

05- JENKINS- AS PIPELINE



Jenkins is a continuous Integration and Continuous Delivery System. Jenkins ia a goto tool when implementing CICD.

Some Features

1. Simple Installation and Configuration
2. Simple web based administration
3. Flexible build jobs and pipelines
4. Customisation and flexibility using the plugin eco system.
5. Scalability through distributed builds using build agents and executors
6. Excellent Online documentation and active user base.

Jenkins Terminology

1. Master

Central Service from which the administration console is served from . Also responsible for

1. Configuration of build jobs , pipeline and plugins via the web administration console it serves
2. Build job Scheduling
3. Build job dispatching to build agents

2.Agent

Jenkins Build agent is a separate server which receives instructions in the form of build job dispatches from the jenkins master. Each agent may have its own os . One agent may have windows os for building the .net packages.

This defines where the execution should take place. Agents are used for declarative pipelines.

3. Build Project

Defines the build configuration properties required to build a unique piece of software.

Different Project types exist for different Project.

1. pipeline
2. freestyle
3. Multi-config project
4. Fodler
5. github organization
6. MultiBranch Pipeline

4.Label

user defined metadata used to group together different build agents

5.Node

Defines by label a machine which capable of executing a build project . This defines where the execution should take place. Nodes are defined in the Scripted pipeline.

6.Artifacts

Output Files generated as an outcome of a build . An artifact for example could be a jar file or zip File

7.pipelines

Modelling concept that embodies and encapsulates the core build logic and workflow.

8.stage

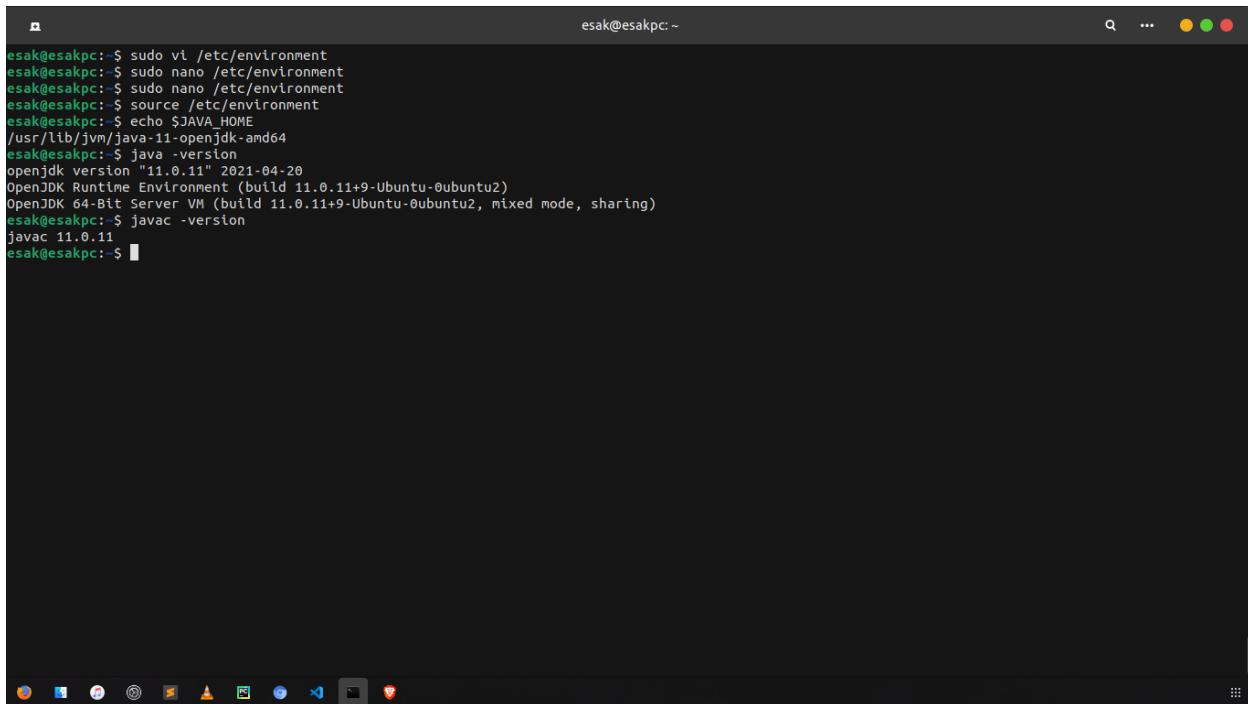
used to segment distinct sub sections of the overall pipeline

An individual stage may consist of multiple pipeline steps

9.Workspace

Dedicated FS directory where build work is performed for the current pipeline or build job. The workspace will be created on a particular node , for which pipeline or build jobs has been configured to run on.

Installation



The screenshot shows a terminal window on a Linux desktop environment. The terminal window title is "esak@esakpc: ~". The window contains the following command history:

```
esak@esakpc: $ sudo vi /etc/environment
esak@esakpc: $ sudo nano /etc/environment
esak@esakpc: $ sudo nano /etc/environment
esak@esakpc: $ source /etc/environment
esak@esakpc: $ echo $JAVA_HOME
/usr/lib/jvm/java-11-openjdk-amd64
esak@esakpc: $ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2, mixed mode, sharing)
esak@esakpc: $ javac -version
javac 11.0.11
esak@esakpc: $
```

The desktop taskbar at the bottom shows several icons, including a browser, file manager, terminal, and system status indicators.

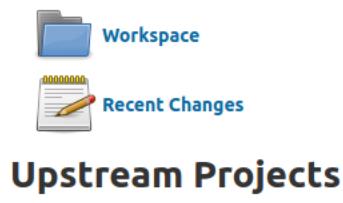
Build Chaining FreeStyle Project

The screenshot shows the Jenkins configuration page for 'BuildJob1'. The 'Post-build Actions' tab is selected. A dropdown menu is open under the 'Command' section, with 'Build other projects' highlighted. Other options in the menu include: Aggregate downstream test results, Archive the artifacts, Publish JUnit test result report, Record Fingerprints of files to track usage, Git Publisher, E-mail Notification, Editable Email Notification, Set GitHub commit status (universal), Set build status on GitHub commit [deprecated], and Delete workspace when build is done. Below the menu is a button labeled 'Add post-build action ▾'. At the bottom of the panel are 'Save' and 'Apply' buttons.

The screenshot shows the 'Build other projects' configuration dialog. It lists 'Projects to build' with 'BuildJob2.' entered into the input field. A red error message says: 'No such project 'Build'. Did you mean 'BuildJob1''? There are three radio button options for triggering: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. Below the dialog are 'Save' and 'Apply' buttons.

We trigger the build job 2 when build job1 is successfully triggered

If we check the BuildJob2 it has the upstream Project



Upstream Projects

BuildJob1

Permalinks

- Last build (#3), 12 sec ago
- Last stable build (#3), 12 sec ago
- Last successful build (#3), 12 sec ago
- Last failed build (#1), 5 min 57 sec ago
- Last unsuccessful build (#1), 5 min 57 sec ago
- Last completed build (#3), 12 sec ago

Build Triggers based on Github

we want to trigger the build when there is a change happened in the github

Step 2: Add github webhooks

Jenkins Pipeline

Jenkins Pipelines

The screenshot shows the Jenkins Pipeline editor interface. On the left, a pipeline graph for a job named 'deploy-venvapp2' is displayed, showing stages like 'Clone', 'Build', and 'Publish'. On the right, a 'Choose step type' dialog is open, listing various Jenkins steps such as 'Shell Script', 'Print Message', 'Retry the body up to N times', 'Sleep', 'Windows Batch Script', 'Archive the artifacts', 'Allocate node', and 'Allocate workspace'. A large watermark at the bottom of the slide reads 'Productivity. developers can maintain their focus'.

Declarative Pipeline

Declarative Pipelines - Schema

```
pipeline
    agent
    --environment
    --libraries
    --options
    --parameters
    --tools
    stages
        **stage
            --agent
            --environment
            --tools
            steps
                script
                --post
            --post
```

-- Optional
** Multiple

- Key points of the declarative syntax:
 - Guides the structure of the pipeline
 - Composed of several optional and mandatory sections
 - Some sections embedded in parent sections
 - Always starts with the **pipeline** keyword to indicate that it is a

Declarative Pipelines – Quick Example

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                //build step logic
            }
        }
        stage('Test') {
            steps {
                //test step logic
            }
        }
        stage('Deploy') {
            steps {
                //deploy step logic
            }
        }
    }
}
```

Key points of this example are the syntax provides



```
pipeline{
    agent any
    tools {nodejs "node"}
    # we are telling the jenkins nodejs has to be installed as prerequest. We want jenkins to refer the global Tool Configuration where we insta
    stages{
        stage('cloning'){
            steps{
                git clone "https://***.git"
            }
        }
        stage('Dependencies'){
            steps{
                sh 'npm install'
            }
        }
        stage('Test'){
            steps{
                sh 'npm test'
            }
        }
        stage('Deploy'){
            steps{
                sh 'npm deploy'
            }
        }
    }
}
```

Scripted Pipeline

Lets add Maven in Global Tool Configuration

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), 'My Views', 'Lockable Resources', 'Credentials', and 'New View'. Below this are sections for 'Build Queue' (empty) and 'Build Executor Status' (two idle executors). The main area is titled 'Manage Jenkins' and contains several configuration items with icons: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench icon), 'Reload Configuration from Disk' (refresh icon), 'Manage Plugins' (puzzle piece icon), and 'System Information' (info icon). A watermark for 'cloudacademy/react-webapp' is visible at the bottom.

And then we click on the global tool configuration item.

Maven will be automatically installed and it uses the name **M3**

The screenshot shows the Jenkins Global Tool Configuration page for Maven. It lists existing Maven installations and allows adding new ones. A new entry for 'Maven' is being added, with the 'Name' field set to 'M3' and the 'Install automatically' checkbox checked. Other options include 'Install from Apache' (version 3.6.0) and 'Delete Maven'. At the bottom, there are 'Save' and 'Apply' buttons. A watermark for 'cloudacademy/react-webapp' is visible at the bottom.

We ensure that the install automatically is enabled,

@@

Decalrative Pipeline

Lets Build a Gradle Project

Some References

<https://gist.github.com/HarshadRanganathan/97feed7f91b7ae542c994393447f3db4>

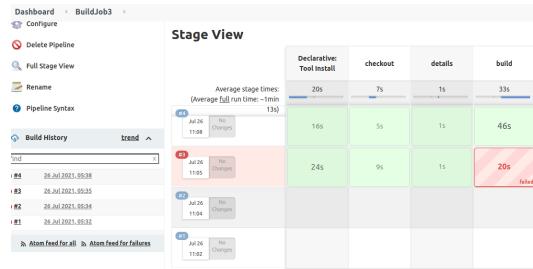
Pipeline: SCM Step

Jenkins - an open source automation server which enables developers around the world to reliably build, test, and deploy their software

 <https://www.jenkins.io/doc/pipeline/steps/workflow-scm-step/>



Jenkins



We are building this Project using Gradle
First Install the Gradle Globally in Jenkins
using the Tool
Second, Write the Declarative Pipeline code as
Below

```
pipeline{
    agent{
        label 'master'
    }
    tools {
        gradle 'gradle-4.10.2'
    }
    environment {
        VERSION = 'jellybean'
    }
    stages{
        stage('checkout'){
            steps{
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: 'master']],
                    extensions: [[$class: 'WipeWorkspace']],
                    userRemoteConfigs: [[url: 'https://github.com/clou']
                ])
            }
        }
        stage('details'){
            steps{
                echo "Running ${env.BUILD_ID} on ${env.JENKINS_URL}"
                echo "${env.VERSION}"
            }
        }
        when{
            environment name: 'VERSION' , value: 'jellybean'
        }
        stage('build'){
            steps{
                sh "gradle build"
            }
        }
    }
}
```

Declarative Pipeline Syntax

First Declarative Pipeline

```
pipeline{  # First Line is pipeline which make it as a declarative pipeline
    agent any # Agent defines on which machine we have to execute the Code
    stages{
        stage("Build"){
            steps{
                echo "First Build Process"
            }
        }
    }
}
```

Scripted Pipeline

```
It always Start with Node
node {
    echo "This is scripted pipeline "
}
```

If we have master slave architecture in Jenkins we can specify the agent name and it will be running in that Slave machine

```
we are going to run in slave1 MAchine

agent {
label 'slave1'
}

WE can rewrite the same as below

agent {
node {
label 'slave1'                      # we specify the slave Machine
customWorkspace "/home/ec2-user/customWorkspace"   # if we want to override the default folder of jenkins
}
}
```

Console Output

```
Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on Slave1 in /home/ec2-user/workspace/pipeline-helloworld
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

if We want to run the jenkins Script

```

pipeline{
    agent any
    stages{
        stage("Build"){
            steps{
                script{                      # Using this Script Tag we can specify/write the groovy scripts
                    def name = "esak"

                    if (name == "esak")
                        print("Hi ${name}")
                    else
                        print("Hi Guest")

                    sleep 2
                    echo "End of Program"
                }
            }
        }
    }
}

```

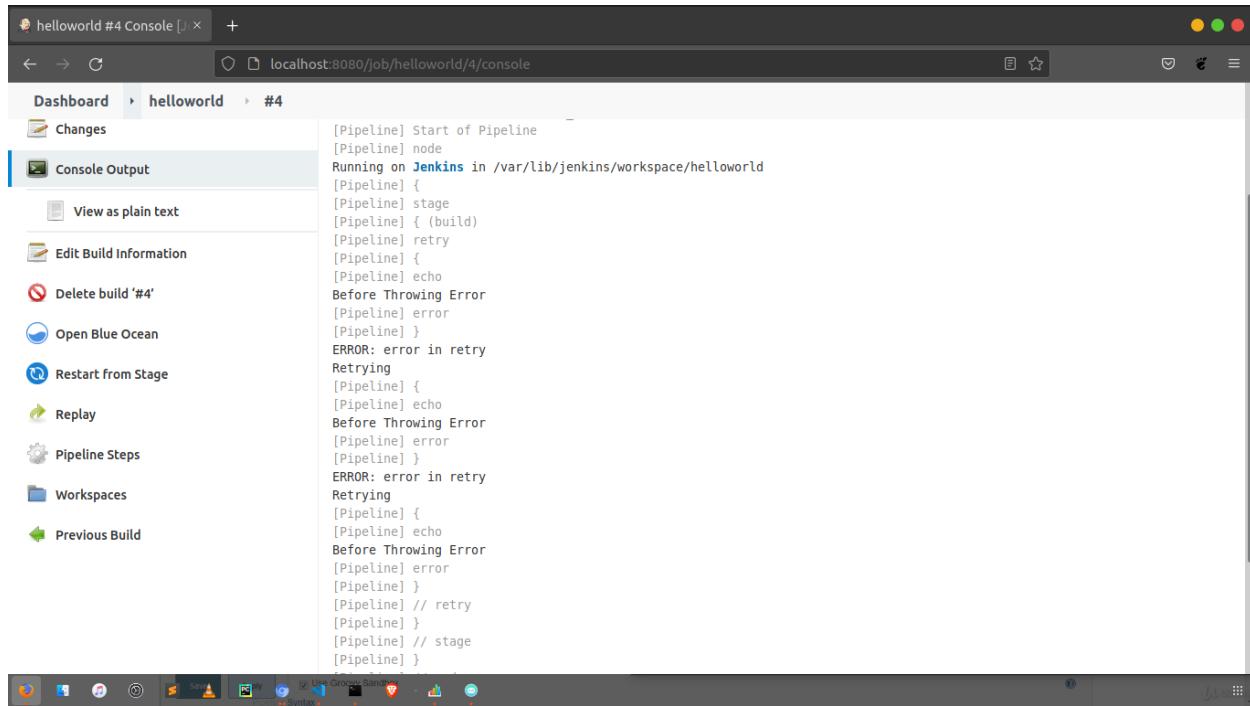
Retry Option

```

we are going to retry a piece of code for 3 times if it throws any error
pipeline{
    agent any
    stages{
        stage('build'){
            steps{
                retry(3){
                    echo "Before Throwing Error"
                    error "error in retry"
                }

                echo "====="
                echo "after retry(3)"  # this wont be printed if we get any error Message
            }
        }
    }
}

```



Timeout

```

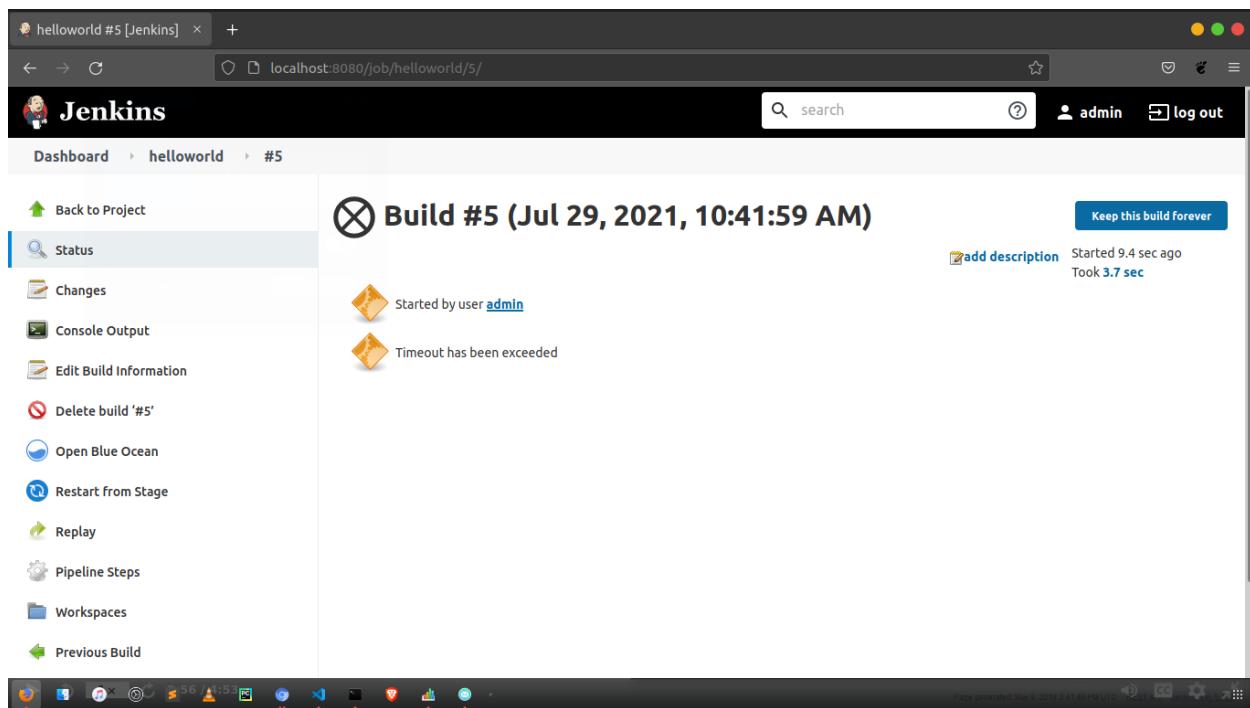
pipeline{
    agent any
    stages{
        stage('build'){
            steps{
                timeout(time: 1 , unit: 'SECONDS'){
                    echo "sleeping in timeout"
                    echo "----"
                    sleep 2
                }
            }
        }
    }
}
```

```

pipeline{
    agent any
    stages{
        stage('build'){
            steps{
                retry(3){
                    timeout(time: 1 , unit: 'SECONDS'){
                        echo "sleeping in timeout"
                        echo "----"
                        sleep 2
                    }
                }
            }
        }
    }
}
```

Check the colour of the Build#5 , it means timeout

as per the code we have provide the timeout option as 1 seconds , but we have waited for 2 seconds causing the build to Failure



Tool is used to refer the already installed global tool configuration tools and made it available in the Jenkins pipeline

```
pipeline{
    agent any
    stages{
        stage('build'){
            tools{
                maven 'M3'      ### 'M3' is the name of the maven installation in the global configuration
            }
            steps{
                sh 'mvn --version'
            }
        }
    }
}
```

```

[Pipeline] Running on Jenkins in /var/lib/jenkins/workspace/helloworld
[Pipeline] {
[Pipeline] stage
[Pipeline] { (build)
[Pipeline] tool
Unpacking https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.6.0/apache-maven-3.6.0-bin.zip to /var/lib/jenkins/tools/hudson.tasks.Maven_MavenInstallation/M3 on Jenkins
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ mvn --version
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-25T00:11:47+05:30)
Maven home: /var/lib/jenkins/tools/hudson.tasks.Maven_MavenInstallation/M3
Java version: 11.0.11, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.11.0-25-generic", arch: "amd64", family: "unix"
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Option and Retry

```

# retry the build job for 3 times if we get any error

pipeline{
    agent none
    stages{
        stage('build'){
            options{
                retry(3)
            }
            steps{
                echo "Before Error Statement"
                error "Error in the build Job"
                echo "After Error Statement"
            }
        }
    }
}

```

Predefined variable List

Global Variable Reference

The docker variable offers convenient access to Docker-related functions from a Pipeline script. Methods needing a Jenkins agent will implicitly run a node {...} block if you have not wrapped them in one. It is a good idea to enclose a block of steps which should all run on the same node in such a block yourself.

 <https://opensource.triology.de/jenkins/pipeline-syntax/globals>

currentBuild.result= 'FAILURE' will make the script to fail

```

pipeline{
    agent none
    stages{
        stage('build'){
            options{
                retry(3)
            }
        }
    }
}

```

```

        steps{
            echo "Before Error Statement"
            script{
                currentBuild.result = 'FAILURE'
            }
            echo "After Error Statement"
        }
    }
}

```

eventhough we specified retry and if timeout exceeds there wont be any more retries it simply exists out

TimeStamp

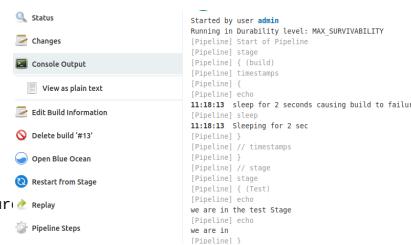
If we want to have timestamp , just like the log use timestamp plugin

```

pipeline{
    agent none
    stages{
        stage('build'){
            options{
                timestamps()
            }
            steps{
                echo "sleep for 2 seconds causing build to failur"
                sleep 2
            }
        }

        stage('Test'){
            steps{
                echo "we are in the test Stage"
                echo "we are in"
            }
        }
    }
}

```



```

pipeline {
    agent none
    environment {
        name1 = "esakki"
        name2 = "sankar"
    }
    stages {
        stage('build') {
            steps{
                echo "name1 ${name1}"
                echo "name2 ${name2}"
                sh "printenv"
            }
        }
    }
}

```

Lets Add Secret

New Credentials [Jenkins] +

localhost:8080/credentials/store/system/domain/_/newCredentials

Jenkins

Dashboard > Credentials > System > Global credentials (unrestricted) >

[Back to credential domains](#)

[Add Credentials](#)

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: esak

Treat username as secret

Password:

ID: esak

Description: esak_with_esak

OK

New Credentials [Jenkins] +

localhost:8080/credentials/store/system/domain/_/newCredentials

Jenkins

Dashboard > Credentials > System > Global credentials (unrestricted) >

[Back to credential domains](#)

[Add Credentials](#)

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: esak_secret_text

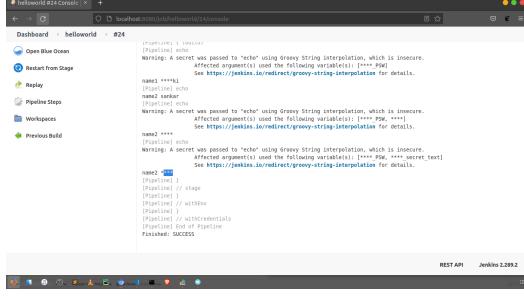
Description: esak_secret_text

OK

REST API Jenkins 2.289.2

```
pipeline {
    agent none
    environment {
        name1 = "esakki"
        name2 = "sankar"
        esak = credentials('esak')
        esak_secret_text = credentials('esak_secret_text')
    }
    stages {
        stage('build') {

```



```

steps{
    echo "name1 ${name1}"
    echo "name2 ${name2}"
    echo "username ${esak_USR}"
    echo "password ${esak_PSW}"
    echo "name2 ${esak_secret_text}"
}

}

}

# esak is a field having username and password.
# To access the username part we have to use esak_USR
# to access the password part user esak_PSW
# USR and PSW will be cpas always

```

Printing the Secrets in Jenkins, it will be in the protected format

When in Jenkins

when

Required: No

At least one of the following conditions must be specified. If there is more than one condition, all conditions must return true for the stage to execute



```

pipeline{
    agent none
    environment {
        DEPLOY_TO = 'PROD'
        DEPLOY_VERSION = '2.1'
    }
    stages {
        stage('build'){
            when {
                // environment name: 'DEPLOY_TO' , value: 'PRODUCTION'
                // equals expected: 'production', actual: DEPLOY_TO
                // not {
                //     equals expected: 'production' , actual: DEPLOY_TO
                // }
                // we can use groovy expression
                // expression {
                //     DEPLOY_TO == 'QA'
                // }
                // allof - when all the Conditions are true stage will be executed
                // allof {
                //     environment name:DEPLOY_TO, value: "PRODUCTION"
                //     environment name:DEPLOY_VERSION, value:"2.1"
                // }
                // anyOf - when any the Conditions are true stage will be executed
                anyOf{
                    environment name:DEPLOY_VERSION, value:'2.1'
                    environment name:DEPLOY_TO, value:'PROD'
                }
            }
            steps{
                echo "Building"
            }
        }
    }
}

```

Checking the Branch Condition

```

pipeline{
    agent none
    environment {

```

```

        DEPLOY_TO = 'PROD'
        DEPLOY_VERSION = '2.1'
    }
    stages {
        stage('build master'){
            when {
                branch 'master'
            }

            steps{
                echo "Building the master"
            }
        }

        stage('build dev'){
            when {
                branch 'dev'
            }
            steps {
                echo "Building the dev"
            }
        }
    }

}

If we want to do a Build process based on the commit message if the commit message has "some_text" , build will be executed

when {
    changelog '.*some_text.*'
}

when {
    changeRequest()
}

```

i have created a PR

#1 MyPR_Update Jenkinsfile #1

esak21 wants to merge 1 commit into `main` from `dev`

Conversation 0 **Commits** 1 **Checks** 0 **Files changed** 1 +1 -10

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one—assign yourself

Labels
None yet

Click the Scan Repository Now, Now PR has been picked up like below

The screenshot shows the Jenkins job helloworld_github dashboard. On the left, there's a sidebar with links like GitHub, Rename, Config Files, Pipeline Syntax, Credentials, and New View. Below the sidebar, there are two sections: 'Build Queue (1)' which lists 'helloworld_github > PR-1' and 'Build Executor Status' which shows 1 idle and 2 idle executors. At the bottom right, it says 'REST API Jenkins 2.289.2'. The browser address bar shows 'localhost:8080/job/helloworld_github/'.

Build has been executed

The screenshot shows the Jenkins job helloworld_github view for 'Pull Requests (1)'. The left sidebar has links for Up, Status, Configure, Scan Repository Now, Scan Repository Log, Multibranch Pipeline Events, People, Build History, Project Relationship, Check File Fingerprint, Open Blue Ocean, and GitHub. The main area shows a table for 'Pull Requests (1)'. The table has columns: S, W, Name, Last Success, Last Failure, Last Duration, Fav. There is one row for 'PR-1' with icons for status, pipeline, and favoriting. Below the table, there are links for 'Icon: S M L', 'Legend', 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. The browser address bar shows 'localhost:8080/job/helloworld_github/view/change-requests/'. The page footer indicates it was generated on May 20, 2016, at 7:12:37 AM UTC.

RELEASE-Update Jenkinsfile #2

[Open](#) esak21 wants to merge 1 commit into `main` from `dev`

```
pipeline{
    agent none
    environment {
        DEPLOY_TO = 'PROD'
        DEPLOY_VERSION = '2.1'
    }
    stages {
        stage('build master'){
            when {
                // When PR IS Created
                // changeRequest()
                // When Pr is created with the Title RELEASE-
                changeRequest title: "RELEASE-"
            }
            steps{
                echo "Building the master"
            }
        }
    }
}
```

if the committed files are .js file then kick start the build process

```
chnageset glob: "*.js"
chnageset glob: "*.js", casesensitive: true
```

Run Jobs in parallel

```
pipeline {
    agent none
    stages {
        stage('Staging Running in Parallel ') {
            failFast true // if one of the stage failed , rest of the stage is also failed
            parallel {
                stage('stage1') {
                    steps{
                        echo "stage1 executing"
                        sleep 10
                    }
                }
                stage('stage2'){
                    steps{
                        echo "stage2 executing"
                        sleep 2
                    }
                }
                stage ('stage3') {
                    steps{
                        echo "stage3 executing "
                        sleep 10
                    }
                }
            }
        }
    }
}
```

```
}
```

Input

```
pipeline{
    agent any

    stages{
        stage('Build'){
            input{
                message "Please specify environment:"
                ok "OK"
                submitter "dummyuser,admin@localhost.com"
                submitterParameter "whoIsSubmitter"
                parameters {
                    string(name: 'environment', defaultValue: 'Dev', description: 'Environment to build for (Valid values: Dev, Test, Prod)')
                    string(name: 'version', defaultValue: '1.0', description: 'Version number to build for')
                    booleanParam(name: 'to_deploy_to_environment', defaultValue: true, description: '')
                    choice(choices: 'US-EAST-1\nUS-WEST-2', description: 'What AWS region?', name: 'region')
                    text(name:'myText', defaultValue:'myTextValue', description:'myText')
                    password(name:'myPassword', defaultValue:'myPasswordValue', description:'myDescription')
                    file(name:'myFile', description:'fileDescription')
                    credentials(name:'myCredentials', description:'myCredentialsDesc', required:true)
                }
            }

            steps{
                echo "We are building for ${environment}, ${version}, and we are deploying to environment: ${to_deploy_to_environment}"
                echo "region:${region}, myText: ${myText}, myPassword: ${myPassword}, and myFile: ${myFile}"
                echo "submitter is: ${whoIsSubmitter}"
                echo "selected credentials is: ${myCredentials}"
            }
        }
    }
}
```

The screenshot shows two browser tabs side-by-side. The left tab is titled 'Paused for Input: helloworld' and displays the Jenkins Pipeline Syntax for build #46. It shows the 'Please specify environment' step with fields for environment (set to 'Dev'), version (set to '1.0'), and region (set to 'US-EAST-1'). The right tab is titled 'helloworld #46 Console' and shows the Jenkins Console Output for the same build. The output logs show the pipeline starting, reading the environment variables, and echoing them back.

```
pipeline{
    agent any

    stages{
        stage('build'){
            tools{
                maven 'M3'
            }
            steps{
                echo "in the Build satge"
                sh 'mvn --version'
            }
        }
    }
    post{
```

The screenshot shows two Jenkins windows side-by-side. On the left, the 'Pipeline Syntax' window displays a Groovy script for a pipeline job named 'helloworld'. The script includes sections for success, failure, and always triggers, each containing an echo command. On the right, the 'helloworld #47 Console' window shows the build log output, which matches the pipeline script. The log ends with 'Finished: SUCCESS'.

```

success{
    echo "post=Success is called"
}
failure{
    echo "post-> Failure is called"
}
always{
    echo "post-always is called so many times"
}
}

```

OPTIONS at Pipeline Level

If we disable the concurrent Build , then it will wait for the build to Finish

The screenshot shows the Jenkins interface with the 'Stage View' and 'Build History' sections. The Stage View on the right displays four stages for build #57, showing their execution times and status. The Build History on the left lists recent builds, with build #57 currently active. A sidebar on the left provides options like Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, and Pipeline Syntax.

build	Declarative Post Action
3s	521ms

Average stage times:
(Average full run time: ~3s)

```

pipeline{
    agent any
    options{
        //buildDiscarder(logRotator(numToKeepStr: '1')) // it will maintain only one build run
        // all the Build run will be deleted from the jenkins server
        // We can disable same job is active in 2 instances
        //disableConcurrentBuilds()

        // It will avoid every commit we push , it avoid the build behaviour
        //overrideIndexTriggers(true)
        // If the Build is unstable we will skip the result of the stages which were dependant
        //skipStagesAfterUnstable()
        //
        //checkoutToSubdirectory("mypackages")
        // when using docker , each stage will use a new docker container
        newContainerPerStage()
    }
}

```

```

stages{
    stage('build'){
        tools {
            maven 'M3'
        }
        steps{
            sh "mvn --version"
            sleep 20
        }
    }
}

```

Parameters at pipeline level same as the stage

First time it run with the default parameters ,

The screenshot shows the Jenkins Pipeline helloworld Stage View. On the left, there is a sidebar with various options: Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, and Rename. The main area is titled "Stage View" and displays a single build step named "build". Below the step, it says "Average stage times: (Average full run time: ~12s)". A progress bar indicates the time is 7s. At the bottom, there is a button labeled "#60 Jul 29 No".

second time it will ask for the values

The screenshot shows the Jenkins Pipeline helloworld configuration page. On the left, there is a sidebar with various options: Back to Dashboard, Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, and Rename. The main area is titled "Pipeline helloworld" and shows the pipeline configuration. It defines a parameter "env" with a value "DEV" and a credential "mycred" with a secret text "esak_secret_text". A "Build" button is present. To the right, the pipeline code is displayed:

```

pipeline {
    agent any
    parameters {
        string(name: 'env', defaultValue: 'DEV', description: 'ENV')
        credentials(name: 'mycred', description: 'Enter credential')
    }
    stages{
        stage('build'){
            steps{
                echo "building"
            }
        }
    }
}

```

Triggers

```

Cron

pipeline{
    agent none
    triggers {
        cron('0 1 2 3 1-7')
    }
    stages{

```

```

        stage('build'){
            echo "building the Java Project"
        }
    }

pipeline{
    agent any
    triggers {
        //cron('0 1 2 3 1-7')
        pollSCM('* * * * *')
    }
    stages{
        stage('build'){
            steps{
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: "origin/main"]],
                    userRemoteConfigs: [
                        url: 'https://github.com/esak21/MySimpleApplication.git'
                    ]
                ])
                echo "building the Java Project"
            }
        }
    }
}

If we want to trigger the Job based on the Previous Job ( comma will act as OR Condition Here )
pipeline{
    agent any
    triggers {
        //cron('0 1 2 3 1-7')
        //pollSCM('* * * * *')
        upstream(upstreamProjects: 'pipeline-triggers-upstream-job1,pipeline-triggers-upstream-job2',
            threshold: hudson.model.Result.SUCCESS)//UNSTABLE, FAILURE, NOT_BUILT, ABORTED
    }
    stages{
        stage('build'){
            steps{
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: "origin/main"]],
                    userRemoteConfigs: [
                        url: 'https://github.com/esak21/MySimpleApplication.git'
                    ]
                ])
                echo "building the Java Project"
            }
        }
    }
}

```

Jenkins With Docker Declarative Pipeline

```

pipeline {
    agent {
        docker{
            image 'maven:3.8.1-jdk-11-slim'
            args '-e someEnv=dev'
            alwaysPull false
            // If jenkins has write Permission it will create the Folder in the server and copy the artifacts to it
            customWorkspace '/var/lib/jenkins/workspace/MyCustomSpace'
            // if we have any slaves to run we can use it like
            //label 'slave1'
        }
    }
    stages{
        stage('Build'){
            steps{
                sh 'mvn -version'
                sh 'echo ${someEnv}'
            }
        }
    }
}

```

DockerFile

```
DockerFile has to be in the Root Directory

pipeline{
    agent any
    stages {
        stage('checkout'){
            steps{
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: 'origin/main']],
                    userRemoteConfigs:[[
                        url: 'https://github.com/esak21/MySimpleApplication.git'
                    ]]
                ])
            }
        }
        stage("build"){
            steps{
                sh 'cat /etc/lsb-release'
                echo 'Build Completed'
            }
        }
    }
}

If the dockerFile is not available in the root of the project directory we have to specify it manually

inside the stages we have to specify it
stage('dockerFile'){
    steps{
        dockerfile{
            dir 'prod/docker'
            If we specify the file with different name then
            filename 'mycustomDockerFile'
        }
    }
}
```

We can run groovy code in the declarative pipeline using script tag

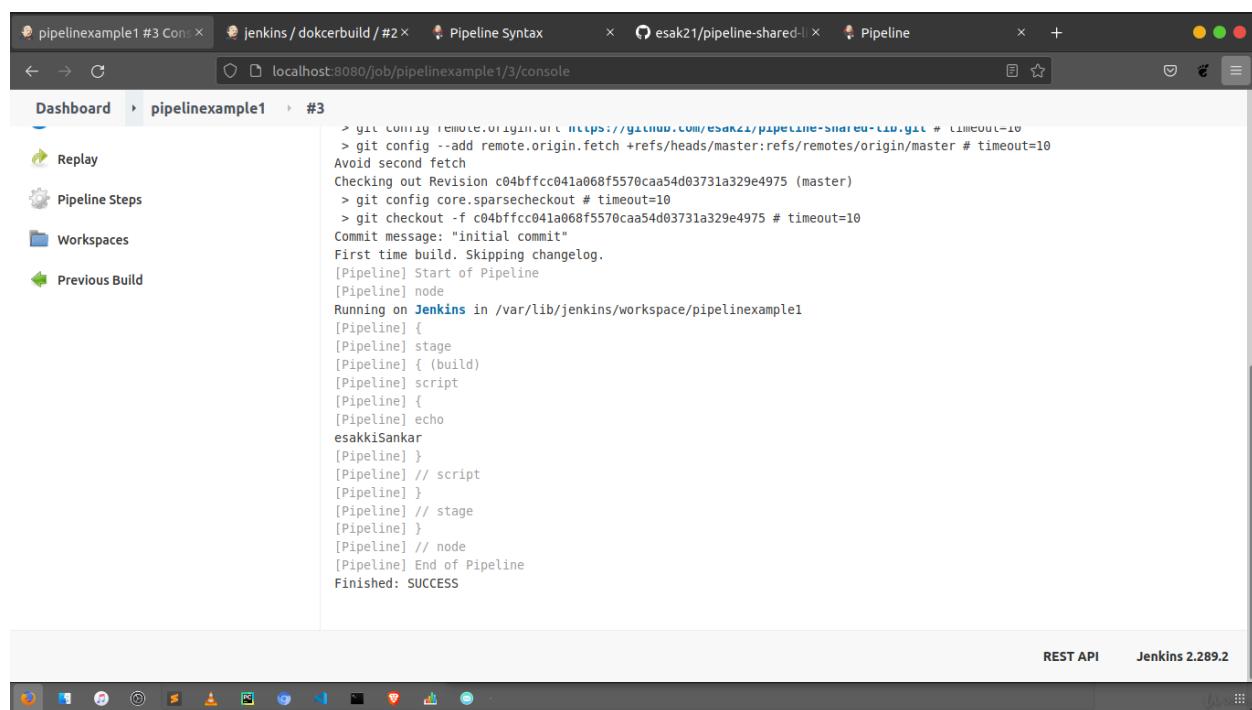
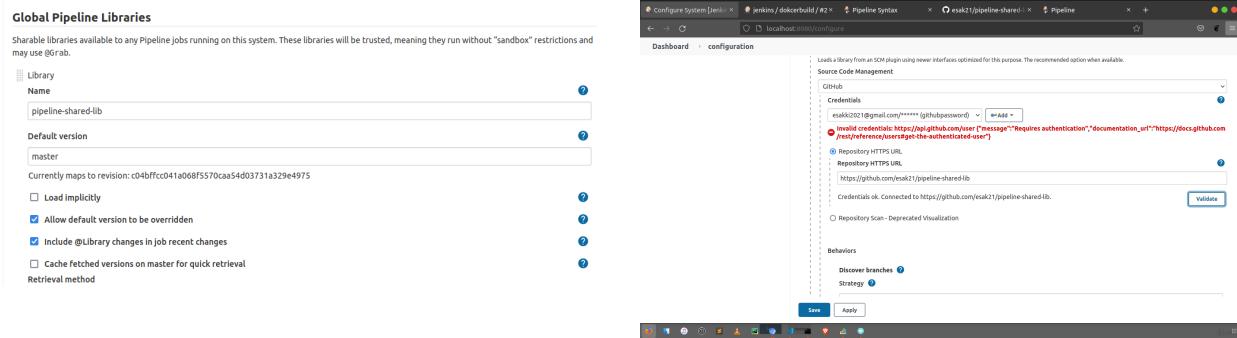
```
def nameOfPerson = "john"

def printName(name){
    echo name
}

pipeline{
    agent any
    stages{
        stage('build'){
            steps{
                script{
                    printName(nameOfPerson)
                }
            }
        }
    }
}
```

Adding common Pipeline groovy code as lib

pipeline-shared-lib



```

@Library('pipeline-shared-lib') _      // we are importing all the functions

pipeline {
    agent any
    stages {
        stage('build') {
            steps {
                script {
                    swissknife.printName "esakkisankar"
                }
            }
        }
    }
}
Above code uses the master Branch , if incase you want to use other branches like dev the
@Library('pipeline-shared-lib@dev') _

@Library(['pipeline-shared-lib@dev', 'pipeline-shared-lib-code']) _

```

Scripted Pipeline

```

### Scripted pipeline always start with "node"

node {
    echo "This is scripted pipeline "
}

node {
    stage('build'){
        // we can run this echo Step in any node
        node {
            echo "This is scripted pipeline "
        }
    }
}

#####
Without node block also we can write the code

stage{
    echo "Stage1"
}

stage {
    echo "stage 2"
}

#####
# Using Docker Image with the scripted pipeline
# uncheck the Groovy sandbox
# https://stackoverflow.com/questions/38276341/jenkins-ci-pipeline-scripts-not-permitted-to-use-method-groovy-lang-groovyobject

node {
    stage('build') {
        def mavenImage = docker.image('maven:3.8.1-jdk-11-slim')
        mavenImage.inside{
            sh 'mvn --version'
        }
    }
}

node {
    stage('build') {
        def mavenImage = docker.image('maven:3.8-jdk-8')
        // To pull the Image
        mavenImage.pull()

        mavenImage.inside(
            '-e someENV=dev',
            { sh 'echo $someENV'}
        )
    }
}

#####
To USE Docker File
node {
    stage('Build'){
        checkout([$class: 'GitSCM',
            branches: [[name: "origin/master"]],
            userRemoteConfigs: [
                url: 'https://github.com/pipelineascodecourse/pipeline-agent-dockerfile.git']
        ])
    }

    def myCustomUbuntuImage = docker.build("my-ubuntu:my-latest")

    def myCustomUbuntuImage = docker.build("my-ubuntu:my-latest", "--file mycustomdockerfile .")

    myCustomUbuntuImage.inside {
        sh 'cat /etc/lsb-release'
    }
}

```

```

}

node {
    stage('Build'){
        checkout([$class: 'GitSCM',
            branches: [[name: "origin/master"]],
            userRemoteConfigs: [[
                url: 'https://github.com/pipelineascodecourse/pipeline-agent-dockerfile.git']]
        ])

        def myCustomUbuntuImage = docker.build("my-ubuntu:my-latest","--tag mydockerfile:example .")

        myCustomUbuntuImage.inside {
            sh 'cat /etc/lsb-release'
        }
    }
}

## Getting SHeLL values and exit status
node{
    stage('build'){
        def shelloutput = sh(
            script: "ls -la",
            returnStdout: true
        ).trim()

        echo "Shell Output is ${shelloutput}"

        def shellstatus = sh(
            script: "exit 1",
            returnStatus: true
        )

        echo "Shell outout is ${shellstatus}"
    }
}

# Lets REtry the Build
node {
    stage("build"){
        retry(3) {
            error "Manual Error thrown by me"
        }
    }
}

### TImeOut
# ##### Pipeline will be aborted after the timeout was exceeded
node {
    stage("build"){
        timeout(time: 1, unit:'SECONDS') {
            sleep 5
        }
    }
}

#####
node {
    timestamps{

        stage("build"){
            echo "we have timestamp"
        }
        stage("deploy"){
            echo "we have timestamp"
        }
    }
}

#####
# Credentials
node {
    withCredentials( [ usernamePassword(
        credentialsId: "artifactory",
        usernameVariable: 'USER_NAME',
        passwordVariable: 'PASSWD'
    )])
}

```

```

        )] ){
            echo "USER is ${USER_NAME}"
            echo "PASSWORD is ${PASSWD}"
        }
    }

#####
If we want to implement when we have to use the if condition

node {
    def name ="mycoolday"
    def isGroovyCool = false

    withEnv(['DEPLOY_TO=production']){
        stage('build'){

            //when env condition
            if(env.DEPLOY_TO == 'production')
                println "name is ${name}"

            if (isGroovyCool == false)
                println "groovy is always Cool"

            if ( name == "mucoolday" && !isGroovyCool)
                println "name is ${name} and groovy is Cool"
        }
    }

    node {
        if (env.BRANCH_NAME == "dev"){
            stage('build'){
                echo "building the Dev stage"
            }
        }
    }
}

## When PR is Ready

node{
    stage('Build'){
        println "env.CHANGE_ID: ${env.CHANGE_ID}"
        println "env.CHANGE_URL: ${env.CHANGE_URL}"
        println "env.CHANGE_TITLE: ${env.CHANGE_TITLE}"
        println "env.CHANGE_AUTHOR: ${env.CHANGE_AUTHOR}"
        println "env.CHANGE_AUTHOR_DISPLAY_NAME: ${env.CHANGE_AUTHOR_DISPLAY_NAME}"
        println "env.CHANGE_AUTHOR_EMAIL: ${env.CHANGE_AUTHOR_EMAIL}"
        println "env.CHANGE_TARGET: ${env.CHANGE_TARGET}"

        if(env.CHANGE_TITLE == "when-pr"){
            echo "pull request is found"
        } else {
            echo "pull request is not found"
        }
    }
}

## Running Test IN Parallel
# 3 jobs will run in parallel and when its completed next stage will be executed
# if any of them is failed pipeline will be failed

node {
    stage('build in parallel'){
        parallel 'parallel1':{
            echo "Pip 1 is running "
            sleep 10
        },
        "par2":{
            echo "pip2 is running"
            sleep 5
        },
        "par3": {
            echo "par3 is running"
            sleep 10
        },
        failFast:true
    }
    stage("deploy"){

}

```

```

        echo "deploying"
    }
}

# Getting Input

node {
    def userInput = ""
    stage("build"){
        userInput = input ( id: 'userInput',
            message: 'please specify the environment',
            submitterParameter: "wholeSubmitter",
            parameters:[
                string(name:"environment", defaultValue:"dev", description:"Server to be deployed"),
                string(name:"version", defaultValue: "1.0"),
                choice(choices:"US_EAST_1\\nUS_WEST_2", description :"aws region", name:"region"),
                credentials(name:"mycdredentials", required:true, description:"AWS CREDENTIALS")
            ])
        echo "we are building for ${userInput.environment}"
        echo "we are building in ${userInput.region}"
        echo "whole code ${userInput.wholeSubmitter}"
    }
}

If the user doesn't provide the values within the specific period, if user don't specify any input within the timeout we get some exception

node{
    try{

        stage("build"){
            echo "building"
            currentBuild.result = "SUCCESS"
        }
    }catch(err){

        }finally{
            def currentResult = currentBuild.result
            echo "always will be executed"
            if ( currentResult == "SUCCESS"){
                println "we are in finally block"
            }
        }
    }

# BuildDiscarder
Only one Build History will be maintained by Jenkins

node {
    properties(
        [
            buildDiscarder(logRotator(numToKeepStr: '1'))
        ]
    )

    stage("build"){
        echo "Hello World"
    }
}

# Disable Concurrent Builds

node {
    properties(
        [
            buildDiscarder(logRotator(numToKeepStr: '10')),
            disableConcurrentBuilds()
        ]
    )

    stage("build"){
        sleep(time:10, unit:'SECONDS')
        echo "Hello World"
    }
}
## Download workspace into a custom Folder
node {

```

```

properties(
    [
        buildDiscarder(logRotator(numToKeepStr: '10')),
        disableConcurrentBuilds()
    ]
)
stage("build"){
    ws("/var/lib/jenkins/workspace/scp1/myfolder") {

        checkout([
            $class: 'GitSCM',
            branches: [[name: "origin/main"]],
            userRemoteConfigs:[[
                url: 'https://github.com/esak21/IACS_BRAINIACS.git'
            ]]
        ])

        sleep(time:10, unit:'SECONDS')
        echo "Hello World"
    }
}

## PULLING DOCKER IMAGE FOR EACH STAGE
node{
    stage('build'){

        def mavenImage = docker.image("maven:3.5.3-jdk-10-slim")
        mavenImage.inside{
            sh "mvn -v"
        }

    }
    stage('deploy'){
        def mavenImage = docker.image("maven:3.5.3-jdk-10-slim")
        mavenImage.inside{
            sh "mvn -v"
        }
    }
}

#####
# imp Parametres #####
node() {
    // adds job parameters within jenkinsfile
    properties([
        parameters([
            string(
                defaultValue: 'Dev',
                description: 'Environment to build for (Valid values: Dev, Test, Prod)',
                name: 'environment'
            ),
            booleanParam(
                defaultValue: true,
                description: '',
                name: 'to_deploy_to_environment'
            ),
            choice(
                choices: 'US-EAST-1\\nUS-WEST-2',
                description: 'What AWS region?',
                name: 'region'
            ),
            text(
                name:'myText',
                defaultValue:'myTextValue',
                description:'myText'
            ),
            password(
                name:'myPassword',
                defaultValue:'myPasswordValue',
                description:'myDescription'
            ),
            file(
                name:'myFile',
                description:'fileDescription'
            ),
            credentials(
                name:'myCredentials',
                description:'myCredentailsDesc',
                type:'usernamePassword'
            )
        ])
    ])
}

```

```

        required:true)
    ])
}

echo "We are building for ${params.environment}, and we are deploying to environment: ${params.to_deploy_to_environment}"
echo "region:${params.region}, myText: ${params.myText}, myPassword: ${params.myPassword}, and myFile: ${params.myFile}"
echo "selected credentials is: ${params.myCredentials}"
}

node {
    def mavenHome = tool name:'M3' , type: 'maven'
    sh "${mavenHome}/bin/mvn -v "
}

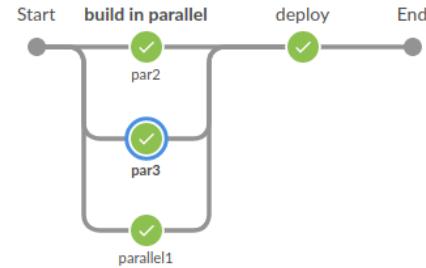
#####
# Triggers
node {
    properties([
        pipelineTriggers([
            cron('0 1 2 3 1-7')
        ])
    ])

    stage('build'){
        echo "helloworld"
    }
}

#####
# Git polling log will poll the git for every minute
# pollSCM every 1 minute s
node{
    properties([
        pipelineTriggers([
            pollSCM('* * * * *')
        ]),
    ])
    stage('checkout'){
        checkout([
            $class: 'GitSCM',
            branches:[name: "origin/main"],
            userRemoteConfigs:[
                url: "https://github.com/esak21/IACS_BRAINIACS.git"
            ]
        ])
    }
}
}

```

✓ scp1 < 24		Pipeline	Changes	Tests
Branch: -	⌚ 14s	No changes		
Commit: -	⌚ 3 minutes ago	Started by user admin		



Integrate Jenkins With SonarQube, Artifactory ,

Issues Faced in Sonar Qube

change the Vm memory to 2 GB

```
sysctl -w vm.max_map_count=262144
systemctl restart docker
```

Run the Below Docker Compose File

```
### JENKINS FILE

node {
    def GRADLE_HOME = tool name:gradle-4.10.2
    def SONARQUBE_HOSTNAME= 'sonarqube'
    stage('checkout'){
        git url: 'https://github.com/cloudacademy/devops-webapp.git'
    }

    stage('build'){
        sh "${GRADLE_HOME}/bin/gradle build"
    }

    stage('sonar-scanner'){
        def sonarqubescannerHome = tool name: 'sonar' , type: 'hudson.plugins.sonar.SonarRunnerInstallation'

        withCredentials([string(credentialId: 'sonar', variable:'sonarLogin')]){
            sh "${sonarqubescannerHome}/bin/sonar-scanner -e -Dsonar.host.url=http://${SONARQUBE_HOSTNAME}:9000 -Dsonar.login=${sonarLogin}"
        }
    }
}
```

Continuous Code Quality

0 Bugs
0 Vulnerabilities
0 Code Smells

Projects Analyzed

Multi-Language

20+ programming languages are supported by SonarQube thanks to our in-house code analyzers, including:

Java	C/C++	C#	COBOL	ABAP	HTML	RPG	JavaScript	TypeScript	Objective C	XML
VB.NET	PL/SQL	T-SQL	Flex	Python	Groovy	PHP	Swift	Visual Basic	PL/I	

Quality Model

Click on the Quality Profile

Quality Profiles

Quality Profiles are collections of rules to apply during an analysis.
For each language there is a default profile. All projects not explicitly assigned to some other profile will be analyzed with the default.

All Profiles ▾

Profile	Projects	Rules	Updated	Used
C#, 1 profile(s)	Default	198	Never	Never
Sonar way	Built-in			
Flex, 1 profile(s)	Default	46	Never	Never
Sonar way	Built-in			
Java, 1 profile(s)	Default	299	Never	Never
Sonar way	Built-in			

Recently Added Rules

- Skipped unit tests should be either removed or fixed
- PHP, not yet activated
- Failed unit tests should be fixed
- PHP, not yet activated
- Source files should have a sufficient density of co...
- PHP, not yet activated
- Source files should not have any duplicated blocks
- PHP, activated on 1 profile(s)
- Lines should have sufficient coverage by tests
- PHP, not yet activated
- Branches should have sufficient coverage by tests
- PHP, not yet activated
- Skipped unit tests should be either removed or fixed
- XML, not yet activated
- Failed unit tests should be fixed
- XML, not yet activated
- Source files should have a sufficient density of co...
- XML, not yet activated

Admin - > Security → users

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Users

Create and administer individual users.

Search by login or name...

	SCM Accounts	Groups	Tokens
A Administrator admin	sonar-administrators sonar-users	0	

1 of 1 shown

Lets Generate a Token 2780c240511f8d738e4d6272df000a4db226b3f9

Tokens

Generate Tokens

Enter Token Name Generate

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Copy 2780c240511f8d738e4d6272df000a4db226b3f9

Name	Created
Jenkins	July 26, 2021

Revoke

Done

Install Jenkins in the EC2 machine

Install Sonar Qube Scanner Plugin

Manage Jenkins → Manage Plugins → Plugin Manager

Click Available → SonarQube and Select **Install without restart**

SonarQube

Updates Available Installed Advanced

Install	Name	Version	Released
<input checked="" type="checkbox"/>	SonarQube Scanner External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	2.13.1	2 mo 27 days ago
<input type="checkbox"/>	Sonar Gerrit External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly. Warning: This plugin version may not be safe to use. Please review the following security notices: • Credentials stored in plain text	2.4.3	1 yr 3 mo ago

[Install without restart](#) [Download now and install after restart](#) Update information obtained: 18 min ago [Check now](#)

e | 34.221.8.230:8080/updateCenter/

Plugin Center

GitHub Branch Source	Success
Pipeline: GitHub Groovy Libraries	Success
Pipeline: Stage View	Success
Git	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Loading plugin extensions	Success
SonarQube Scanner	Success
Loading plugin extensions	Success

[Go back to the top page](#)
(you can start using the installed plugins right away)

Restart Jenkins when installation is complete and no jobs are running

Step 2

Global Tool Configuration → Name as "sonar" and values as SonarQube Scanner 3.2.0.1227

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name: sonar

Install automatically

Install from Maven Central

Version: SonarQube Scanner 3.2.0.1227

Delete Installer

Add Installer

Delete SonarQube Scanner

Add SonarQube Scanner

List of SonarQube Scanner installations on this system

Save Apply

This screenshot shows the 'SonarQube Scanner installations' configuration page. It displays a single entry named 'sonar' with the version 'SonarQube Scanner 3.2.0.1227'. The 'Install automatically' checkbox is checked. There are buttons for 'Delete Installer' and 'Delete SonarQube Scanner'. At the bottom, there are 'Save' and 'Apply' buttons.

Install Gradle Plugin as well

name as gradle-4.10.2 values as gradle-4.10.2

Gradle

Gradle installations

Add Gradle

Gradle

name: gradle-4.10.2

Install automatically

Install from Gradle.org

Version: Gradle 4.10.2

Delete Installer

Add Installer

Save Apply

This screenshot shows the 'Gradle installations' configuration page. It displays a single entry named 'gradle-4.10.2' with the version 'Gradle 4.10.2'. The 'Install automatically' checkbox is checked. There are buttons for 'Delete Installer' and 'Add Installer'. At the bottom, there are 'Save' and 'Apply' buttons.

Step 3

The screenshot shows the Jenkins 'Credentials' page. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main area has a title 'Credentials' with a magnifying glass icon. Below it, a table header includes columns for 'T', 'P', 'Store', 'Domain', 'ID', and 'Name'. A note says 'Icon: S M L'. Underneath, a section titled 'Stores scoped to Jenkins' lists 'Jenkins' under 'Domains' with '(global)' next to it.

Configure SonarQube Credentials

Click Credentials ⇒ Click global → Click Credentials ⇒

Set Kind to Sonar , Scope as Global

This screenshot shows the 'Add Credentials' dialog in Jenkins. The path is 'Dashboard > Credentials > System > Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Secret text'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field contains a series of dots. The 'ID' field is 'sonar'. The 'Description' field is 'sonar'. At the bottom is an 'OK' button.

This screenshot shows the 'Global credentials (unrestricted)' list in Jenkins. The path is 'Dashboard > Credentials > System > Global credentials (unrestricted)'. The table has columns for 'ID', 'Name', 'Kind', and 'Description'. One row is shown: 'sonar' (key icon), 'sonar' (name), 'Secret text' (kind), and 'sonar' (description). An 'X' icon is at the end of the row. A note at the top says 'Credentials that should be available irrespective of domain specification to requirements matching.' and 'Icon: S M L'.

Create and Execute Jenkins Gradle pipeline Job

Dashboard > BuildJob1 >

General Build Triggers Advanced Project Options Pipeline

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/cloudacademy/devops-jenkins-sonarqube

Credentials

- none - Add

Advanced... Add Repository

Save Apply

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

Lightweight checkout

Pipeline Syntax

Save Apply

Run the Pipeline Job

Stage View

Average stage times:
(Average full run time: ~15s)

	prep	build	sonar-scanner
#2 Jul 26 12:19 No Changes	473ms	1s	7s
#1 Jul 26 12:10 No Changes	1s	26s	822ms failed

Permalinks

- Last build (#1), 8 min 11 sec ago
- Last failed build (#1), 8 min 11 sec ago
- Last unsuccessful build (#1), 8 min 11 sec ago
- Last completed build (#1), 8 min 11 sec ago

Go and Check in Sonar Cube

we can see the WebApp Project in Sonar Cube and its metrics are displayed as below

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration

Perspective: Overall Status Sort by: Name Last analysis: July 26, 2021, 12:19 PM

My Favorites All 1 projects Home

Filters

Quality Gate

- Passed: 1
- Warning: 0
- Failed: 0

Reliability (Bugs)

- A: 1
- B: 0
- C: 0
- D: 0

WebApp Passed 81 xs Java

0 A Bugs 0 A Vulnerabilities 9 A Code Smells 0.0% Coverage 17.0% Duplications

1 of 1 shown

Quality Gate Passed

Bugs 0 Vulnerabilities 0

Code Smells 2h 9

Coverage 0.0%

About This Project

No tags

XS 81 Java 81 Lines of Code

Project Activity

July 26, 2021 2 Show More

Quality Gate (Default) Sonar way

Quality Profiles (Java) Sonar way

Project Key GS Copy

Click on the code-smells it will take you to the issues area

Not secure | 34.221.8.230:9000/project/issues?facetMode=effort&id=GS&resolved=false&types=CODE_SMELL

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

WebApp master

Overview Issues Measures Code Activity Administration

My Issues All

Bulk Change

Filters Clear All Filters

Display Mode Effort

Issues

Type Bug 0 Vulnerability 0 Code Smell 2h

Severity

Resolution

Status

Creation Date

src/.../cloudacademy/example/webapp/DockerServlet.java

- 2 duplicated blocks of code must be removed. last year pitfall
- This block of commented-out lines of code should be removed. 3 years ago misra, unused
- Reorder the modifiers to comply with the Java Language Specification. 3 years ago convention
- Reorder the modifiers to comply with the Java Language Specification. 3 years ago convention
- This block of commented-out lines of code should be removed. 3 years ago misra, unused
- This block of commented-out lines of code should be removed. 3 years ago misra, unused

Click the Red boxes to see the issues

Work on the Code and update the Git Branch

Jenkins Pipeline with Artifactory Integration

Run the below Docker-compose file

```

node{

    def server = Artifactory.server 'artifactory'

    def rtGradle = Artifactory.newGradleBuild()

    withCredentials([usernamePassword(
        credentialsId: 'artifactory',
        usernameVariable: 'USERNAME',
        passwordVariable: 'PASSWORD'
    )]) {
        server.username = "${USERNAME}"
        server.password = "${PASSWORD}"
    }

    def buildInfo

    stage('clone'){
        git url: 'https://github.com/cloudacademy/devops-webapp.git'
    }

    stage('Artifactory Config'){
        rtGradle.tool = "gradle-4.10.2"
        rtGradle.deployer repo:'gradle-release-local', server: server
        rtGradle.resolver repo:'jcenter', server: server
    }

    stage('build'){
        rtGradle.run rootDir: "./", buildFile: 'build.gradle', tasks: 'clean build'
    }

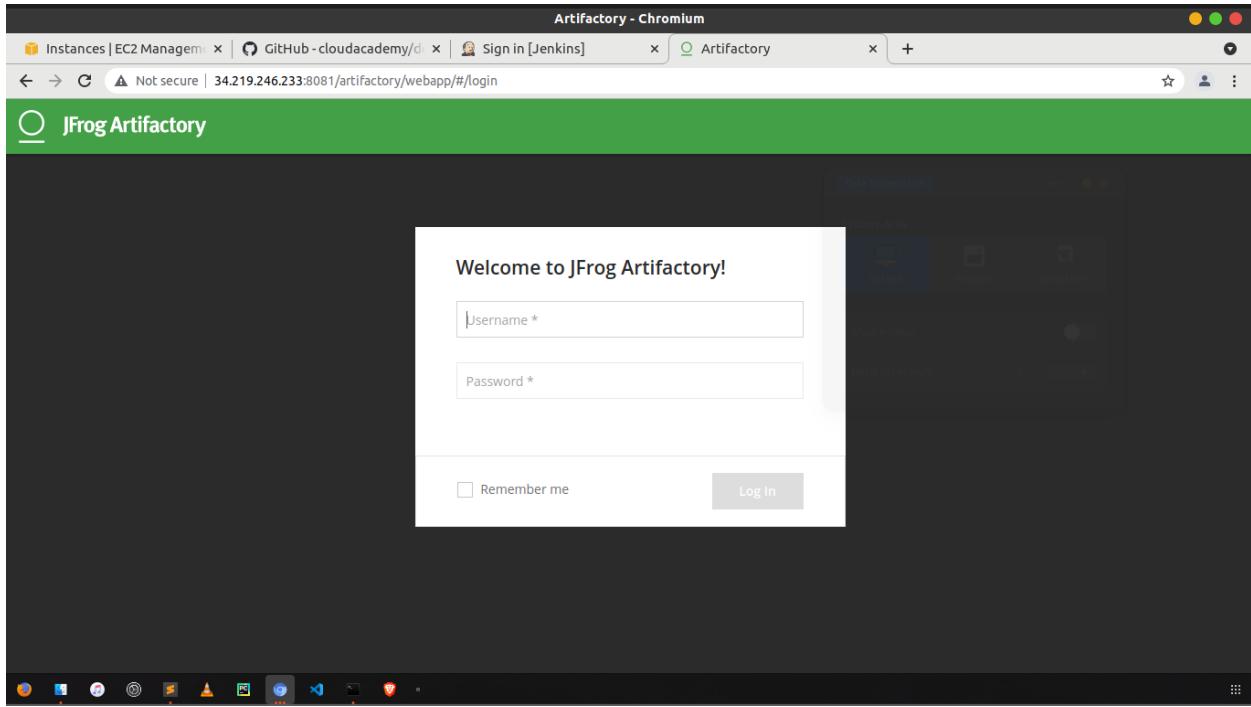
    stage("publish"){
        buildInfo = rtGradle.run rootDir: "./", buildFile: 'build.gradle', tasks: 'artifactoryPublish'
    }

    stage("deploy"){

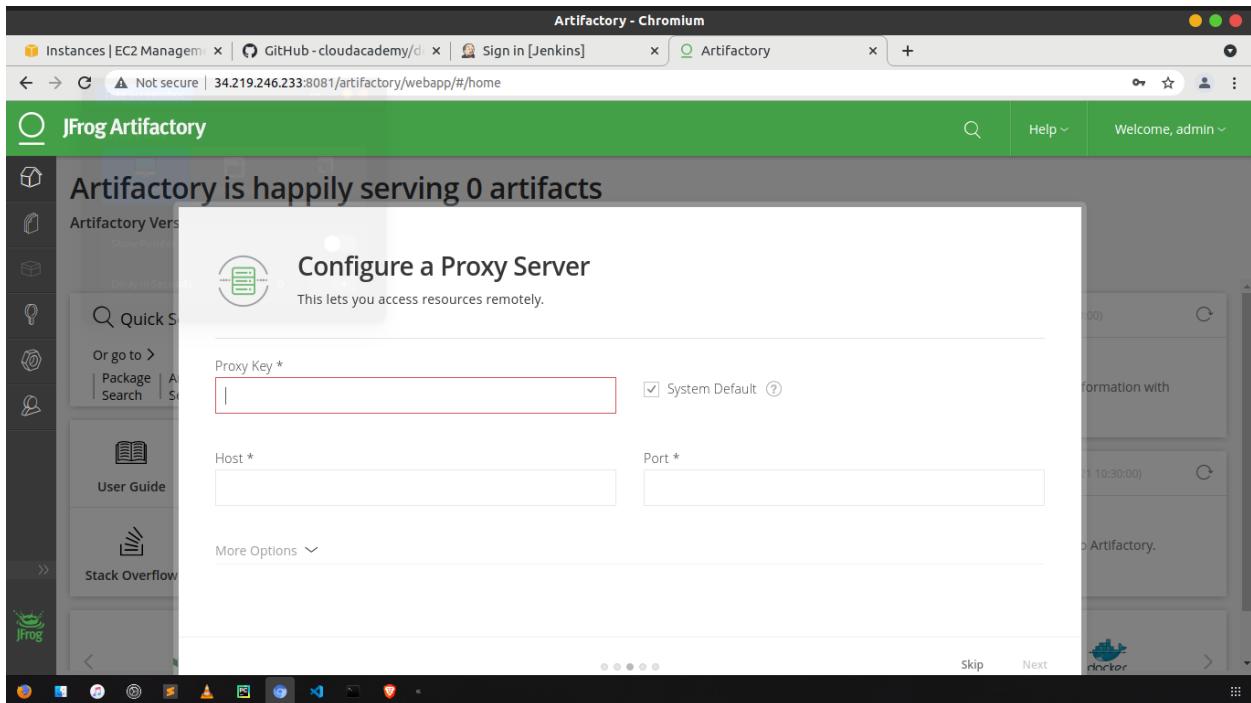
```

```
        server.publishBuildInfo buildInfo  
    }  
}
```

Check artifactory is up and running



Login to Artifactory with admin and password. Once logged in changed the password to qwerty123
Dont Change anything in the below , use default settings



Now we are going to generate Gradle repositories

The screenshot shows the JFrog Artifactory web interface. The main title is "Artifactory is happily serving 0 artifacts". A central modal window is titled "Create Repositories" with the sub-instruction "Select the package type(s) you want - we'll create the default repositories for you! Disabled package types already have default repositories configured." Below this, there is a grid of icons representing various package types: Gradle, Ivy, Maven, SBT, Generic, Bower, Chef, CocoaPods, Conan, CONDA, CRAN, debian, docker, Gems, Git LFS, HELM, npm, NuGet, Opkg, Composer, PyPI, Puppet, RPM, and Vagrant. At the bottom of the modal are "Back", "Skip", and "Create" buttons.

The screenshot shows the JFrog Artifactory web interface. The main title is "Artifactory is happily serving 0 artifacts". A central modal window is titled "Artifactory on-boarding complete!" with the message "Congrats! These are the default repositories we created for you. You're now ready to speed up your software releases! Want to configure your client(s) and get started? Click the Set Me Up button for each repository. Want to learn more about different repository types? Consult the JFrog Artifactory User Guide." Below this, there is a list of repositories: gradle-dev-local, gradle-release-local, jcenter, gradle-dev, and gradle-release. At the bottom of the modal is a "Finish" button.

The screenshot shows the JFrog Artifactory web interface. At the top, there's a header bar with tabs for Instances | EC2 Manager, GitHub - cloudacademy, Sign in [Jenkins], Artifactory, and a plus sign for a new tab. Below the header, the URL is 34.219.246.233:8081/artifactory/webapp/#/home. The main content area has a green header "JFrog Artifactory". It displays the message "Artifactory is happily serving 0 artifacts". Below this, it says "Artifactory Version 6.23.21 (Uptime is 0d 0h 7m 22s)" and "Latest Release: 7.21.8". There's a "Quick Search" bar with a magnifying glass icon. To its right is a "Set Me Up" section with links for "gradle-dev", "gradle-release", "artifactory-build-info", "gradle-dev-local", "gradle-release-local", and "jcenter". Further down are links for "User Guide", "Webinar Signup", "Support Portal", "Stack Overflow", "Blog", and "Rest API". On the right side, there are two sections: "Last Deployed Builds" (27/07/21 10:30:00) which says "No builds to display. Learn how to Integrate your build information with Artifactory." and "Most Downloaded Artifacts" (27/07/21 10:30:00) which says "No artifacts to display. Learn how to deploy your artifacts to Artifactory.". At the bottom, there's a footer bar with various icons.

Setting up Gradle and Artifactory plugin

Manage Jenkins ⇒ Manage Plugins ⇒ Go to Available ⇒ Search for **artifactory**

The screenshot shows the Jenkins Plugin Manager. The title bar says "Available Plugins [Jenkins] - Chromium". The URL is 34.219.246.233:8080/pluginManager/available. The top navigation bar includes "Dashboard", "Plugin Manager", "Back to Dashboard", "Manage Jenkins", and "Update Center". On the right, there's a search bar with "search" and a magnifying glass icon, and a user profile for "Administrator". Below the navigation, there are tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A sub-header says "Install : Name". The main content area lists the "Artifactory" plugin under the "Available" tab. The plugin details are as follows:

Name	Version	Released
Artifactory pipeline	3.12.5	6 days 21 hr ago
lambdatest-automation Artifactory auto generated POM	1.19.4	1 mo 8 days ago

At the bottom of the plugin card, there are buttons for "Install without restart" (highlighted in blue), "Download now and install after restart", and "Check now". Below the plugin cards, there are links for "REST API" and "Jenkins 2.289.2". The bottom of the screen shows the Windows taskbar with various pinned icons.

The screenshot shows the Jenkins Update Center page. The left sidebar has 'Manage Jenkins' and 'Deploy to Jenkins'. The main area lists installed plugins with green checkmarks and 'Success' status: SSH Build Agents, Matrix Authorization Strategy, PAM Authentication, LDAP, Email Extension, Mailer, Loading plugin extensions, Config File Provider, Ivy, Javadoc, Maven Integration, Artifactory, and Loading plugin extensions. Below the list are links to go back to the top page or restart Jenkins.

Lets Configure the Artifactory

Click the Manage jenkins menu item ⇒ configure system

Provide the required details and test the connection

The screenshot shows the Jenkins Configure System page under 'JFrog Platform Instances'. It includes fields for 'Instance ID' (artifactory) and 'JFrog Platform URL' (http://artifactory:8081/). A 'Default Deployer Credentials' section shows 'Username' (admin) and 'Password' (*****). A note says 'Found JFrog Artifactory 6.23.21 at http://artifactory:8081/artifactory'. Buttons for 'Save' and 'Apply' are at the bottom.

Lets Store the artifactory credentials

go to Manage Credentials

The screenshot shows the Jenkins 'Manage Credentials' interface. The URL is 34.219.246.233:8080/credentials/store/system/domain/_/newCredentials. The navigation path is: Dashboard > Credentials > System > Global credentials (unrestricted). A sidebar on the left has 'Back to credential domains' and 'Add Credentials' buttons. The main form is for creating a 'Username with password' credential. It includes fields for 'Scope' (set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'Username' ('admin'), 'Password' ('*****'), 'ID' ('artifactory'), and 'Description' ('artifactory'). There is also a checked checkbox 'Treat username as secret'. At the bottom is an 'OK' button.

Add Gradle Plugin

Go to Global Tool Configuration and add gradle 5.6.4

The screenshot shows the Jenkins 'Global Tool Configuration' page for 'Gradle'. The title is 'Gradle'. Under 'Gradle installations', there is a 'Gradle' entry with 'name' set to 'gradle-5.6.4'. A checked checkbox 'Install automatically' is present. Below it is a section for 'Install from Gradle.org' with a dropdown 'Version' set to 'Gradle 5.6.4'. A red 'Delete Installer' button is visible. At the bottom are 'Save' and 'Apply' buttons.

Lets Build a Jenkins Pipeline For Gradle Project

Create a new project as pipeline Project

New Item [Jenkins] - Chromium

Instances | EC2 Manager | GitHub - cloudacademy/d | New Item [Jenkins] | Artifactory

Not secure | 34.219.246.233:8080/view/all/newJob

Jenkins

Administrator log out

Dashboard All

Enter an item name

BuildJob1

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
OK for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Build Project from GIT SCM

BuildJob1 Config [Jenkins] - Chromium

Instances | EC2 Manager | GitHub - cloudacademy/d | BuildJob1 Config [Jenkins] | Artifactory

Not secure | 34.219.246.233:8080/job/BuildJob1/configure

Dashboard BuildJob1

General Build Triggers Advanced Project Options Pipeline

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/cloudacademy/devops-jenkins-artifactory

Credentials

- none - Add

Advanced... Add Repository

Save Apply

Trigger the Build

The screenshot shows the Jenkins Pipeline BuildJob1 interface. On the left, a sidebar menu includes options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area is titled "Pipeline BuildJob1" and "Stage View". It displays five stages: Clone (2s), Artifactory Configuration (136ms), Gradle Build (35s), Gradle Publish (2s), and Artifactory Publish Build Info (401ms). Below the stages, a summary states "Average stage times: (Average full run time: ~46s)". A build history section shows a single build (#1) from July 27 at 10:52, with "No Changes". The status bar at the bottom indicates a successful build.

Build completed without any issues

The screenshot shows the Jenkins Build #1 page for BuildJob1. The main title is "Build #1 (27-Jul-2021 05:22:12)". Key details include: Started by user Administrator, Revision: 2b2a90f15d9000153bc9e2a26395224553edfd32, Repository: https://github.com/cloudacademy/devops-webapp.git, and Artifactory Build Info. The sidebar on the left lists options such as Status, Changes, Console Output, Edit Build Information, Delete build '#1', Git Build Data, Artifactory Build Info, Replay, Pipeline Steps, and Workspaces. The status bar at the bottom shows the build took 46 seconds.

Check the Build status

Click on the artifactory link

Notice how there are *Artifactory Build Info* menu links that will take you directly into the build information registered within Artifactory. These links currently use the internal Artifactory

docker host name within the URL, so it will not work from your browser. To get these links to work, replace the internal Artifactory docker host name with the Public IP address that is assigned to your `cicd.platform.instance` EC2 instance. For example:

`http://artifactory:8081/artifactory/webapp/builds/BuildJob1/1`

becomes

`http://54.149.60.36:8081/artifactory/webapp/builds/BuildJob1/1`

if its not opening go to the artifactory itself and check for it

The screenshot shows the JFrog Artifactory web interface. On the left, there are two cards: 'Last Deployed Builds' (BuildJob1 #1) and 'Most Downloaded Artifacts' (No artifacts to display). On the right, the main panel displays 'Builds - BuildJob1' with details: Agent (Jenkins2.285.2), Pipeline (Pipeline), Started (27 July, 2021 10:52...), Duration (46.2 seconds), Principal (Admin), and Artifactory Principal (admin). Below this, a table shows 'Published Modules' with one entry: org.cloudacademy.example.webapp.BuildJob1.1.0, which has 2 artifacts and 55 dependencies.

We can view the war File in the gradle-release-local repo as below

The screenshot shows the JFrog Artifactory Artifact Repository Browser. The left sidebar shows a tree view of repositories: Simple, flyway-commandline-3.2.1-macosx-x64.tar.gz, flyway-commandline-3.2.1-windows-x64.zip, last_updated.txt, robots.txt, gradle-release, artifactory-build-info, gradle-dev-local, gradle-release-local, org.cloudacademy.example.webapp.BuildJob1, 1.0, BuildJob1-1.0.war, ivy-1.0.xml, jcenter, jcenter-cache, and Trash Can. The right panel shows the details for 'BuildJob1-1.0.war' under the 'gradle-release-local' repository. The 'General' tab is selected, showing the following information:

Name:	BuildJob1-1.0.war
Repository Path:	gradle-release-local/org/cloudacademy/example/webapp/BuildJob1/1.0/BuildJob1-1.0.war
Module ID:	N/A
Deployed By:	admin
Size:	18.84 MB
Created:	27-07-21 05:22:58 +00:00
Last Modified:	27-07-21 05:22:57 +00:00
Downloads:	0
Remote Downloads:	0

The 'Actions' dropdown menu includes 'Download' and 'Actions'.

Lets search the Artifacts

JFrog Artifactory

Search Artifacts

Search Type: Quick
Buildjob1 | Add search criteria... | Clear | Search

Search Results
Filter by Artifact | Delete

Artifact	Path	Repository	Modified

Artifactory - Chromium

Instances | EC2 Manager | devops-jenkins-artifactory | BuildJob1 #1 [Jenkins] | artifactory | HTTP Status 404 - Not Found | Artifactory | + | Welcome, admin |

Not secure | 34.219.246.233:8081/artifactory/webapp/#/search/quick/eyJzZWY2giOjxdWljaylsInF1ZXJSIjoiQnVpbGRKb2IxIn0=

JFrog Artifactory

Search Artifacts

Search Type: Quick
Buildjob1 | Add search criteria... | Clear | Search

Search Results - 1 Items | Stash Results

Filter by Artifact | Delete

Artifact	Path	Repository	Modified
BuildJob1-1.0.war	org/cloudacademy/example/webapp/BuildJob1/1.0	gradle-release-local	27-07-21 05:22:57 +00:00

<https://github.com/cloudacademy/devops-jenkins-artifactory/blob/master/Jenkinsfile>

Scripted Pipeline Syntax

To upload or download files you first need to create a spec which is a JSON file that specifies which files should be uploaded or downloaded and the target path.

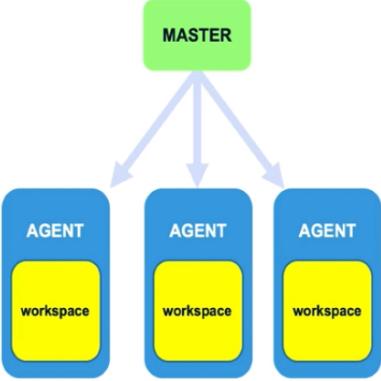
<https://www.jfrog.com/confluence/display/JFROG/Scripted+Pipeline+Syntax>



Jenkins Multi Node

Jenkins master can be configured to distribute build jobs amongst multiple Jenkins build agents
Requirement needed when you don't want to compromise the security of the master server.
Connection between master and agents are based on the TCP/IP network connections

Jenkins - Multi Node Deployment



Multi Node

Create a multi node setup when:

- build job numbers increase
- frequency of builds increase
- complexity of builds increase
- require multiple software frameworks and/or operating system build requirements

frequency of builds, complexity of builds,

Jenkins – Build Agent Credential Management

Master Server

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: jenkins

Private Key: Enter directly
Key: -----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEA3zH0YCJS/Q0/F48tCEnSasQInV8pxT3tNy1OKo4o6e20+b
9xqj0AGfr5wymBeRT19gzwkgh25PLJNz88un7MDH1GS/tZt6jNyk6ODXaG
xW1tP0nDze6dKzBbe4WTg5vcEaV2bjyztXzvZVRjPzaZgVn3MLitikOh2D5oE
bf31brGC/Xifm6dPCx2u/9kj5CofcheXp+41OjGqC9E0gXcLnvtkMKGYkHLpw
-----END RSA PRIVATE KEY-----

Passphrase:

ID:

Description: Jenkins Agent Credential

copying in the private key portion

Lets Create a New Node

Jenkins – Add Node

Master Server

The screenshot shows the Jenkins 'Add Node' dialog. At the top, it says 'Jenkins – Add Node' and 'Master Server'. Below that is a navigation bar with links like 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. The main form has a 'Node name' field containing 'BuildNode1' and a radio button selected for 'Permanent Agent'. A tooltip explains that this adds a plain, permanent agent to Jenkins. There are also sections for 'Build Queue' (empty) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom right are 'OK' and 'Cancel' buttons.

Permanency dictates whether the agent

Jenkins – Add Node Config

Master Server

The screenshot shows the Jenkins 'Add Node Config' dialog for 'BuildNode1'. It contains fields for Name (BuildNode1), Description (BuildNode1), # of executors (4), Remote root directory (/var/lib/jenkins/), Labels (linux), Usage (Use this node as much as possible), Launch method (Launch agent agents via SSH), Host (54.202.191.228), Credentials (jenkins (Jenkins Agent)), and Host Key Verification Strategy (Known hosts file Verification Strategy). At the bottom left is an 'Advanced...' button. At the bottom right are 'OK' and 'Cancel' buttons.

The remote root directory.

View the Dashboard

Jenkins - Agent Connected

Master Server

Back to Dashboard | Manage Jenkins | New Node | Configure

Build Queue
No builds in the queue.

Build Executor Status

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	master	Mac OS X (x86_64)	In sync	13.88 GB	771.00 MB	13.88 GB	0ms
2	BuildNode1	Linux (amd64)	In sync	13.93 GB	1024.00 MB	13.93 GB	2707ms
	Data obtained		38 sec	38 sec	34 sec	34 sec	37 sec

Refresh status

we are now presented with a view of the stats associated

Typical Master slave Jenkins

Jenkins - Build Agent Connectivity

Troubleshooting

Typical problems often encountered when launching agents are:

- Inbound SSH connections on port 22 are blocked on the Agent
- The SSH daemon has not been started on the Agent
- SSH key mismanagement

are inbound SSH connections on port 22

Jenkins Pipeline

it contains all of our pipeline information
can be treated as other coding Files , it can be versioned as well.
this File should not have credentials stored in it. we should use internal jenkins credential store and then jenkins have them to query at the runtime.

```
environment{  
AWS_ACCESS_KEY_ID = credentials('key-id')  
AWS_SECRET_ACCESS_KEY= credentials('secret-id')  
}
```

BlueOcean

blueocen makes pipeline as first class citizen when it comes to administration
blueocean interface

The screenshot shows the Jenkins Blue Ocean interface. At the top, a blue header bar displays the text "Jenkins Blue Ocean". Below this, a white section is titled "Continuous Delivery for every team". A sub-section below it states: "Blue Ocean puts Continuous Delivery in reach of any team without sacrificing the power and sophistication of Jenkins." Six circular icons represent different features: "Visual Pipeline Editor" (pencil), "Pipeline visualization" (eye), "Diagnosis" (heartbeat), "Github & Git" (GitHub logo), "Personalization" (person), and "100% Open Source" (keyhole). Below each icon is a brief description. At the bottom of the interface, a video player bar indicates the video is at 0:26 of 3:46. The text "and see some examples of the Blue Ocean user interface" is overlaid on the video player bar.

Jenkins Blue Ocean – Visual Pipeline Editor

The user interface is intuitive

A screenshot of the Jenkins Blue Ocean Visual Pipeline Editor. It shows a pipeline diagram with nodes: Start, Clone, Build, Publish, and End. The Build node has two parallel steps, P1 and P2. A sidebar on the right lists various step types: Shell Script (selected), Print Message, Enforce time limit, Retry the body up to N times, Sleep, Windows Batch Script, Archive the artifacts, Allocate node, and Allocate workspace. A Jenkins logo is in the bottom right corner.

Jenkins Blue Ocean – Pipeline Execution

yellow for test failures, and red for build failures.

A screenshot of the Jenkins Blue Ocean Pipeline Execution view. It shows the same pipeline diagram as the editor, but all nodes are marked with green checkmarks, indicating successful execution. Below the diagram, there is a section for artifact archiving and a button to restart the publish process. A Jenkins logo is in the bottom right corner.

Click on the View will give the build logs

Artifact views can be used to easily get the artifacts

Jenkins Blue Ocean – Build Artifacts

The screenshot shows the Jenkins Blue Ocean interface. At the top, there's a navigation bar with tabs for Pipeline, Changes, Tests, Artifacts, and Logout. Below the navigation is a summary card for a pipeline run named 'devops-webapp2' with a status of '2'. It shows the Branch: master, Commit: 7392b27, and a timestamp of 15 minutes ago. The 'Artifacts' tab is selected. A table lists two artifacts: 'pipeline.log' and 'build/libs/devops-webapp2_master-1.0.war'. Both have download icons next to them. A 'Download All' button is at the bottom of the table. The background features a video player UI with a play button, volume, and settings icons.

INstall Blueocean

Blueocean is the new user experience for Jenkins

Jenkins Blue Ocean – Installation

The screenshot shows the Jenkins plugin manager. A search bar at the top contains the text 'blue ocean'. Below it, there are tabs for Updates, Available, Installed, and Advanced. The Available tab is selected. A table lists several Blue Ocean-related plugins, all of which are version 1.4.1. The plugins include 'Blue Ocean', 'Common API for Blue Ocean', 'Config API for Blue Ocean', 'Dashboard for Blue Ocean', 'Events API for Blue Ocean', 'Git Pipeline for Blue Ocean', 'GitHub Pipeline for Blue Ocean', 'i18n for Blue Ocean', 'JWT for Blue Ocean', 'Personalization for Blue Ocean', 'Pipeline implementation for Blue Ocean', 'REST API for Blue Ocean', and 'REST Implementation for Blue Ocean'. At the bottom of the screen, there's a message: 'and then search for the Blue Ocean plugin'.

We can use the BlueOcean to Create the Jenkins File as well.

Jenkins and Docker

Integrating Docker into the Jenkins build System

A single Docker container is configured at the pipeline level to be used for the entire pipeline build execution

The slide title is "Jenkins Docker Builds – Single Docker Container". It features a Jenkinsfile code block and a Jenkins logo. The Jenkinsfile contains a pipeline definition with a single agent block and three stages: Clone, Build, and Test, all using the same 'openjdk:latest' Docker image.

```
pipeline {
    agent {
        docker {
            image 'openjdk:latest'
        }
    }
    stages {
        stage('Clone') {
            steps {
                //clone
            }
        }
        stage('Build') {
            steps {
                //build
            }
        }
        stage('Test') {
            steps {
                //test
            }
        }
    }
}
```

Jenkinsfile

Option 1

A single Docker container is configured at the pipeline level to be used for the entire pipeline build execution

Individual docker container configured per stage - resulting pipeline build execution is segmented over multiple docker containers

The slide title is "Jenkins Docker Builds – Individual Docker Containers". It features a Jenkinsfile code block and a Jenkins logo. The Jenkinsfile defines a pipeline with two stages: 'Build API1' and 'Build API2'. Each stage has its own agent block, where the 'Build API1' stage uses a 'maven:3-alpine' Docker image and the 'Build API2' stage uses an 'openjdk:latest' Docker image.

```
pipeline {
    agent none
    stages {
        stage('Build API1') {
            agent {
                docker { image
'maven:3-alpine' }
            }
            steps {
                //build API1
            }
        }
        stage('Build API2') {
            agent {
                docker { image
'openjdk:latest' }
            }
            steps {
                //build API2
            }
        }
    }
}
```

Jenkinsfile

Option 2

Individual docker containers configured per stage - the resulting full pipeline build execution is segmented over multiple docker containers

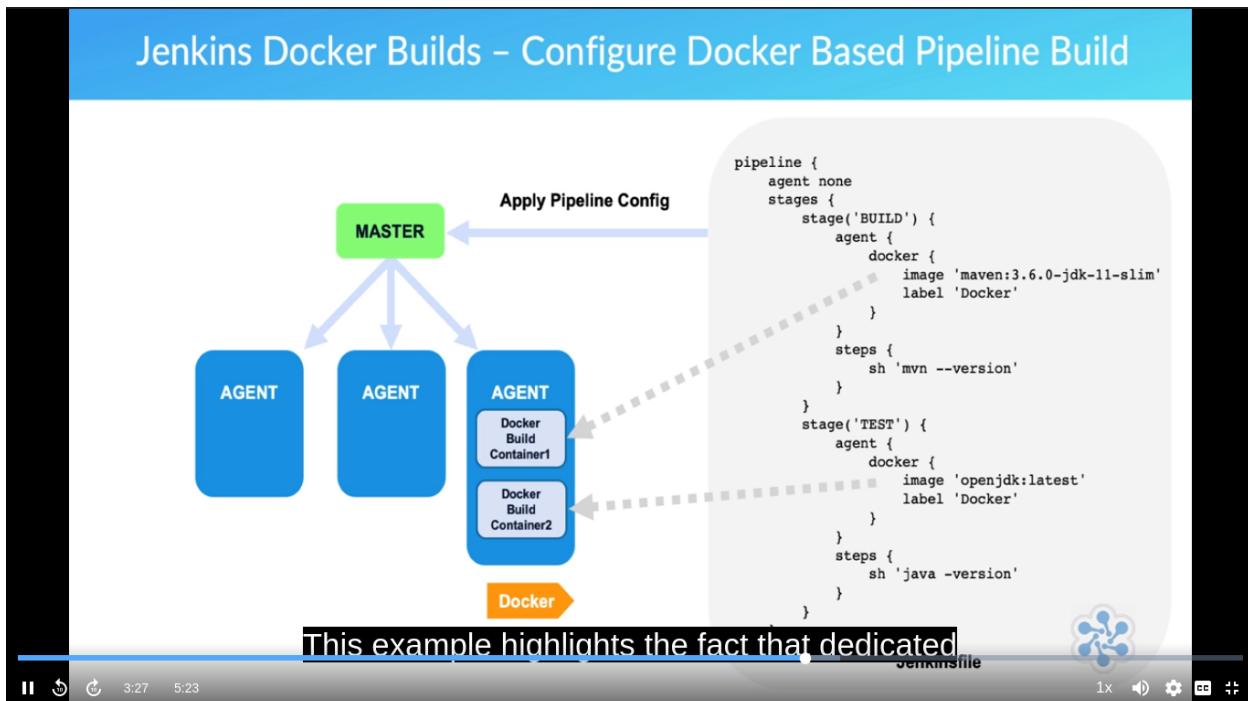
An individual Docker container is configured per stage.

Docker File hosted in the Project Root Directory

Pipeline configuration is instructed to perform a docker build to create a customised docker image complete with updated build tool chain, and then later used for the actual pipeline build execution.

Step 0 :- We need to install docker in all the agent as well the master

We can use Different Docker images for different stages as well



Lets Create A Simple Docker Pipeline Job

Add the below Pipeline code

```
pipeline{  
    agent{  
        label 'master'  
    }  
    stages{  
        stage('docker'){  
            steps{  
                sh "whoami"  
                sh "docker info"  
                sh "docker ps"  
            }  
        }  
    }  
}
```

JENKINS-AS PIPELINE BuildJob4 Config [Jenkins] - Chromium

localhost:8080/job/BuildJob4/configure

Dashboard > BuildJob4 >

Pipeline

Definition

Pipeline script

```
1 pipeline{
2     agent{
3         label 'master'
4     }
5     stages{
6         stage('docker'){
7             steps{
8                 sh "whoami"
9                 sh "docker info"
10                sh "docker ps"
11            }
12        }
13    }
14 }
```

try sample Pipeline... ?

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

BuildJob1 #1 Console [Jenkins] - Chromium

localhost:8080/job/BuildJob1/1/console

Dashboard > BuildJob1 > #1

```
Kernel Version: 5.11.0-25-generic
Operating System: Ubuntu 21.04
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 11.62GiB
Name: esakpc
ID: QC62:NVSR:7JNW:747D:I6VJ:MASL:OUEH:GEE:DSQH:DJBR:CARB:AQIJ
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
127.0.0.0/8
Live Restore Enabled: false

[Pipeline] sh
+ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.289.2

Install Docker and dockerpipeline plugin

The screenshot shows the Jenkins Plugin Manager interface. The search bar at the top contains the text 'docker'. Below it, there are four tabs: 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A table lists available plugins under the heading 'Install'. The first plugin listed is 'Docker Pipeline', which is checked and highlighted with a blue border. Other listed plugins include 'Docker', 'Docker Commons', and 'Docker API'. At the bottom of the table are three buttons: 'Install without restart', 'Download now and install after restart' (which is highlighted in blue), and 'Check now'.

```

pipeline{
    agent none
    stages{
        stage('maven'){
            agent {
                docker {
                    image 'maven'
                    label 'master'
                }
            }
            steps{
                echo "Hello, Maven"
                sh "mvn --version"
            }
        }
        stage('JDK'){
            agent {
                docker{
                    image 'openjdk'
                    label 'master'
                }
            }
            steps{
                echo "hello, JVM"
                sh 'java -version'
            }
        }
    }
}

```

We have Download 2 Images and we build it locally
it will be created automatically and once the process is completed it will delete the images

Integrate Docker with Splunk using Jenkins

GitHub - cloudacademy/devops-webapp: Demonstration Java Servlet based WebApp using Gradle for Build Management

Permalink Failed to load latest commit information. Demonstration Java Servlet based WebApp using Gradle for Build Management You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

 <https://github.com/cloudacademy/devops-webapp>

Jenkins + Docker + Splunk Integration. Contribute to [cloudacademy/devops-jenkins-docker-splunk](#) development by creating an account on GitHub.

cloudacademy/devops
jenkins-docker-splunk

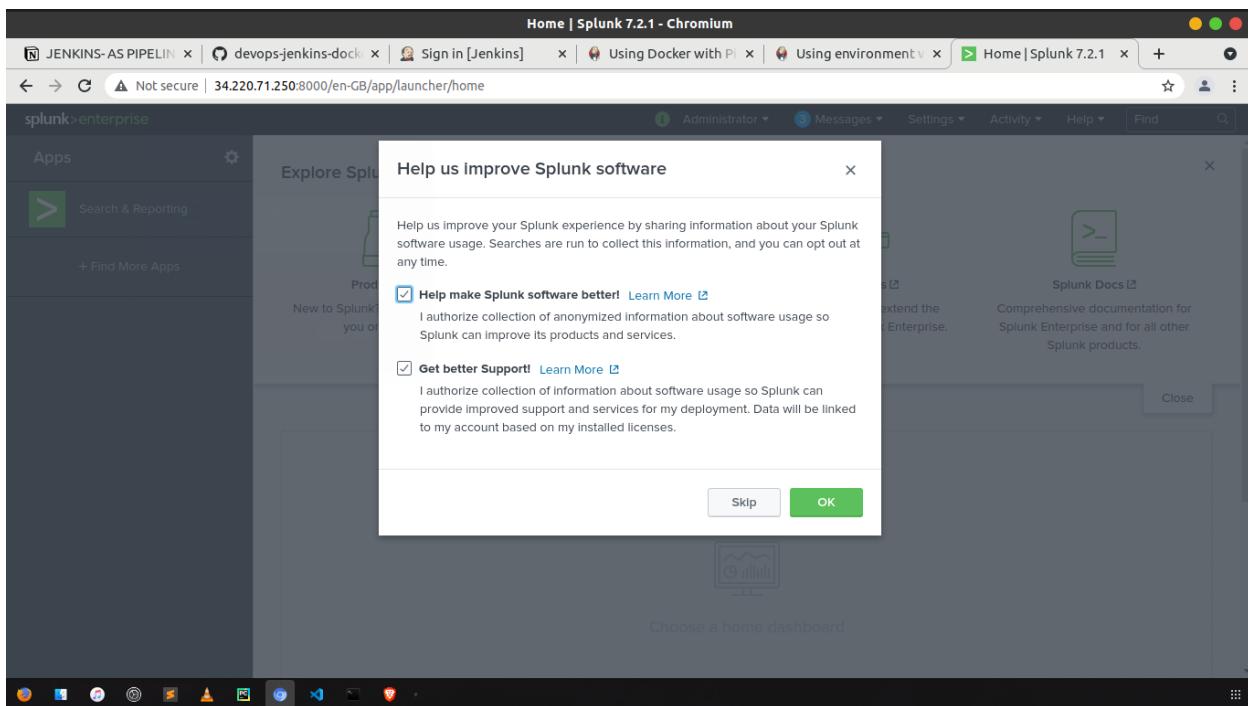
Jenkins + Docker + Splunk Integration

hecking the Docker container in Table Format

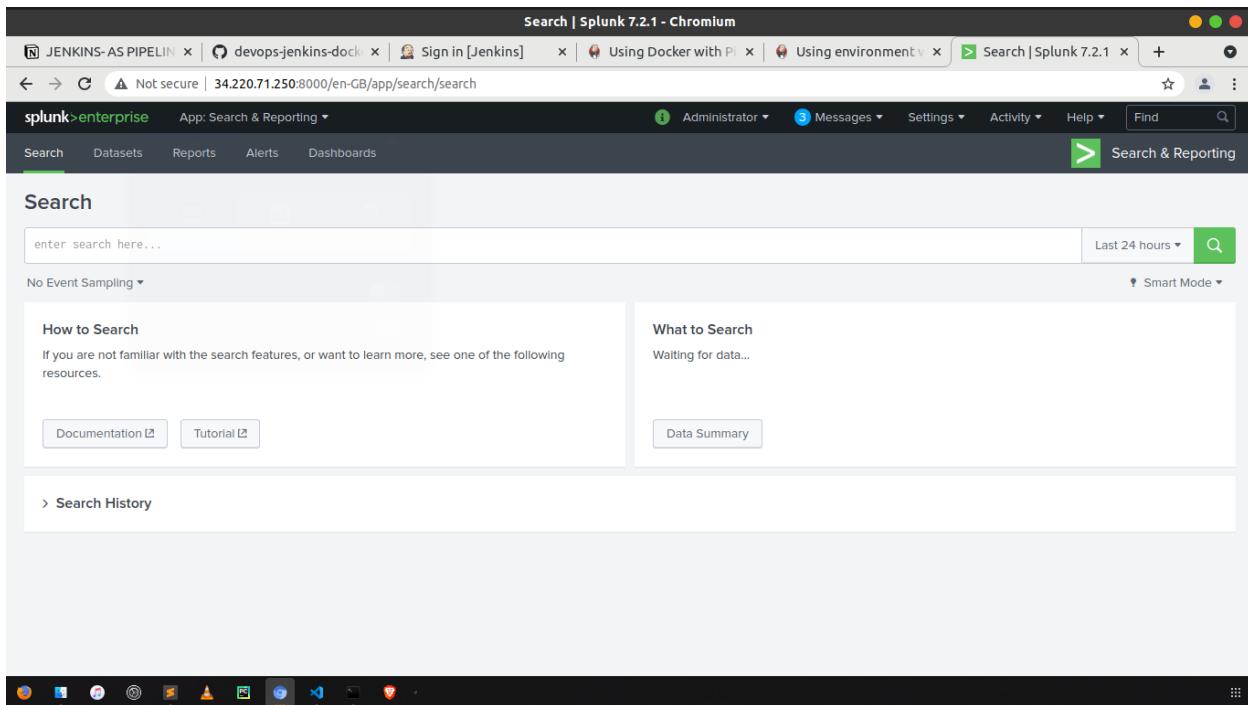
```
docker ps -a --format "table {{.Names}}\t{{.Status}}"  
docker exec -it jenkins cat /var/jenkins_home/secrets/initialAdminPassword
```

Lets Configure the Splunk with Free Trial Licence

1. Login With the provided username admin and password qwerty123



2. Click on the "search and Reporting"



3. Reconfigure Splunk with Trail licence

Goto Setting → Licencing

This screenshot shows the Splunk Manager Licensing interface. At the top, it displays "Licensing - Manager - Splunk - Chromium". The URL is "Not secure | 34.220.71.250:8000/en-GB/manager/system/licensing". The navigation bar includes "splunk>enterprise" and "Apps". On the right, there are links for "Administrator", "Messages", "Settings", "Activity", "Help", and a "Find" search bar.

The main section is titled "Licensing" and indicates "This server is acting as a standalone license server". It features a button to "Change to slave".

Under "Trial license group", it says "This server is configured to use licenses from the Trial license group". There are buttons for "Add license" and "Usage report".

The "Alerts" section notes that licensing alerts notify of indexing warnings and misconfigurations, with a link to "Learn more". It shows "Current" and "Permanent" sections, both of which are currently empty.

The "Local server information" section provides details about the indexer: "Indexer name" (bc8671e4af37), "License expiration" (8 Mar 2020 00:25:43), and "Licensed data volume" (0 MB).

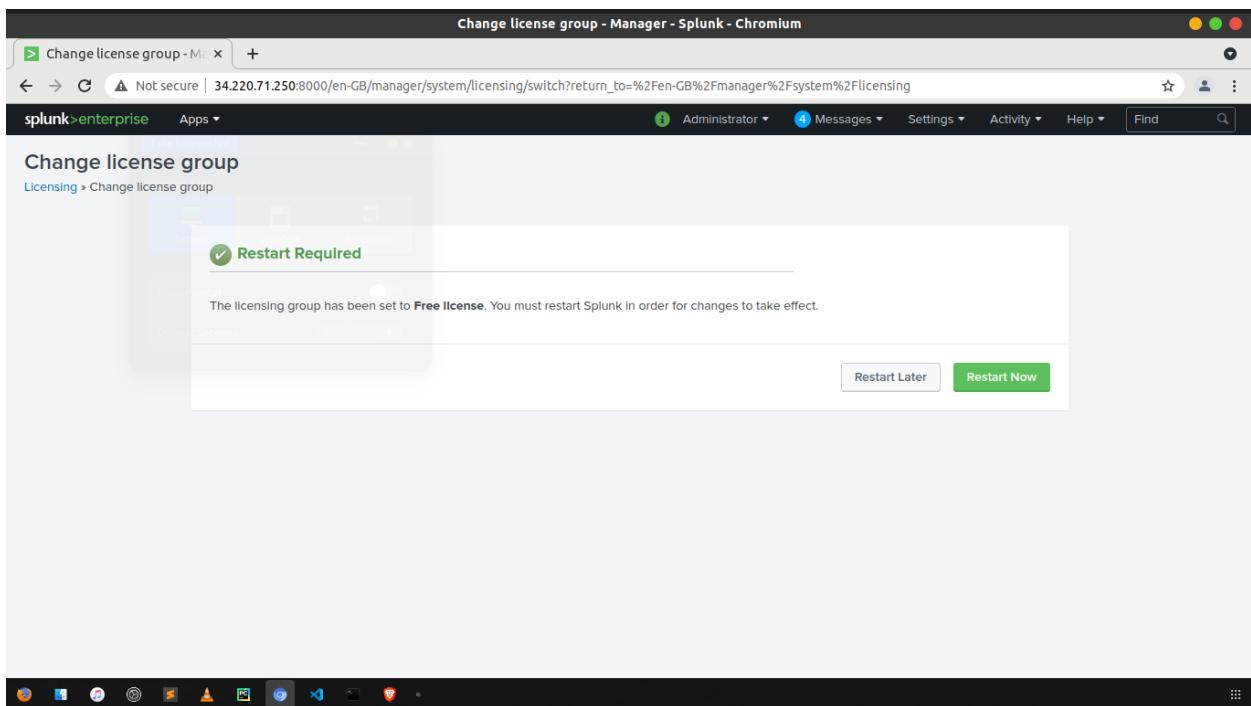
Click on Change Licence Group and Choose Trail Licence

This screenshot shows the "Change license group" dialog box. The title is "Change license group - Manager - Splunk - Chromium". The URL is "Not secure | 34.220.71.250:8000/en-GB/manager/system/licensing/switch?return_to=%2Fen-GB%2Fmanager%2Fsystem%2FLicensing".

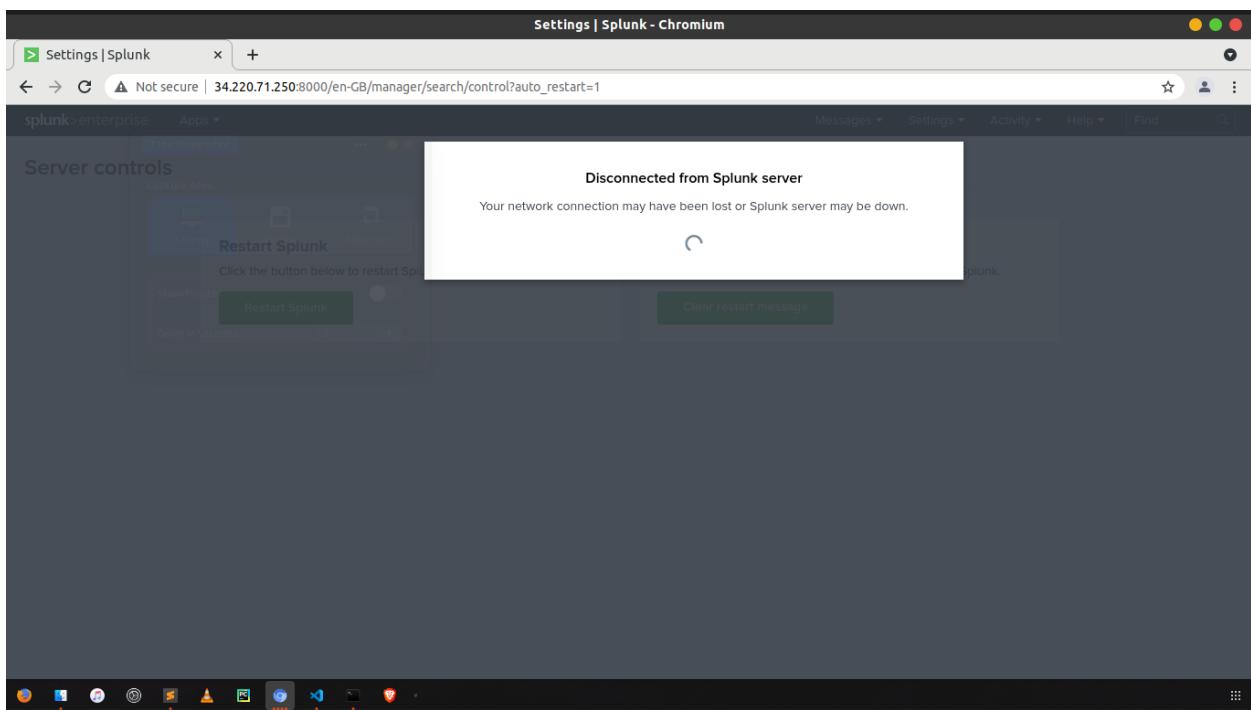
The dialog box contains a list of license types:

- Enterprise license: Adds support for multi-user and distributed deployments, alerting, role-based security, single sign-on, scheduled PDF delivery, and unlimited data volumes. A note states: "There are no valid Splunk Enterprise licenses installed. You will be prompted to install a license if you choose this option."
- Forwarder license: Used for configuring Splunk as a forwarder. A note states: "Use this group when configuring Splunk as a forwarder. Learn more"
- Free license: Used for running Splunk Free. A note states: "Use this group when you are running Splunk Free. This license has no authentication or user and role management, and has a 500MB/day daily indexing volume. Learn more"
- Enterprise Trial license: An included download trial. A note states: "This is your included download trial. IMPORTANT: If you switch to another license, you cannot return to the Trial. You must install an Enterprise license or switch to Splunk Free. There are no valid Splunk Enterprise Trial licenses installed. You will be prompted to install a license if you choose this option."

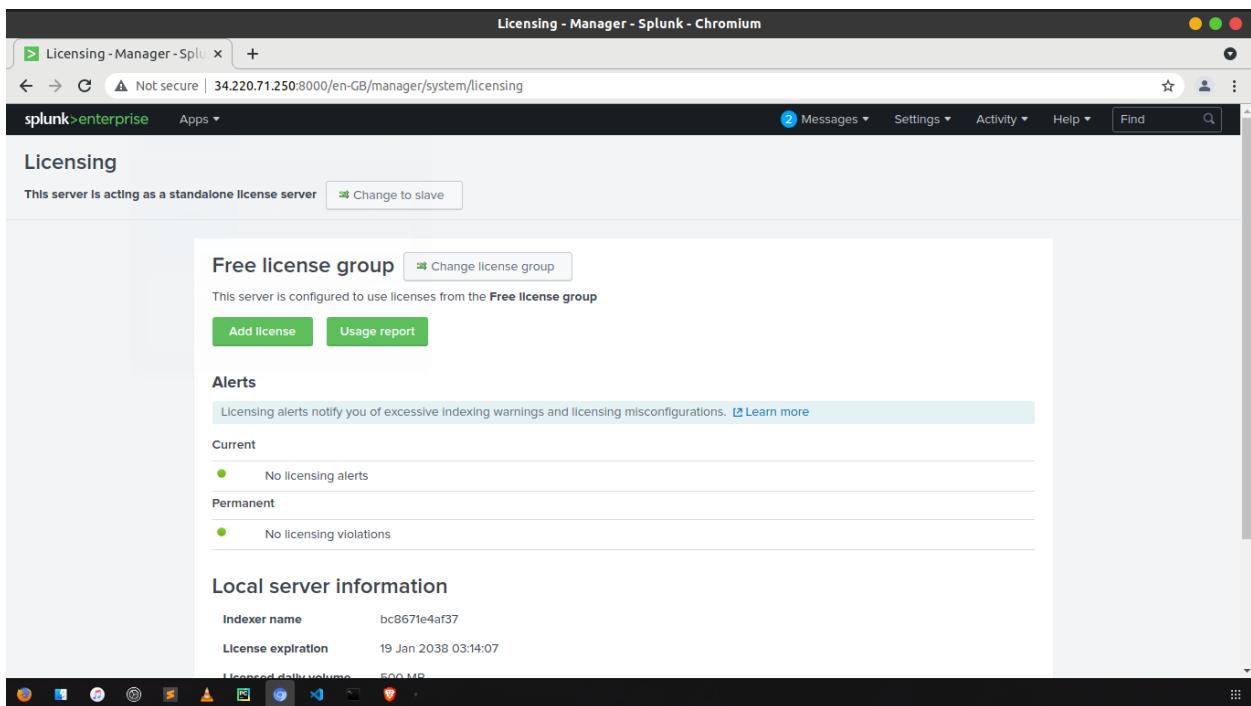
At the bottom right are "Cancel" and "Save" buttons.



Restart the Splunk



Once its Restarted , verify the Splunk is running in free licence



In the following Lab Steps, you will configure a Jenkins build pipeline to build a custom Docker container, which will host a sample Java servlet web application, also built within Jenkins. The sample Java servlet web application is designed to use Log4J2 to perform logging. The corresponding log files generated at runtime will be published to Splunk and, therefore, will be available for analysis within the Search & Reporting area.

In this Lab Step, you will continue using the Jenkins administration web console. You will configure the Gradle and Docker tools required to compile the sample Java servlet web application into a WAR (Web application ARchive) artifact file, and then a docker image that will use Tomcat to serve the WAR artifact file created in the first build step. You will configure the Gradle and Docker tools to be automatically installed at build time by Jenkins.

Step 0 Install Docker pipeline Plugin

The screenshot shows the Jenkins Plugin Manager interface. A search bar at the top contains the text "docker". Below it, tabs for "Updates", "Available", "Installed", and "Advanced" are visible, with "Available" selected. A table lists four Docker-related plugins:

Install	Name	Version	Released
<input type="checkbox"/>	Docker	1.2.2	6 mo 3 days ago
<input type="checkbox"/>	Docker Commons	1.17	1 yr 0 mo ago
<input checked="" type="checkbox"/>	Docker Pipeline	1.26	5 mo 3 days ago
	Docker API		

Below the table are three buttons: "Install without restart", "Download now and install after restart", and "Check now".

Verify it installed sucessfully

The screenshot shows the Jenkins Update Center interface. It displays a list of installed plugins with their status as "Success":

- Pipeline: Stage View
- Git
- SSH Build Agents
- Matrix Authorization Strategy
- PAM Authentication
- LDAP
- Email Extension
- Mailer
- Loading plugin extensions
- Authentication Tokens API
- Docker Commons
- Docker Pipeline
- Loading plugin extensions

At the bottom, there are links to "Go back to the top page" and "Restart Jenkins when installation is complete and no jobs are running".

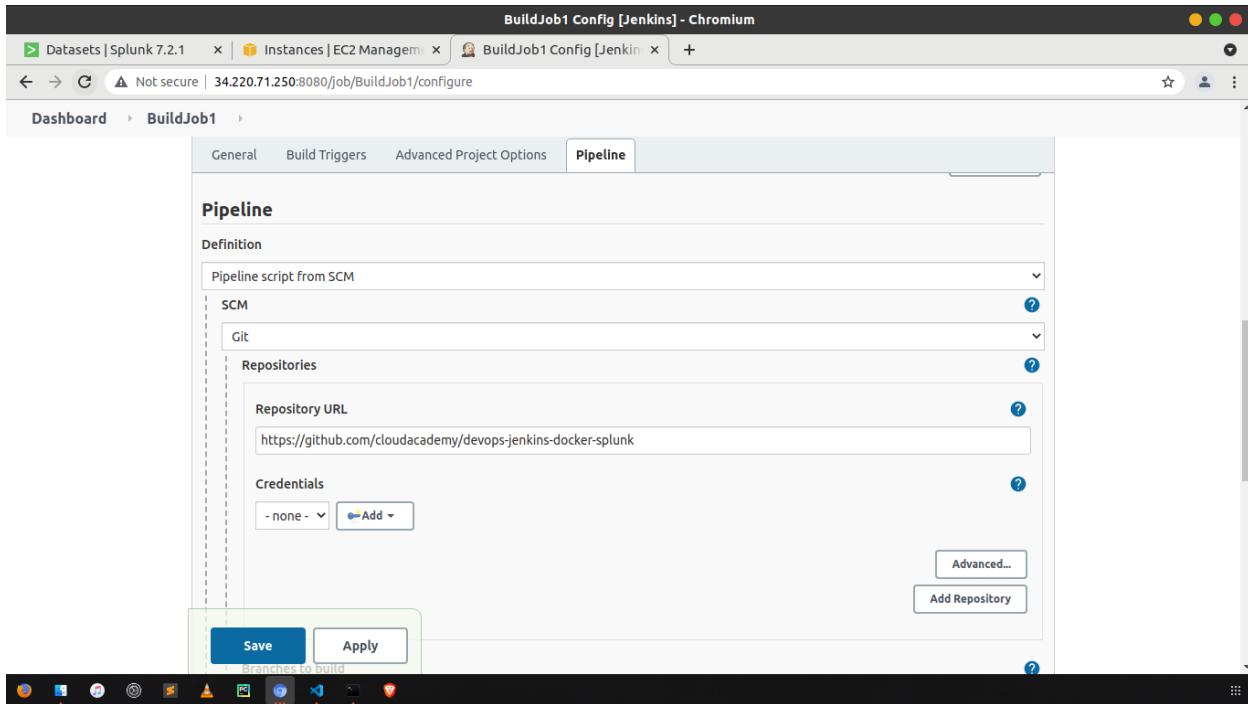
Install Gradle in Jenkins

The screenshot shows the Jenkins Global Tool Configuration interface for Gradle. The 'Gradle' section is active, displaying configuration for 'gradle-4.10.2'. The 'Install automatically' checkbox is checked. Below it, there's an 'Install from Gradle.org' section with a dropdown menu set to 'Gradle 4.10.2'. Buttons for 'Delete Installer' and 'Delete Gradle' are visible on the right. A 'Save' and 'Apply' button are at the bottom. A note at the bottom states: 'List of Gradle installations on this system'.

Install Docker

The screenshot shows the Jenkins Global Tool Configuration interface for Docker. The 'Docker' section is active, displaying configuration for 'docker-latest'. The 'Name' field contains 'docker-latest' and the 'Installation root' field contains 'latest'. A warning message 'latest is not a directory on the Jenkins master (but perhaps it exists on some agents)' is shown. The 'Install automatically' checkbox is unchecked. Buttons for 'Delete Docker' and 'Save' are visible on the right. A note at the bottom states: 'List of Docker installations on this system'.

Lets Create Jenkins Pipeline



Lets Deep Dive into the Jenkins File

[devops-jenkins-docker-splunk/Jenkinsfile at master · cloudacademy/devops-jenkins-docker-splunk](https://github.com/cloudacademy/devops-jenkins-docker-splunk/blob/master/Jenkinsfile)
Jenkins + Docker + Splunk Integration. Contribute to cloudacademy/devops-jenkins-docker-splunk development by creating an account on GitHub.

cloudacademy/c
jenkins-docker-
Jenkins + Docker + Splunk Inte

<https://github.com/cloudacademy/devops-jenkins-docker-splunk/blob/master/Jenkinsfile>

Ax 1 Contributor 0 Issues ☆ 4 St

```
//START-OF-SCRIPT
node {
    def SPLUNK_HOSTNAME='splunk'
    def DOCKER_HOME = tool name: 'docker-latest'
    def GRADLE_HOME = tool name: 'gradle-4.10.2', type: 'hudson.plugins.gradle.GradleInstallation'
    def REPO_URL = 'https://github.com/cloudacademy/devops-webapp.git'
    def DOCKERHUB_REPO = 'cloudacademydevops/webapp'

    stage('Clone') {
        git url: REPO_URL
    }

    stage('Build') {
        sh "${GRADLE_HOME}/bin/gradle build"
        sh "ls -la build/libs/*.war"
        sh "echo ====="
        sh "cp build/libs/*.war docker/webapp.war"
        sh "pwd"
        sh "ls -la"
        sh "ls -la ./docker"
    }

    stage ('Docker Build') {
        docker.withTool('docker-latest') {
            sh "printenv"
            sh "pwd"
            sh "ls -la"
            def image1 = docker.build("${DOCKERHUB_REPO}:${BUILD_NUMBER}", "./docker")
            def image2 = docker.build("${DOCKERHUB_REPO}:latest", "./docker")
        }
    }
}
```

```

/*
stage ('Docker Push') {
    docker.withTool('docker-latest') {
        withCredentials([usernamePassword(credentialsId: 'dockerhub',
                                         usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {
            sh "docker login -u ${USERNAME} -p ${PASSWORD} https://index.docker.io/v1"
            sh "docker push ${DOCKERHUB_REPO}:${BUILD_NUMBER}"
            sh "docker push ${DOCKERHUB_REPO}:latest"
        }
    }
}
*/
}

//END-OF-SCRIPT

```

Lets trigger the Build

```

BuildJob1 #5 Console [Jenkins] - Chromium
Datasets | Splunk | Instances | EC2 | BuildJob1 #5 | docker: not found | Docker: not found | CloudBees Jenkins | docker-jenkins/ | +
Not secure | 34.220.71.250:8080/job/BuildJob1/5/console
Dashboard > BuildJob1 > #5
--> Using cache
--> 518ea47305b4
Step 12/16 : COPY user-seed.conf /opt/splunkforwarder/etc/system/local
--> Using cache
--> ef6308cc3a02
Step 13/16 : RUN mkdir -p /var/log/webapp
--> Using cache
--> 5d8de54bfbef
Step 14/16 : COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf
--> Using cache
--> 6bd415d10024
Step 15/16 : COPY *.war /usr/local/tomcat/webapps/
--> Using cache
--> 89b1fec0ff47
Step 16/16 : ENTRYPOINT ["/usr/bin/supervisord"]
--> Using cache
--> 1e45098f021
Successfully built 1e745098f021
Successfully tagged cloudacademydevops/webapp:latest
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

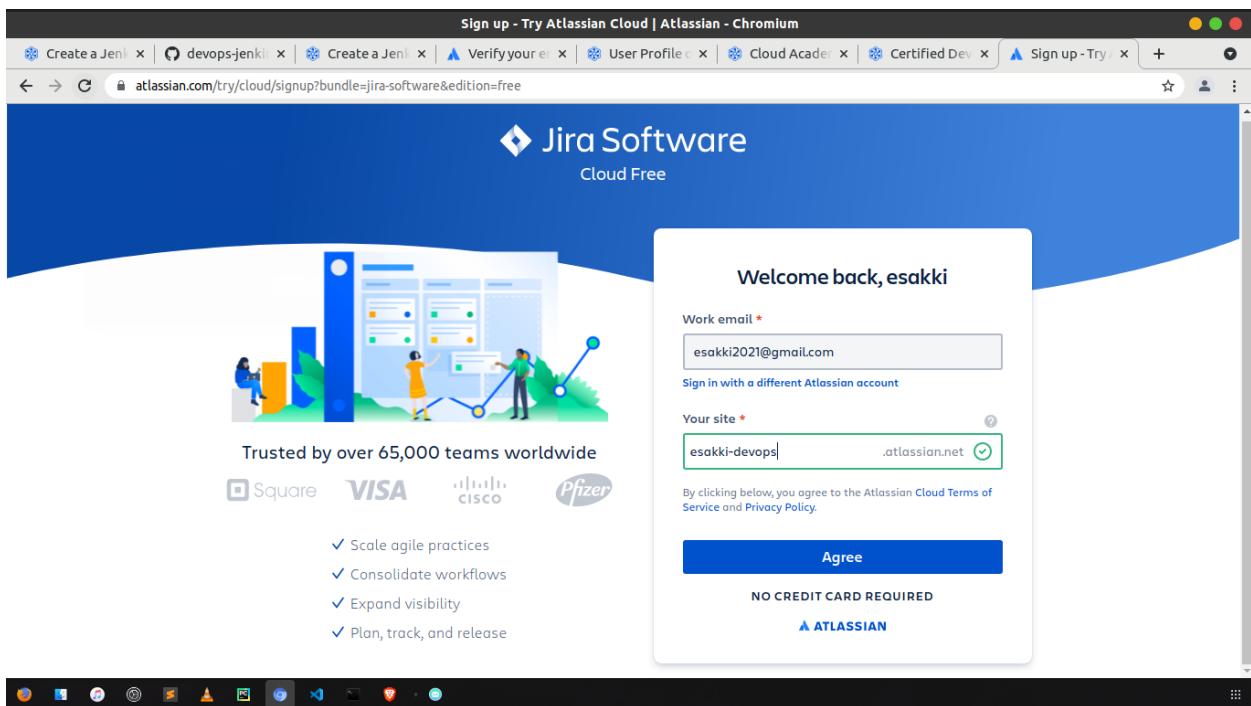
```

REST API Jenkins 2.289.2

Integrating Jenkins and Jira

Integrating Jenkins and Jira together provides you with a solution that can be used to report issues as they happen for any CICD pipeline build job. Jenkins can be configured to publish any and all build results, artifacts, and/or bugs directly into Jira. DevOps teams can then use Jira to project manage the required fixes necessary to get their applications into production.

Create a Atlassian Account



GitHub - cloudacademy/devops-jenkins-jira: Jenkins + Jira Integration

Jenkins + Jira Integration. Contribute to cloudacademy/devops-jenkins-jira development by creating an account on GitHub.

⌚ <https://github.com/cloudacademy/devops-jenkins-jira>

cloudacademy/devops-jenkins-jira

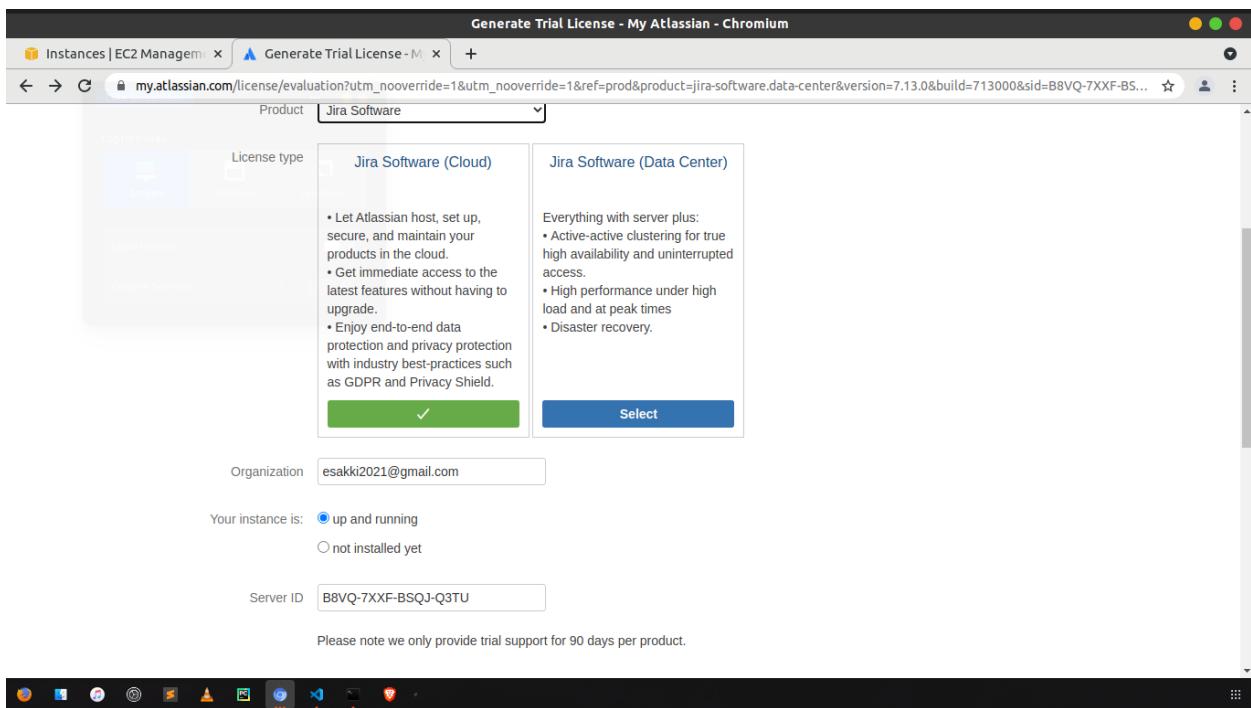
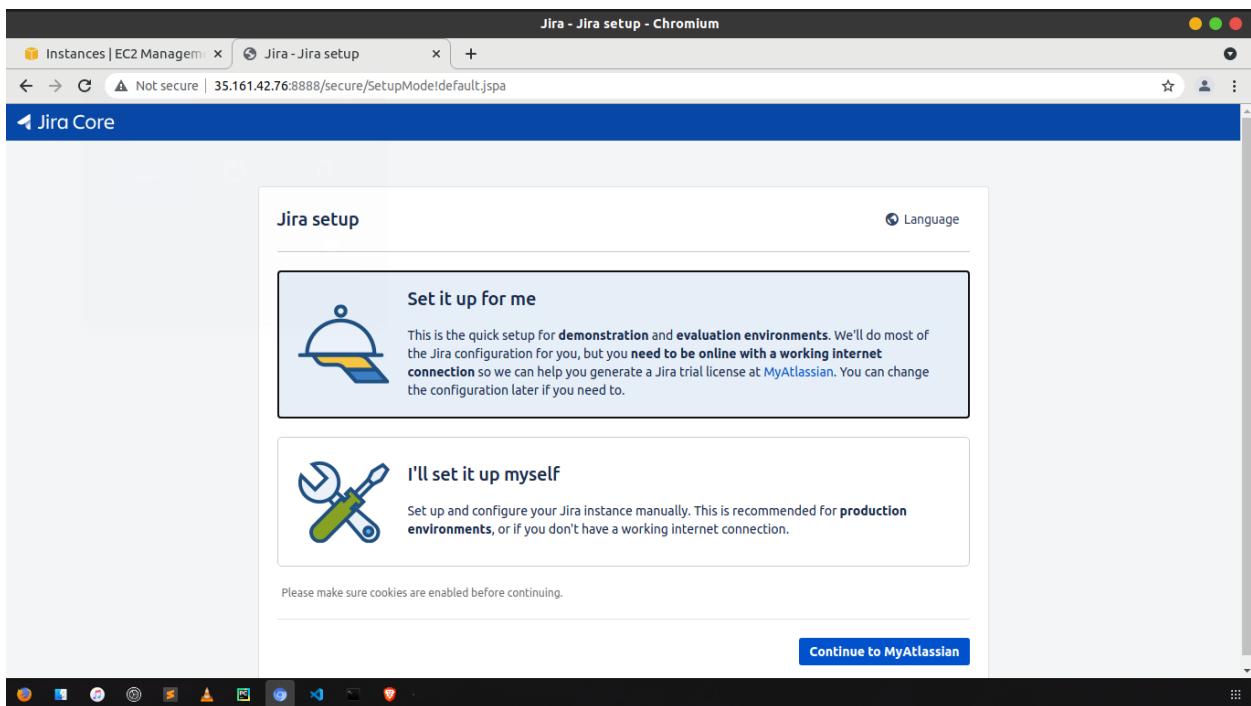
Jenkins + Jira Integration



2 Contributors 0 Issues 3 Stars 13 Forks

Bring the Docker-compose File Up

checking the Jira Installation



Once the Setup Is Completed

goto Seetings → system → Check for advanced in the left Side panel →

The screenshot shows the 'Attachment Settings' page in Jira. The left sidebar has 'Attachments' selected under 'System'. The main content area is titled 'Attachments' and contains three configuration sections: 'Allow Attachments ON' (Attachment Size 1,024.00 MB), 'Enable Thumbnails ON' (Enables the creation of thumbnail images of image attachments), and 'Enable ZIP support ON' (Enables the ability for users to download all the attachments of an issue as a single ZIP file). A note at the top states: 'To add attachments to issues in a project, users must have **Create Attachments** permission in that project.'

Lets Configure Jira

Create OAUTH Token

Settings → App ⇒ OAUTHCREDENTIALS →

The screenshot shows the 'App requests' page in Jira. The left sidebar has 'App requests' selected under 'Apps'. The main content area is titled 'Manage app requests' and displays a list of apps requested by users, including the number of requests per app. It features a search bar, a dropdown for 'All requests', and a table header for 'App details', 'No. of requests', 'Date of request', 'Cost', 'Status', and 'More'. A decorative graphic of overlapping documents with icons is centered below the table.

OAuth credentials - Jira - Chromium

Instances | EC2 Manager | DAT board - Agile board | Atlassian account | Configure System | OAuth credentials | Integrate Jira Cloud

esakki-learn.atlassian.net/secure/admin/oauth-credentials

Jira Software Your work Projects Filters Dashboards People Apps Create

Search

Apps ATLASIAN MARKETPLACE Find new apps Manage apps App requests Promotions OAuth credentials BETA

Create OAuth credentials

App name* jenkins

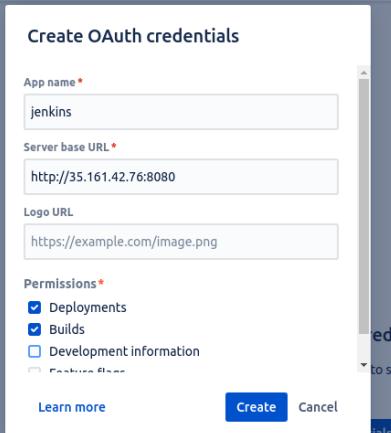
Server base URL* http://35.161.42.76:8080

Logo URL https://example.com/image.png

Permissions* Deployments Builds Development information

Learn more Create Cancel

Any issues with your OAuth credentials? Give us feedback



Install jira pipeline plugin

Available Plugins [Jenkins] - Chromium

Instances | EC2 Manager | DAT board - Agile board | Atlassian account | Available Plugins [Jenkins]

Not secure | 35.161.42.76:8080/pluginManager/available

Dashboard > Plugin Manager

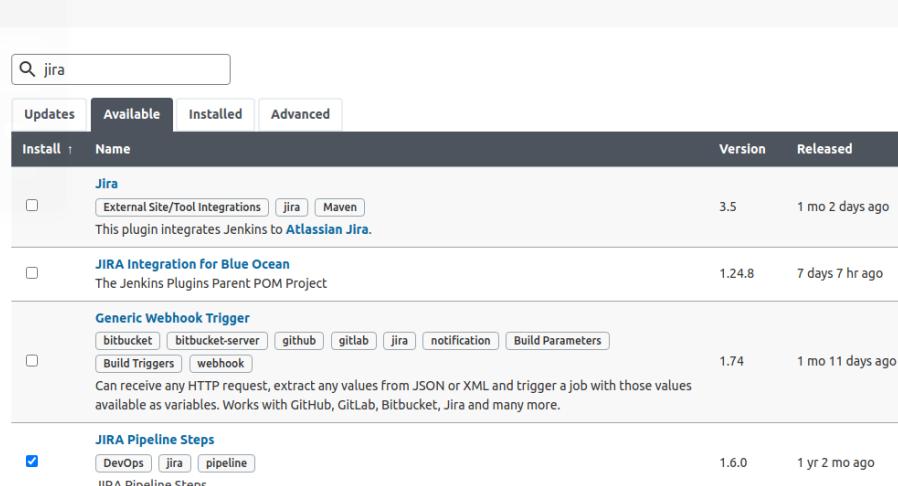
Back to Dashboard Manage Jenkins Update Center

Search: jira

Updates Available Installed Advanced

Install	Name	Version	Released
<input type="checkbox"/>	Jira External Site/Tool Integrations jira Maven This plugin integrates Jenkins to Atlassian Jira.	3.5	1 mo 2 days ago
<input type="checkbox"/>	JIRA Integration for Blue Ocean The Jenkins Plugins Parent POM Project	1.24.8	7 days 7 hr ago
<input type="checkbox"/>	Generic Webhook Trigger bitbucket bitbucket-server github gitlab jira notification Build Parameters Build Triggers webhook Can receive any HTTP request, extract any values from JSON or XML and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more.	1.74	1 mo 11 days ago
<input checked="" type="checkbox"/>	JIRA Pipeline Steps DevOps jira pipeline JIRA Pipeline Steps	1.6.0	1 yr 2 mo ago

Install without restart Download now and install after restart Update information obtained: 17 min ago Check now



Update Center [Jenkins] - Chromium

Instances | EC2 Manager | DAT board - Agile board | Atlassian account | Update Center [Jenkins] | +

← → ⌘ ⌘ Not secure | 35.161.42.76:8080/updateCenter/

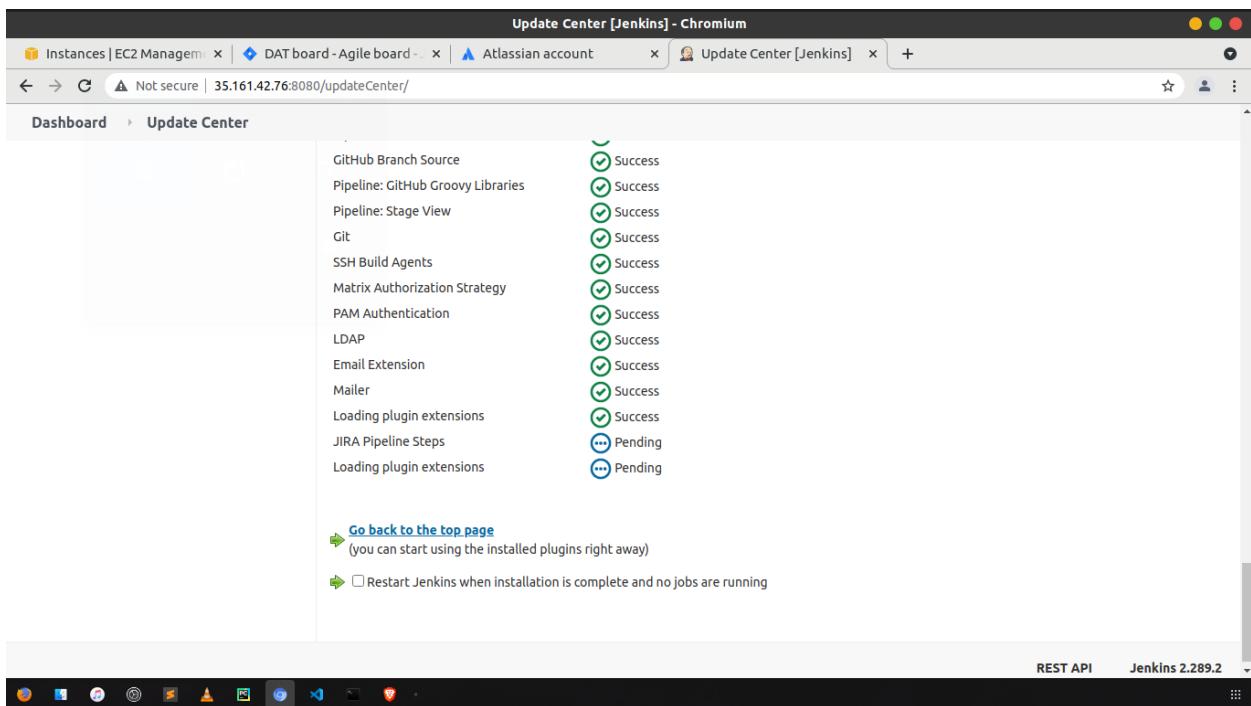
Dashboard > Update Center

Plugin	Status
GitHub Branch Source	Success
Pipeline: GitHub Groovy Libraries	Success
Pipeline: Stage View	Success
Git	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Loading plugin extensions	Success
JIRA Pipeline Steps	Pending
Loading plugin extensions	Pending

[Go back to the top page](#)
(you can start using the installed plugins right away)

Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.289.2



Lets Configure the Plugin

Configure System [Jenkins] - Chromium

Instances | EC2 Mana | DAT board - Agile bo | Atlassian account | Configure System | OAuth credentials | Integrate Jira Cloud | +

← → ⌘ ⌘ Not secure | 35.161.42.76:8080/configure

Dashboard > configuration

Jira Cloud Site

Site name ?
esakki-learn.atlassian.net

ClientID ?
6ijyTUxJYIADntpNVxq6G4UJA8tYHwl

Secret ?
jira Test settings

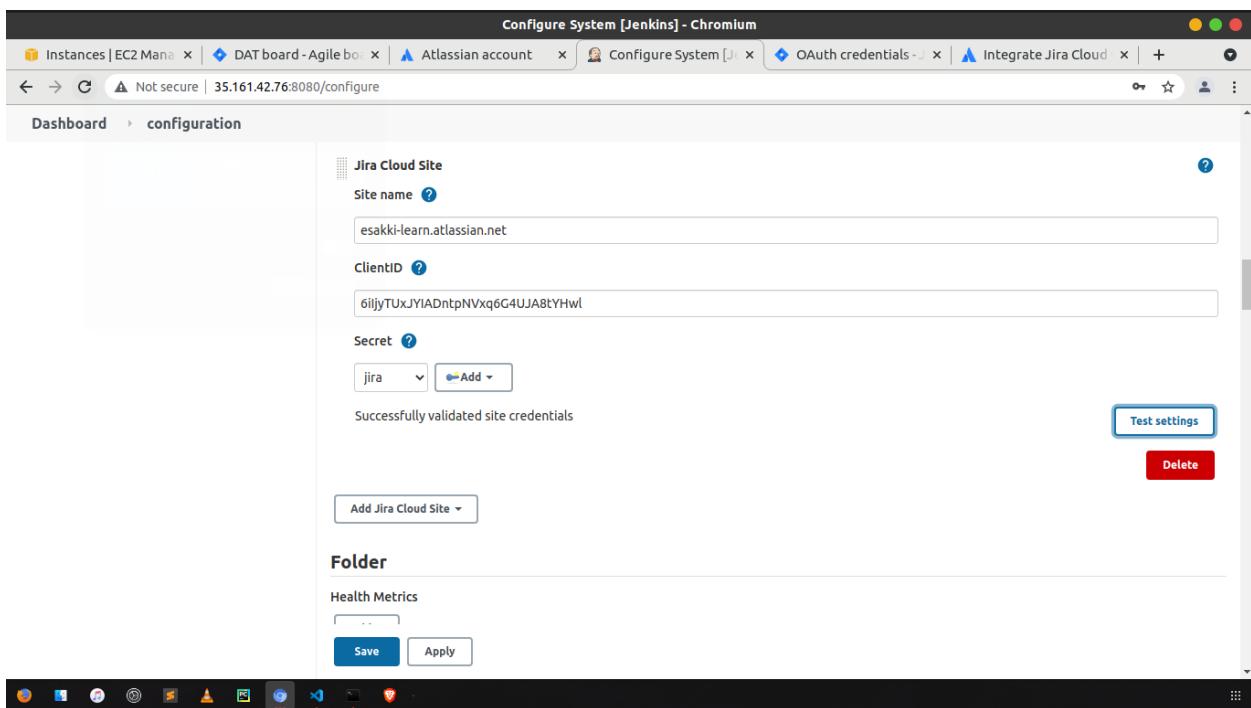
Successfully validated site credentials

Add Jira Cloud Site

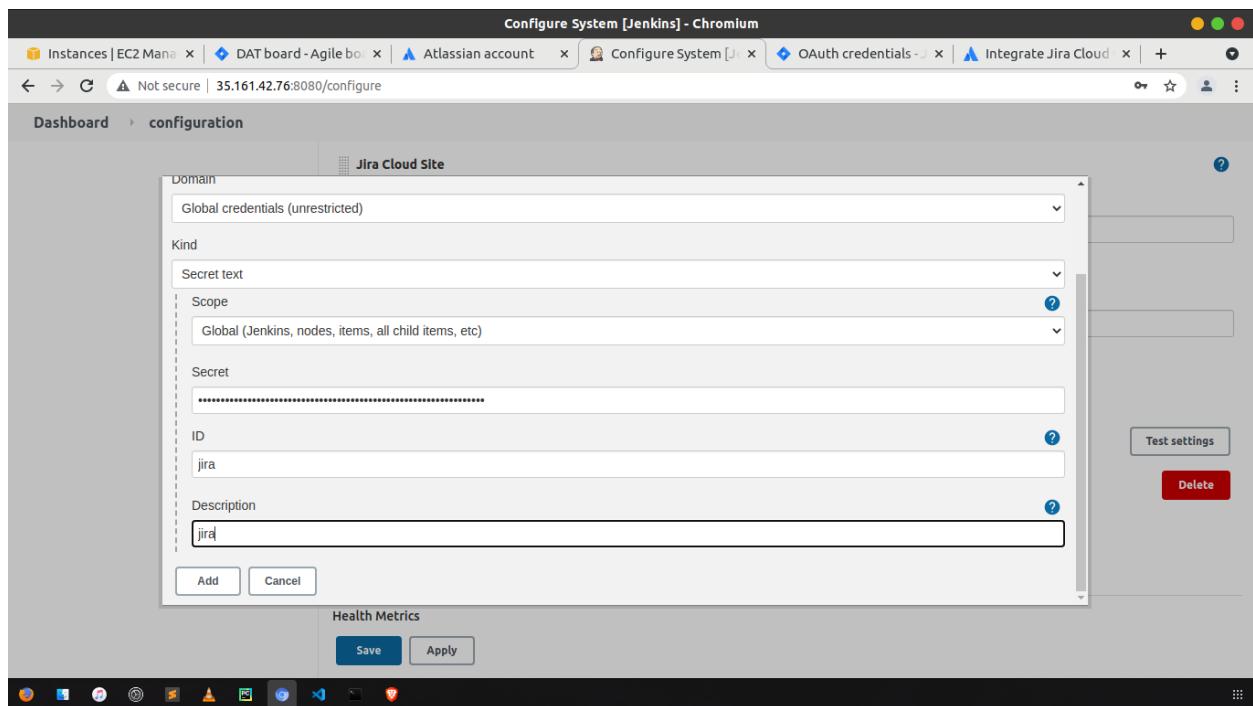
Folder

Health Metrics

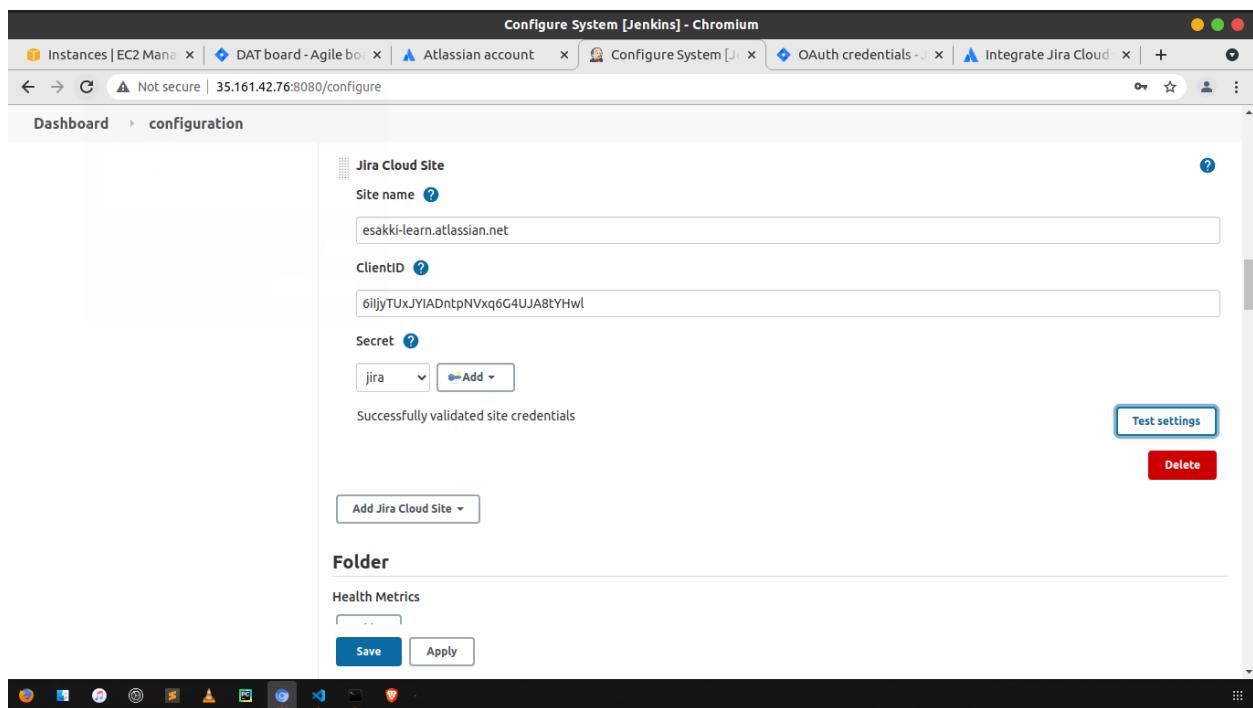
Save Apply



Add Secret



test The Connection



```
node{
def server = Artifactory.server 'artifactory'

def rtGradle = Artifactory.newGradleBuild()
```

```

withCredentials([usernamePassword(
    credentialsId: 'artifactory',
    usernameVariable: 'USERNAME',
    passwordVariable: 'PASSWORD'
)])
{
    server.username = "${USERNAME}"
    server.password = "${PASSWORD}"
}

def buildInfo

stage('clone'){
    git url: '<https://github.com/cloudacademy/devops-webapp.git>'
}

stage('Artifactory Config'){
    rtGradle.tool = "gradle-4.10.2"
    rtGradle.deployer repo:'gradle-release-local', server: server
    rtGradle.resolver repo:'jcenter', server: server
}

stage('build'){
    rtGradle.run rootDir: "./", buildFile: 'build.gradle', tasks: 'clean build'
}

stage("publish"){
    buildInfo = rtGradle.run rootDir: "./", buildFile: 'build.gradle', tasks: 'artifactoryPublish'
}

stage("deploy"){
    server.publishBuildInfo buildInfo
}

```

GitHub - cloudacademy/jenkins-cicd-adv: Jenkins CICD Advanced Demo Code and Configs
 Contains Jenkins code and configuration examples as used within the Jenkins CI/CD - Advanced course.

cloudacademy/jenkins-cicd-adv

Jenkins CICD Advanced Demo Code and Configs

🔗 <https://github.com/cloudacademy/jenkins-cicd-adv>

At 1 Contributor 0 Issues 0 Stars 5 Forks

Integration With Slack

Jenkins and Slack Integration

Slack is a chat software. Slack has hook's to provide communication from your tools to your team. Jenkins can send notification from all your jobs to your team. It can use channels to send specific notification for specific teams. First, create your

🔗 <https://medium.com/@gustavo.guss/jenkins-and-slack-integration-fbb031cd7895>



Slack

<https://techwithesak.slack.com/services/B029CAEMVTR?added=1>

```

def COLOR_MAP = [
    'SUCCESS': 'good',
    'FAILURE': 'danger',
]

def getBuildUser() {
    return currentBuild.rawBuild.getCause(Cause.UserIdCause).getUserId()
}

```

```

node {
    def GRADLE_HOME = tool name: 'gradle-4.10.2', type: 'hudson.plugins.gradle.GradleInstallation'
    def REPO_URL = 'https://github.com/cloudacademy/devops-webapp.git'

    stage('clone'){
        git url: REPO_URL
    }
    try{
        stage('build'){

            throw new Exception("Throw to stop pipeline")
            sh "${GRADLE_HOME}/bin/gradle bud --info 2>1 | tee gradle.build.${BUILD_NUMBER}.log"
            //currentBuild.result = "FAILED"
        }
    }catch(e){
        echo 'Build is failed sending Email'
        emailext (
            subject: "STARTED: Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]'",
            body: """<p>STARTED: Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]':</p>
<p>Check console output at &QUOT;<a href='${env.BUILD_URL}'>${env.JOB_NAME} [${env.BUILD_NUMBER}]</a>&QUOT;</p>""",
            recipientProviders: [[${class: 'DevelopersRecipientProvider'}, ${class: 'RequesterRecipientProvider'}]]
        )
    }

    slackSend(
        channel: '#dalilyalerts',
        color: COLOR_MAP[currentBuild.currentResult],
        message: "**${currentBuild.currentResult}:* Job ${env.JOB_NAME} build ${env.BUILD_NUMBER} by \n More info at: ${env.BUILD_URL}"
    )
}
}

```

Ansible Code to start and delete Docker in a Machine

```

---
- name: playbook to install the docker
  hosts: all
  become: true
  tasks:
    - name: Remove older version of Docker
      yum:
        name: [docker, docker-client, docker-client-latest , docker-common, docker-latest , docker-latest-logrotate, docker-logrotate, docker
        state: absent

    - name: Installing Docker Prerequisite packages
      yum:
        name: [yum-utils, device-mapper-persistent-data, lvm2, python3, python3-pip]
        state: installed

    - name: Download the Docker
      get_url:
        url: https://download.docker.com/linux/centos/docker-ce.repo
        dest: /etc/yum.repos.d/docker-ce.repo
        mode: 0644

    - name: Install docker
      yum:
        name: docker-ce
        state: present

    - name: Install docker pip packages
      pip:
        name: docker

    - name: Start the Docker
      ansible.builtin.service:
        name: docker
        enabled: yes
        state: started

```

```

- name: Add ansadmin to docker group
  ansible.builtin.shell:
    cmd: usermod -aG docker ansadmin

- name: Start the Docker
  ansible.builtin.service:
    name: docker
    enabled: yes
    state: restarted

- name: Check the Docker
  ansible.builtin.shell:
    cmd: sudo docker run -d --name myapphello esak2021/myhello:1.1.1

- name: Get Docker Info
  docker_host_info:
    containers: yes
  register: container_result

- name: Stop the Containers
  docker_container:
    name: myapp
    state: stopped
#loop: "{{ container_result.containers | map(attribute='Id') | list }}"

- name: Delete the Containers
  docker_container:
    name: myapp
    state: absent

- name: Remove Images
  docker_image:
    state: absent
    name: hello-world

- name: Start nginx Container
  docker_container:
    name: myapp
    image: nginx

```

Kick Start Terraform from jenkins

Install terraform Plugin

The screenshot shows the Jenkins plugin manager interface. A search bar at the top contains the text "terrafo". Below it, there are four tabs: "Updates", "Available", "Installed" (which is selected), and "Advanced". Under the "Installed" tab, a table lists the installed plugin. The first row is for the "Terraform" plugin, which is version 1.0.10 and was released 1 year 5 months ago. It has a checked checkbox under "Install". Below the table are two buttons: "Install without restart" and "Download now and install after restart". A status message says "Update information obtained: 3 days 23 hr ago" and a "Check now" button.

The screenshot shows the Jenkins configuration page for the Terraform plugin. At the top, there's a header "Terraform". Below it, a sub-header "Terraform installations". A "Add Terraform" button is visible. A table lists the installed Terraform instance. The first row is for "Terraform", which is version 1.0.10 and was released 1 year 5 months ago. It has a checked checkbox under "Install". The "Name" field is set to "terraform-11". The "Install directory" field is set to "/usr/bin". A checkbox "Install automatically" is unchecked. At the bottom right is a red "Delete Terraform" button.

The screenshot shows the Jenkins plugin manager interface. A search bar at the top contains the text "aws steps". Below it, there are four tabs: "Updates", "Available", "Installed" (selected), and "Advanced". Under the "Installed" tab, a table lists the installed plugin. The first row is for the "Pipeline: AWS Steps" plugin, which is version 1.43 and was released 8 months 20 days ago. It has a checked checkbox under "Install". Below the table are two buttons: "Install without restart" and "Download now and install after restart". A status message says "Update information obtained: 4 days 2 hr ago" and a "Check now" button.

Lets Create a new Jenkins Job

```

pipeline{
    agent any

    parameters{
        choice(name: 'environment', choices: 'dev\\nqa', description: 'Workspace/environment file to use for deployment')
        string(name: 'version', defaultValue: '', description: 'Version variable to pass to Terraform')
        booleanParam(name: 'autoApprove', defaultValue: false, description: 'Automatically run apply after generating plan?')
    }
    environment {
        myinfraaws = credentials('awscredentials')
    }
}

stages{

stage("checkout"){
    steps{
        checkout([
            $class: 'GitSCM',
            branches: [[name: "origin/main"]],
            userRemoteConfigs:[[
                url: "https://github.com/esak21/MySimpleApplication.git"
            ]]
        ])
    }
}
stage("plan"){
    steps{
        script{
            sh (
                script: 'terraform init -input=false',
                returnStatus: true
            ) == 0
            echo "Terraform init was sucessful"
            sh """ terraform plan -input=false -out myappinfraplan"""
            sh """ terraform show -no-color myappinfraplan > myappinfraplan.txt """
        }
    }
}
stage("approval"){
    when {
        not{
            equals expected:true, actual: params.autoApprove
        }
    }
    steps{
        script {
            def plan = readFile 'myappinfraplan.txt'
            input message: "Do you want to apply the plan?", parameters: [text(name: 'Plan', description: 'Please review the plan', defaultValue: plan)]
        }
    }
}
stage('Apply') {
    steps {
        script {
            sh (script:"terraform apply -input=false myappinfraplan ", returnStatus: true) == 0
            echo "Terraform Apply was sucessful"
        }
    }
}
stage("destroyit"){
    steps{
        script {
            def userInput = input(id: 'confirm', message: 'destroy AWS Resources?', parameters: [ [$class: 'BooleanParameterDefinition'
            sh(script: "terraform destroy --auto-approve", returnStatus: true) == 0
            echo "AWS resources were destroyed sucessfully"
        }]
    }
}
}

```

```
}
```

```
post {
```

```
    always {
```

```
        archiveArtifacts artifacts: 'myappinfraplan.txt'
```

```
    }
```

```
}
```

```
}
```

Jenkinsfile for running Terraform

Instantly share code, notes, and snippets. Jenkinsfile for running Terraform You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to <https://gist.github.com/gazoakley/87dcc16d28fd05acda4ba0a4be5ac387>

