      steps {
        script {
            File myfile = new File("/tmp/test.txt")
            println "Is file ${myfile.isFile()}"
            println "Is directory ${myfile.isDirectory()}"
            println "Size of the file ${myfile.length()}"
        }
      }
    }
```

```
  }
}
```

Creating a Directory:

```
pipeline {
  agent any
  stages {
    stage('working with file operations - delete a file/dir') {
      steps {
      script{
        File obj = new File("/tmp/mydir")
        obj.mkdir()
        sh "ls -ld /tmp/mydir"
      }
    }
  }
}
}
```

Delete a file/directory:

```
pipeline {
  agent any
  stages {
    stage('working with IO') {
      steps {
        script {
          File myfile = new File("/tmp/newfile.txt")
          myfile.delete()
          sh "ls -l /tmp/newfile.txt"
        }

      }
    }
  }
}
```

Copying files:

```
pipeline {
  agent any
  stages {
    stage('working with file operations') {
      steps {
```

```
      script {
        File srcfile = new File("/tmp/test.txt")
        File dstfile = new File("/tmp/dest.txt")
        dstfile << srcfile.text
        println "content of our dstfile is ${dstfile.text}"
      }
    }
  }
 }
}
```

9. CI & CD with docker agent

Note: make sure you are installing dockers in the jenkins server.

```
######
yum install docker -y
systemctl enable docker
systemctl start docker
```

Run jenkins as root user to avoid permission issues

```
sed -i 's/JENKINS_USER="jenkins"/JENKINS_USER="root"/g' /etc/sysconfig/jenkins
grep root /etc/sysconfig/jenkins
systemctl restart jenkins
```

Configure the Pipeline via Git SCM and show them the execution. Note: Don't forget to create the credentials

Explanation of total structure with dockers

```
pipeline {
  agent any/none/label/node/docker/dockerfile
  /*
  rest of the syntax
  */
}
```

Here if you are observing you can use agent part for running your code as part of the execution.
We have ample of options like below, you can choose them as per your requirement.

- agent:

This "agent" should be added to the pipeline code. This tell us where our code should run, which means a server/container where code should run.

It accepts diff parameters some of them are below.

You can use any of the below parameters for passing to the agent like below.

Parameters:

- any
Execute the Pipeline, or stage, on any available agent. For example: agent any

- none
When applied at the top-level of the pipeline block no global agent will be allocated for the
entire Pipeline run and each stage section will need to contain its own agent section. For

example: agent none

- label
Execute the Pipeline, or stage, on an agent available in the Jenkins environment with the
provided label. For example: agent { label 'my-defined-label' }

- node
agent { node { label 'labelName' } } behaves the same as agent { label 'labelName' }, but node allows for additional options (such as customWorkspace).

- docker

helps in dynamically create the contianer from the respective image.

```
agent { docker 'maven:3-alpine' }
```

or

```
agent {
    docker {
        image 'maven:3-alpine'
        label 'my-defined-label'
```

```
            args '-v /tmp:/tmp'
            }
        }
```

- dockerfile

Execute the Pipeline, or stage, with a container built from a Dockerfile contained in the source repository. In order to use this option, the Jenkinsfile must be loaded from either a Multibranch Pipeline, or a "Pipeline from SCM." Conventionally this is the Dockerfile in the root of the source repository: agent { dockerfile true }. If building a Dockerfile in another directory, use the dir option: agent { dockerfile { dir 'someSubDir' } }. You can pass additional arguments to the docker build … command with the additionalBuildArgs option, like agent { dockerfile { additionalBuildArgs '--build-arg foo=bar' } }.

Examples:

Pipeline with single container:

```
pipeline {
  agent {
    docker {
      image 'maven'
    }
  }

  stages {
    stage('working with dockers with PaC') {
      steps {
        script {
          sh "mvn -version"
        }
      }
    }
  }
}
```

Pipeline with multiple containers:

```
pipeline {
```

```
  agent any
  stages {
    stage('working inside maven pod') {
      agent {
        docker {
          image 'maven'
          label 'mymavenpod'
          args '-v /tmp:/tmp'
        }
              }
        steps {
          script{
            echo "Hi I am inside maven pod"
          }
        }


    }
    stage('working inside tomcat') {
      agent {
        docker {
          image 'tomcat:8.0'
          lable 'my tomcat image'
          customWorkspace '/mydir/mydata'
        }
              }
        steps {
          script{
            echo "Hi all i am inside tomcat pod"
          }
        }

    }

  }
}
```

Note: make sure that you are giving {agent any } in the first section otherwise you cannot declare agent in the subsequent stages !

10. CI/CD Design & Implementation

1. Java Web based Application Deployment Process :

In this project we need to have one image with java,maven,tomcat installed and configured. Hence we are creating our own customised image using below docker file.

Base Image:

```
[root@jenkins myweb]# cat Dockerfile
FROM centos:7
# Installing Java
ENV JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
ENV PATH=$PATH:$JAVA_HOME
RUN yum install java-1.8.0-openjdk-devel wget git -y
EXPOSE 8080

# Installing Maven
ENV Mvn_Version=3.6.3
ENV M2_HOME=/usr/local/apache-maven/apache-maven-${Mvn_Version}
ENV M2="${M2_HOME}/bin"
ENV PATH=$PATH:$M2

RUN wget https://downloads.apache.org/maven/maven-3/${Mvn_Version}/binaries/apache-maven-${Mvn_Version}-bin.tar.gz && \
   tar xvfz apache-maven-${Mvn_Version}-bin.tar.gz && \
   mkdir /usr/local/apache-maven/apache-maven-${Mvn_Version} -p && \
   mv apache-maven-${Mvn_Version}/* /usr/local/apache-maven/apache-maven-${Mvn_Version}/
```

```
# Installing and configuring Tomcat

ENV Tomcat_Version=8.5.59
RUN     wget http://www-eu.apache.org/dist/tomcat/tomcat-8/v${Tomcat_Version}/bin/apache-tomcat-${Tomcat_Version}.tar.gz && \
        tar xvfz apache-tomcat-${Tomcat_Version}.tar.gz && \
        mkdir -p /opt/tomcat/ /opt/myapplication/ -p && \
        mv apache-tomcat-${Tomcat_Version}.tar.gz /tmp/ && \
        mv apache-tomcat-${Tomcat_Version}/* /opt/tomcat/.
COPY context.xml /opt/tomcat/webapps/manager/META-INF/
COPY tomcat-users.xml /opt/tomcat/conf/
CMD ["/opt/tomcat/bin/catalina.sh", "run"]
```

Context file data context.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at
  http://www.apache.org/licenses/LICENSE-2.0
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<Context antiResourceLocking="false" privileged="true" >
  <Manager
sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

Tomcat users file data tomcat-users.xml:

```
<tomcat-users>
<!--
```

```
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
-->

      <role rolename="manager-gui"/>
      <user username="dvsdevops" password="dvsdevops" roles="manager-gui"/>

</tomcat-users>
```

CI Implementation:

```
pipeline {
  agent any
  stages {
    stage('Git checkout') {
      steps {
        checkout([$class: 'GitSCM', branches: [[name: '*/master']],
doGenerateSubmoduleConfigurations: false, extensions: [[$class: 'CheckoutOption']],
submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'gitaccess', url:
'https://github.com/shan5a6/myweb.git']]])
      }
    }
    stage('Building the image') {
      steps {
        sh """
        docker build -t "javawebapplication" .
        """
      }
    }
  }
}
```

CD Implementation:

```
pipeline {
  agent any
```

```
  stages {
    stage('Checking the application status') {
      steps {
        script {
          def container_status = sh(script: "docker ps -a|grep -i javawebapplication",
returnStatus: true) == 0
          if ("${container_status}" ==  "true")
          {
            sh 'docker rm -f javawebapplication'
          }
        }
      }
    }
    stage('Starting the application') {
      steps {
        sh "docker run -d --name javawebapplication -p 8000:8080 javawebapplication"
      }
    }
  }
}
```

2. Nodejs Application Deployment process:

Explain the Application Deployment manually:
https://github.com/shan5a6/nodeJsApplication.git

CI Implementation:

```
pipeline {
  agent any
  stages {
    stage('Git checkout') {
      steps {
        checkout([$class: 'GitSCM', branches: [[name: '*/master']],
doGenerateSubmoduleConfigurations: false, extensions: [[$class: 'CheckoutOption']],
submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'gitaccess', url:
'https://github.com/shan5a6/nodeJsApplication.git']]])
      }
    }
    stage('Building the image') {
      steps {
        sh """
```

```
      docker build -t "nodejsapplication" .
      """
    }
  }
 }
}
```

CD Implementation :

```
pipeline {
  agent any
  stages {
    stage('Checking the application status') {
      steps {
        script {
          def container_status = sh(script: "docker ps -a|grep -i nodejsapplication",
returnStatus: true) == 0
          if ("${container_status}" ==  "true")
          {
            sh 'docker rm -f nodejsapplication'
          }
        }
      }
    }
    stage('Starting the application') {
      steps {
        sh "docker run -d --name nodejsapplication -p 8001:3000 nodejsapplication"
      }
    }
  }
}
```

3. Springboot Application Deployment process :

Explain the Application Deployment manually ::
https://github.com/shan5a6/javaSpringBoot.git

CI Implementation :

```
pipeline {
  agent any
  stages {
```

```
    stage('Git checkout') {
      steps {
        checkout([$class: 'GitSCM', branches: [[name: '*/master']],
doGenerateSubmoduleConfigurations: false, extensions: [[$class: 'CheckoutOption']],
submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'gitaccess', url:
'https://github.com/shan5a6/javaSpringBoot.git']]])
      }
    }
    stage('Building the image') {
      steps {
        sh """
        docker build -t "springapplication" .
        """
      }
    }
  }
}
```

CD Implementation :

```
pipeline {
  agent any
  stages {
    stage('Checking the application status') {
      steps {
        script {
          def container_status = sh(script: "docker ps -a|grep -i springapplication",
returnStatus: true) == 0
          if ("${container_status}" == "true")
          {
            sh 'docker rm -f springapplication'
          }
        }
      }
    }
    stage('Starting the application') {
      steps {
        sh "docker run -d --name springapplication -p 8002:8080 springapplication"
      }
    }
  }
}
```

DVS Technologies Aws & Devops