DEVOPS TOOL INTEGRATION ON A SIMPLE JAVA CALCULATOR PROGRAM

Submitted By:
Aditya Tiwari (MT2019011)

Submitted To:
Prof. B Thangaraju

Github Repository Link
https://github.com/Aditya-tiwari17/CalculatorDevops.git

Docker Hub Repository link
https://hub.docker.com/repository/docker/adityat17/calculatordevops

TOOLS AND TECHNOLOGIES USED

1. Git and GitHub

Git is a distributed version control tool that can manage a development project's source code history, while GitHub is a cloud-based platform built around the Git tool.

2. JUnit

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

3. Jenkins

Jenkins is a free and open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase and RTC, and can execute Apache Ant, Apache Maven and sbt based projects as well as arbitrary shell scripts and Windows batch commands.

4.Maven

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. Maven addresses two aspects of building software: how software is built, and its dependencies. Unlike earlier tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging.

5.Docker

Docker is a set of platform as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers.[6] Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.[7] All containers are run by a single operating system kernel and therefore use fewer resources than virtual machines.

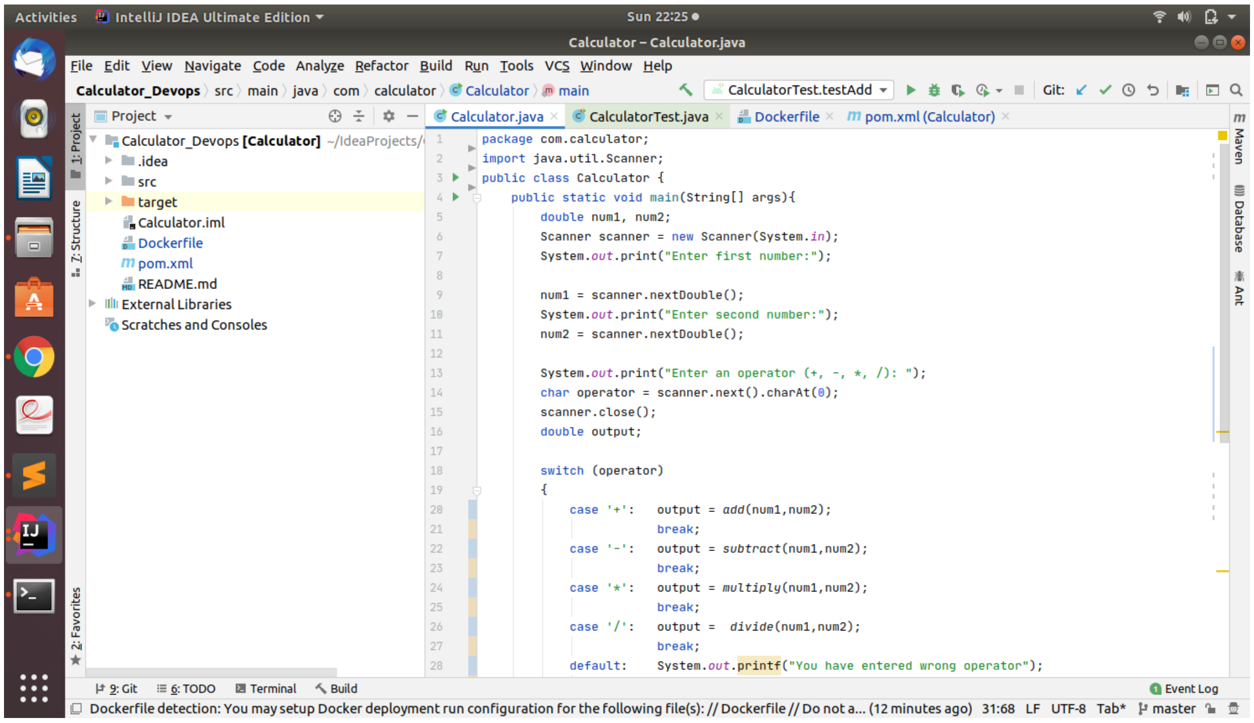
6.Rundeck

Rundeck is an open source automation service with a web console, command line tools and a WebAPI. It lets you easily run automation tasks across a set of nodes.

VARIOUS CODE FILES

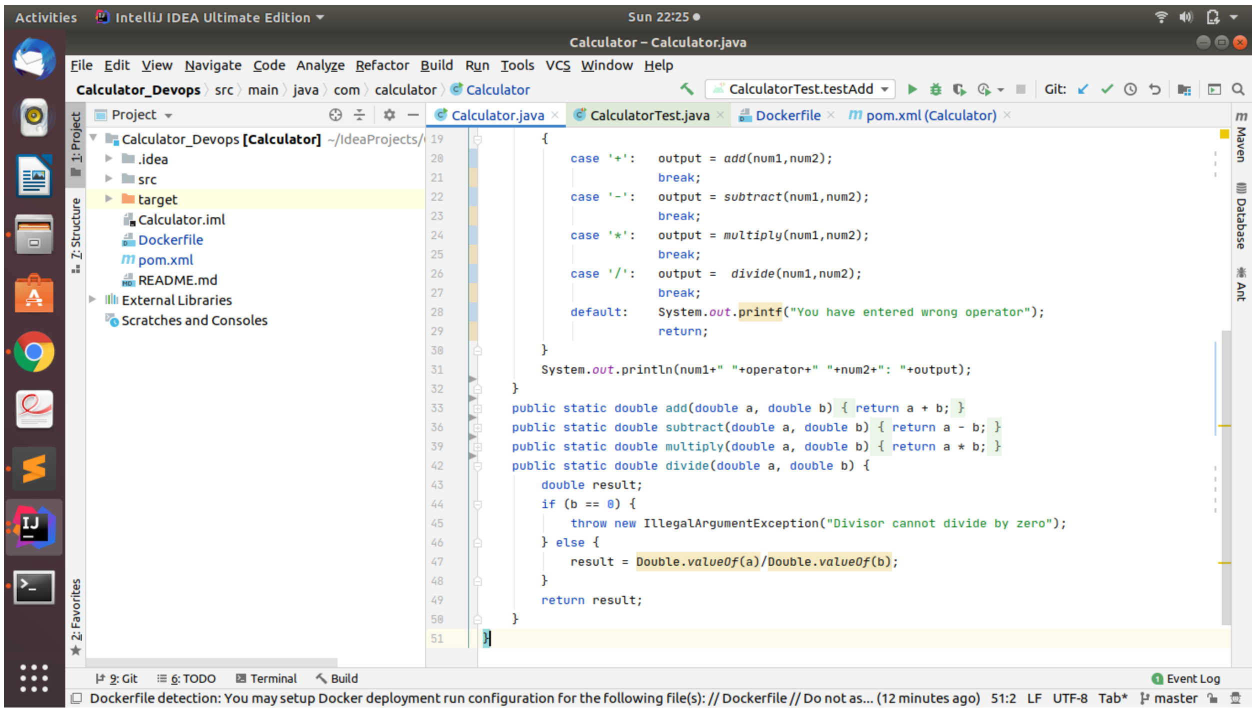
-

- **Program Source Code**



```java
package com.calculator;
import java.util.Scanner;
public class Calculator {
    public static void main(String[] args){
        double num1, num2;
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number:");

        num1 = scanner.nextDouble();
        System.out.print("Enter second number:");
        num2 = scanner.nextDouble();

        System.out.print("Enter an operator (+, -, *, /): ");
        char operator = scanner.next().charAt(0);
        scanner.close();
        double output;

        switch (operator)
        {
            case '+':   output = add(num1,num2);
                        break;
            case '-':   output = subtract(num1,num2);
                        break;
            case '*':   output = multiply(num1,num2);
                        break;
            case '/':   output =  divide(num1,num2);
                        break;
            default:    System.out.printf("You have entered wrong operator");
```

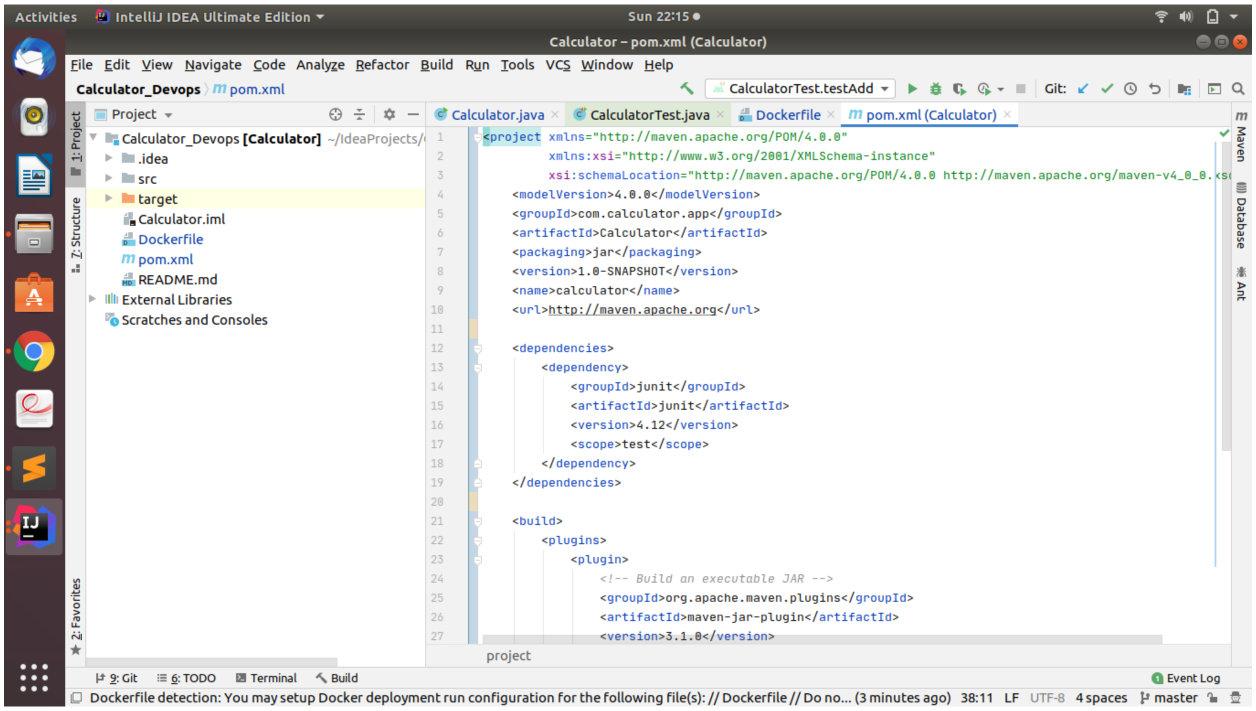
Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do not a... (12 minutes ago)



```java
        {
            case '+':   output = add(num1,num2);
                        break;
            case '-':   output = subtract(num1,num2);
                        break;
            case '*':   output = multiply(num1,num2);
                        break;
            case '/':   output =  divide(num1,num2);
                        break;
            default:    System.out.printf("You have entered wrong operator");
                        return;
        }
        System.out.println(num1+" "+operator+" "+num2+": "+output);
    }
    public static double add(double a, double b) { return a + b; }
    public static double subtract(double a, double b) { return a - b; }
    public static double multiply(double a, double b) { return a * b; }
    public static double divide(double a, double b) {
        double result;
        if (b == 0) {
            throw new IllegalArgumentException("Divisor cannot divide by zero");
        } else {
            result = Double.valueOf(a)/Double.valueOf(b);
        }
        return result;
    }
}
```

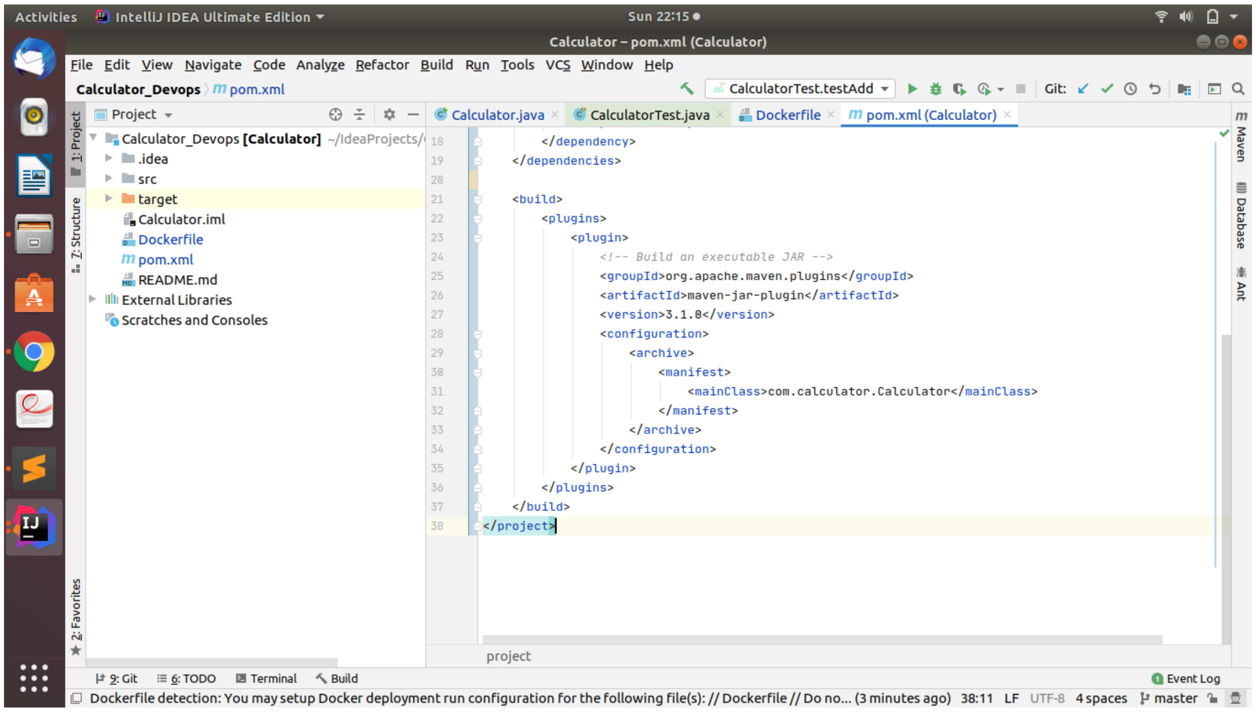
Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do not as... (12 minutes ago)

- **POM.XML**



```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xs
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.calculator.app</groupId>
    <artifactId>Calculator</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>calculator</name>
    <url>http://maven.apache.org</url>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <!-- Build an executable JAR -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.1.0</version>
```

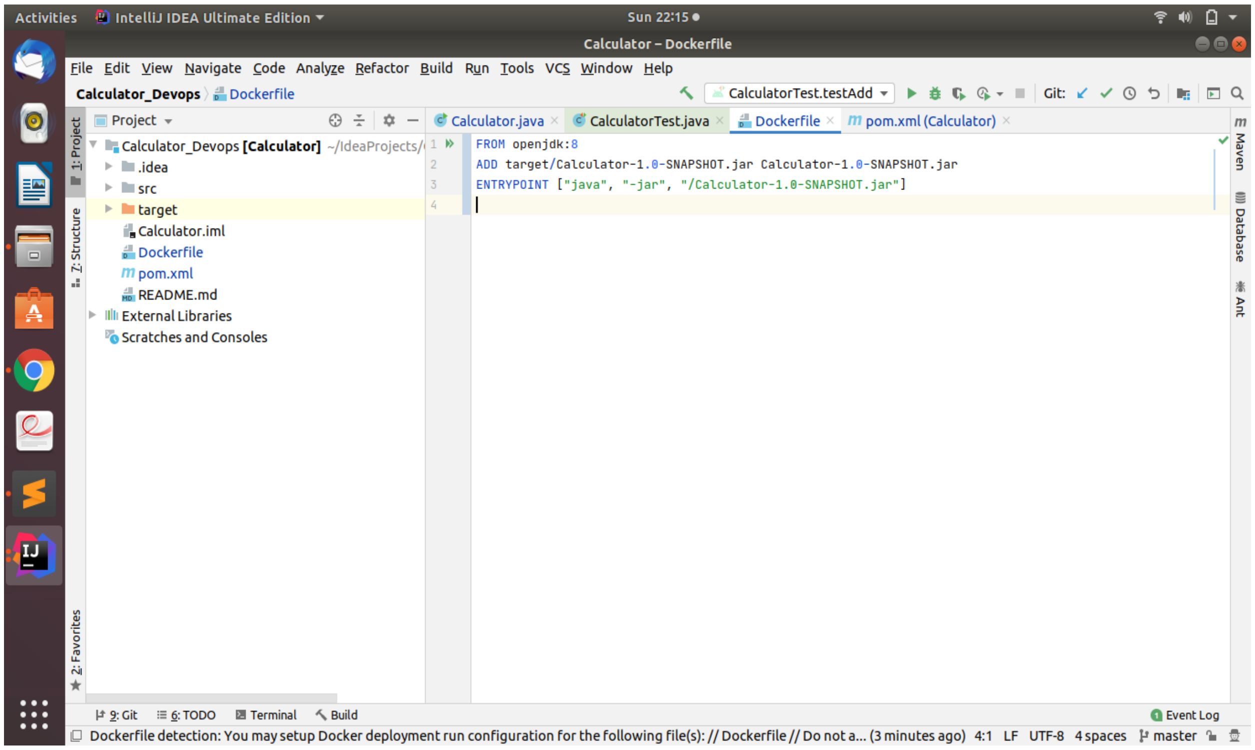
Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do no... (3 minutes ago)



```
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <!-- Build an executable JAR -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.1.0</version>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>com.calculator.Calculator</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do no... (3 minutes ago)

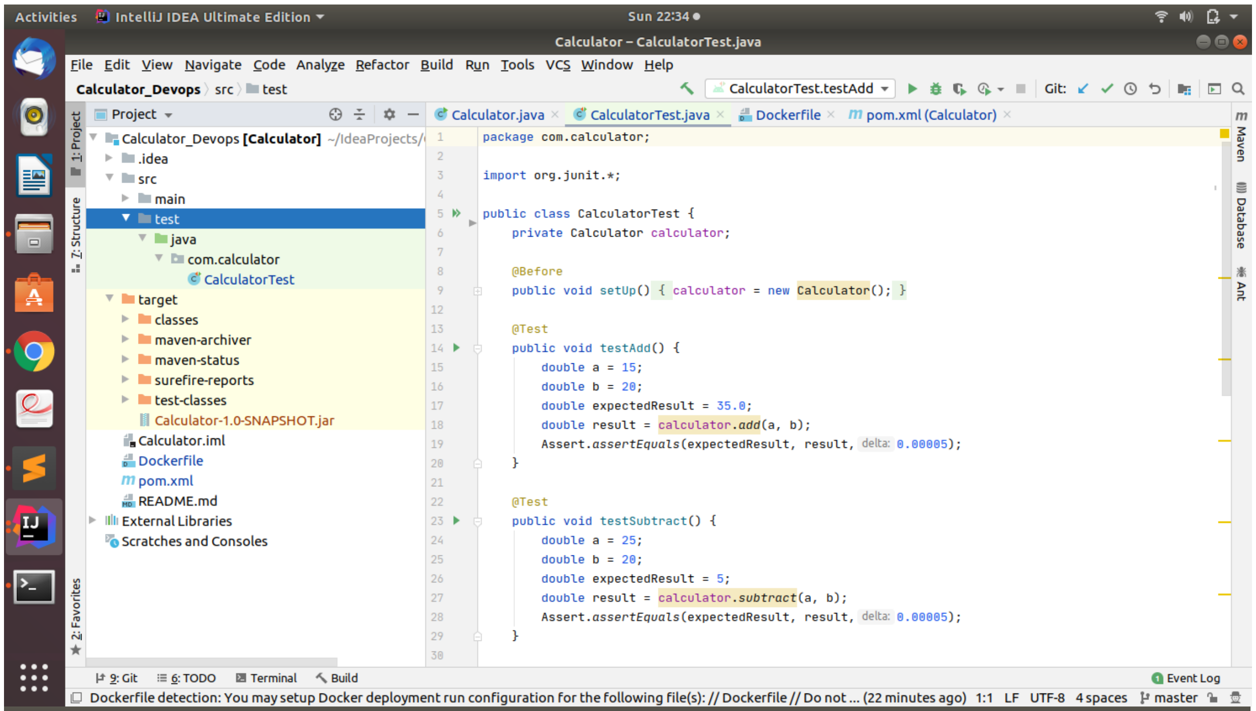
- **Dockerfile**



```
FROM openjdk:8
ADD target/Calculator-1.0-SNAPSHOT.jar Calculator-1.0-SNAPSHOT.jar
ENTRYPOINT ["java", "-jar", "/Calculator-1.0-SNAPSHOT.jar"]
```

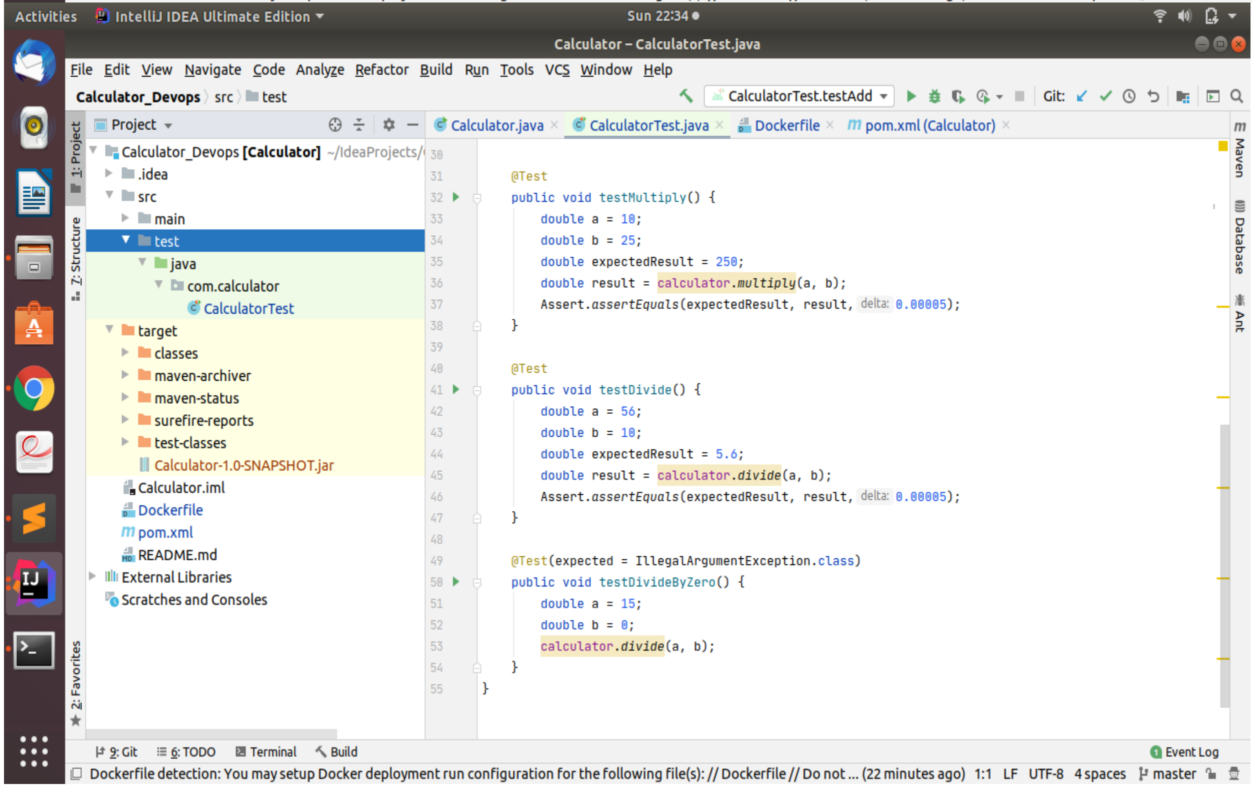
Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do not a... (3 minutes ago)

- **JUnit Test File**



```
package com.calculator;

import org.junit.*;

public class CalculatorTest {
    private Calculator calculator;

    @Before
    public void setUp() { calculator = new Calculator(); }

    @Test
    public void testAdd() {
        double a = 15;
        double b = 20;
        double expectedResult = 35.0;
        double result = calculator.add(a, b);
        Assert.assertEquals(expectedResult, result, delta: 0.00005);
    }

    @Test
    public void testSubtract() {
        double a = 25;
        double b = 20;
        double expectedResult = 5;
        double result = calculator.subtract(a, b);
        Assert.assertEquals(expectedResult, result, delta: 0.00005);
    }

```

Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do not ... (22 minutes ago) 1:1 LF UTF-8 4 spaces master

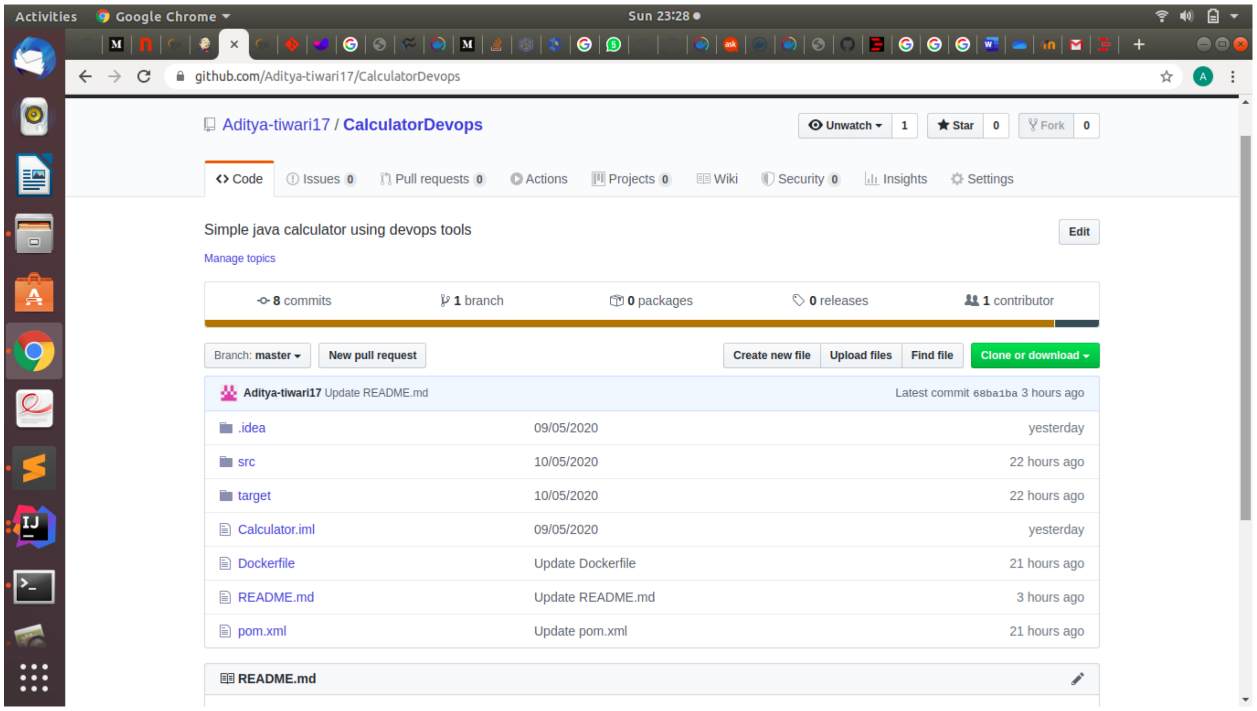


```

    @Test
    public void testMultiply() {
        double a = 10;
        double b = 25;
        double expectedResult = 250;
        double result = calculator.multiply(a, b);
        Assert.assertEquals(expectedResult, result, delta: 0.00005);
    }

    @Test
    public void testDivide() {
        double a = 56;
        double b = 10;
        double expectedResult = 5.6;
        double result = calculator.divide(a, b);
        Assert.assertEquals(expectedResult, result, delta: 0.00005);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
        double a = 15;
        double b = 0;
        calculator.divide(a, b);
    }
}
```

Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do not ... (22 minutes ago) 1:1 LF UTF-8 4 spaces master

STEPS INVOLVED

1. Create Github Repoitory and Push Project files



Aditya-tiwari17 / **CalculatorDevops**

Simple java calculator using devops tools

8 commits | **1** branch | **0** packages | **0** releases | **1** contributor

Aditya-tiwari17 Update README.md — Latest commit 68ba1ba 3 hours ago

.idea	09/05/2020	yesterday
src	10/05/2020	22 hours ago
target	10/05/2020	22 hours ago
Calculator.iml	09/05/2020	yesterday
Dockerfile	Update Dockerfile	21 hours ago
README.md	Update README.md	3 hours ago
pom.xml	Update pom.xml	21 hours ago

README.md

2. Install plugin in Jenkins

Jenkins -> Plugin Manager -> Available

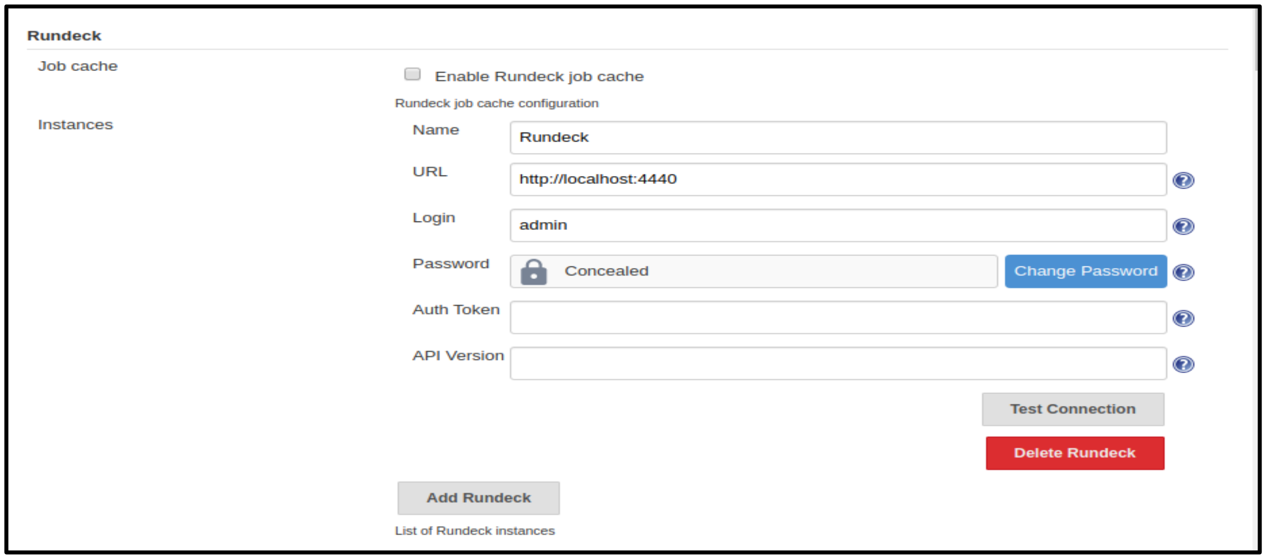
Then search and download following plugins:

- Git
- GitHub plugin
- Unleash Maven plugin
- Docker plugin
- Pipeline
- Rundeck

3. System Configuration

Jenkins -> Manage Jenkins -> Configure System

- **Rundeck**

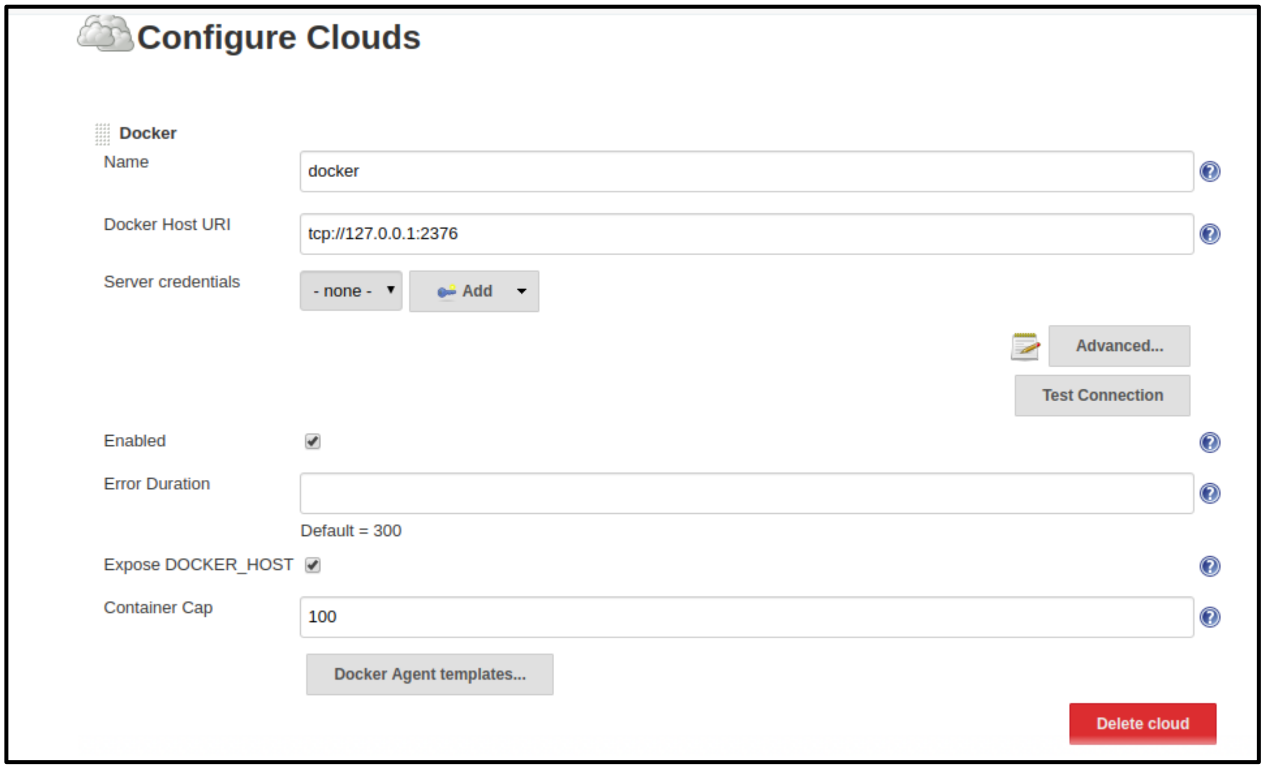


Rundeck

Job cache	☐ Enable Rundeck job cache
	Rundeck job cache configuration
Instances	
Name	Rundeck
URL	http://localhost:4440
Login	admin
Password	Concealed
Auth Token	
API Version	

List of Rundeck instances

- **Cloud**



Configure Clouds

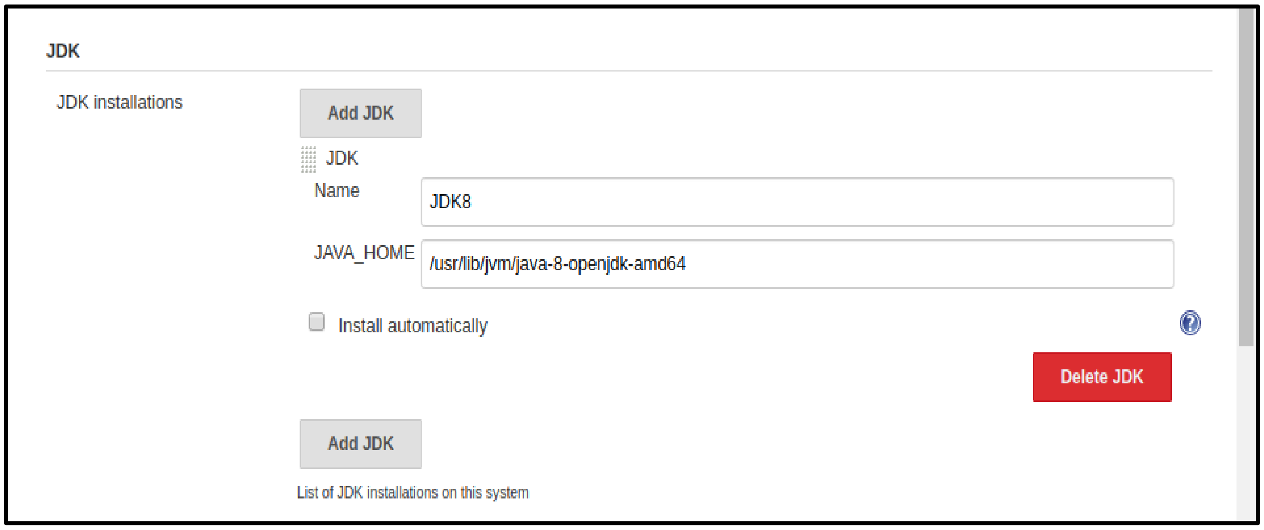
Docker

Name	docker
Docker Host URI	tcp://127.0.0.1:2376
Server credentials	- none -
Enabled	✓
Error Duration	
	Default = 300
Expose DOCKER_HOST	✓
Container Cap	100

4. Global Tool Configuration

Jenkins -> Manage Jenkins -> Global Tool Configuration

- **JDK**



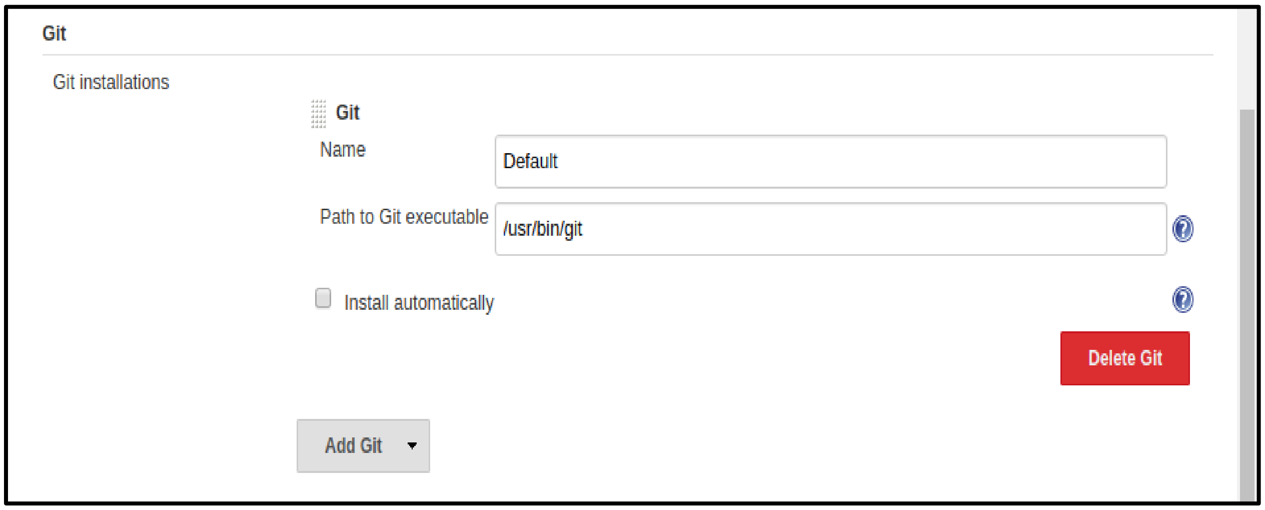
JDK

JDK installations

JDK
Name: JDK8
JAVA_HOME: /usr/lib/jvm/java-8-openjdk-amd64
Install automatically: [ ]

List of JDK installations on this system

- **Git**

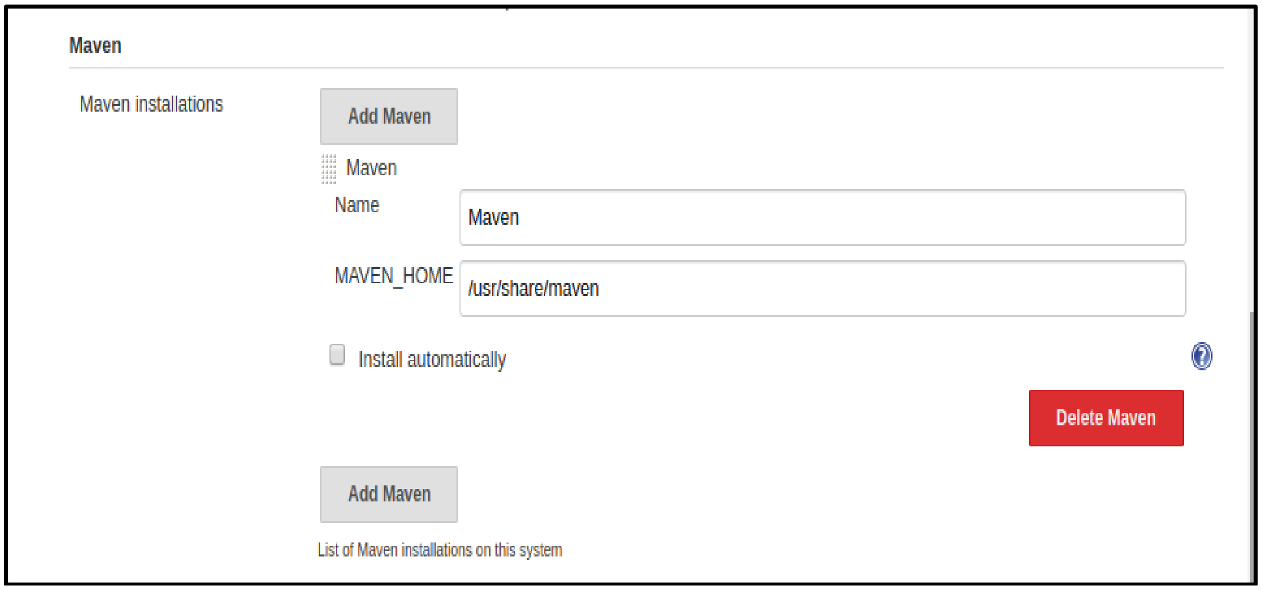


Git

Git installations

Git
Name: Default
Path to Git executable: /usr/bin/git
Install automatically: [ ]

- **Maven**



Maven

Maven installations

Add Maven

Maven

Name: Maven

MAVEN_HOME: /usr/share/maven

☐ Install automatically

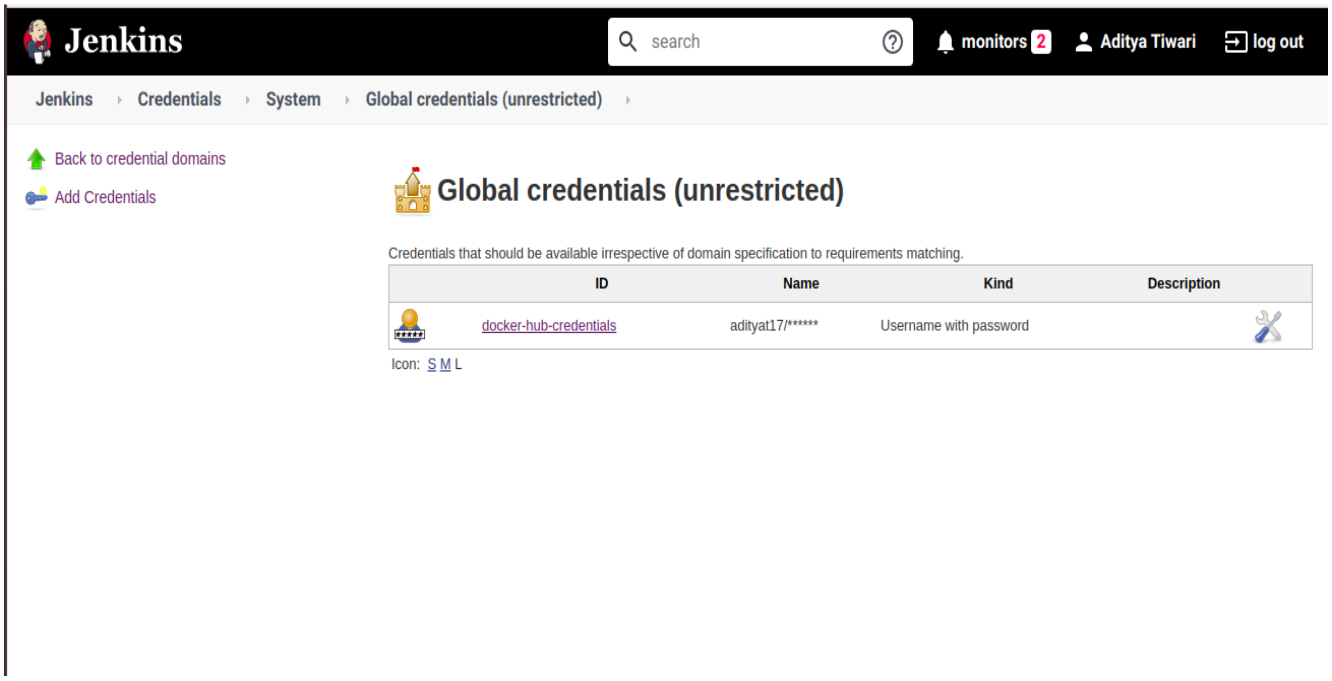
Delete Maven

Add Maven

List of Maven installations on this system

5. Setting Docker Credentials in Jenkins

Jenkin -> Credentials -> System -> Global Credentials -> Provide username and password of DockerHub -> Provide an ID for these credentials as *docker-hub-credentials.*



Jenkins | search | monitors 2 | Aditya Tiwari | log out

Jenkins › Credentials › System › Global credentials (unrestricted) ›

Back to credential domains

Add Credentials

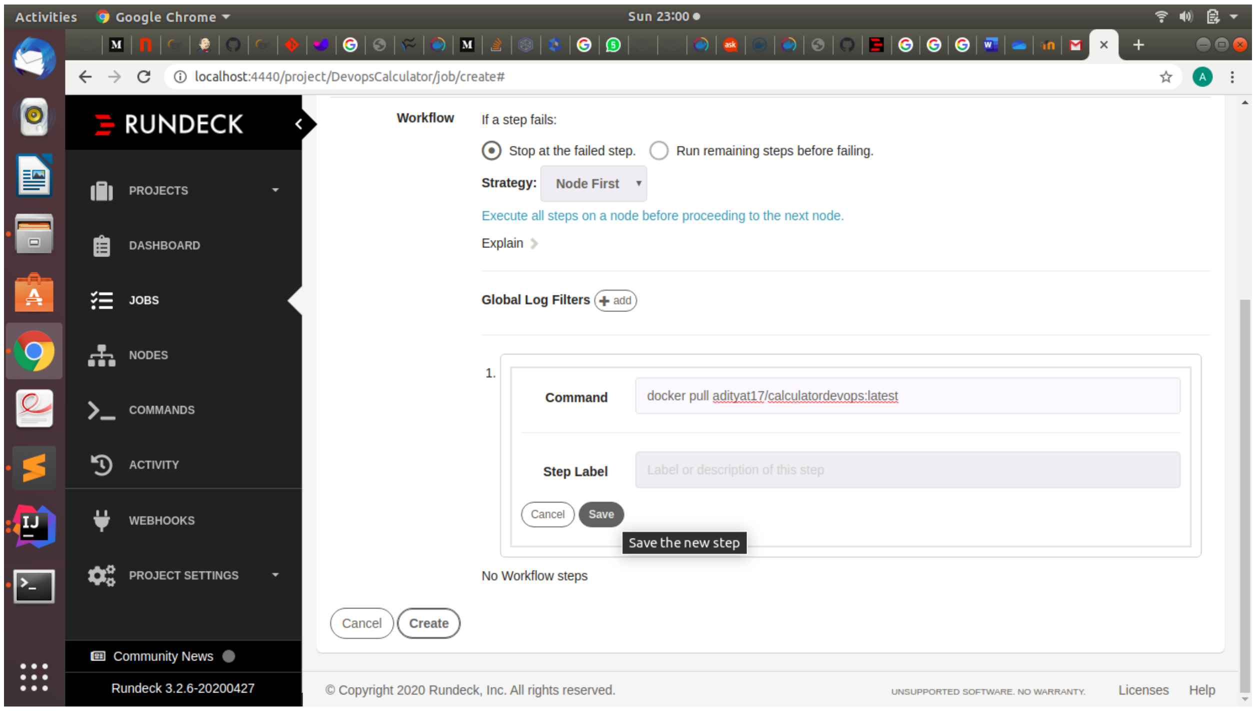
Global credentials (unrestricted)

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

ID	Name	Kind	Description
docker-hub-credentials	adityat17/******	Username with password	

Icon: S M L

6. Create Rundeck Job



localhost:4440/project/DevopsCalculator/job/create#

Workflow

If a step fails: [✓ Stop at the failed step.] Run remaining steps before failing.

Strategy: Node First

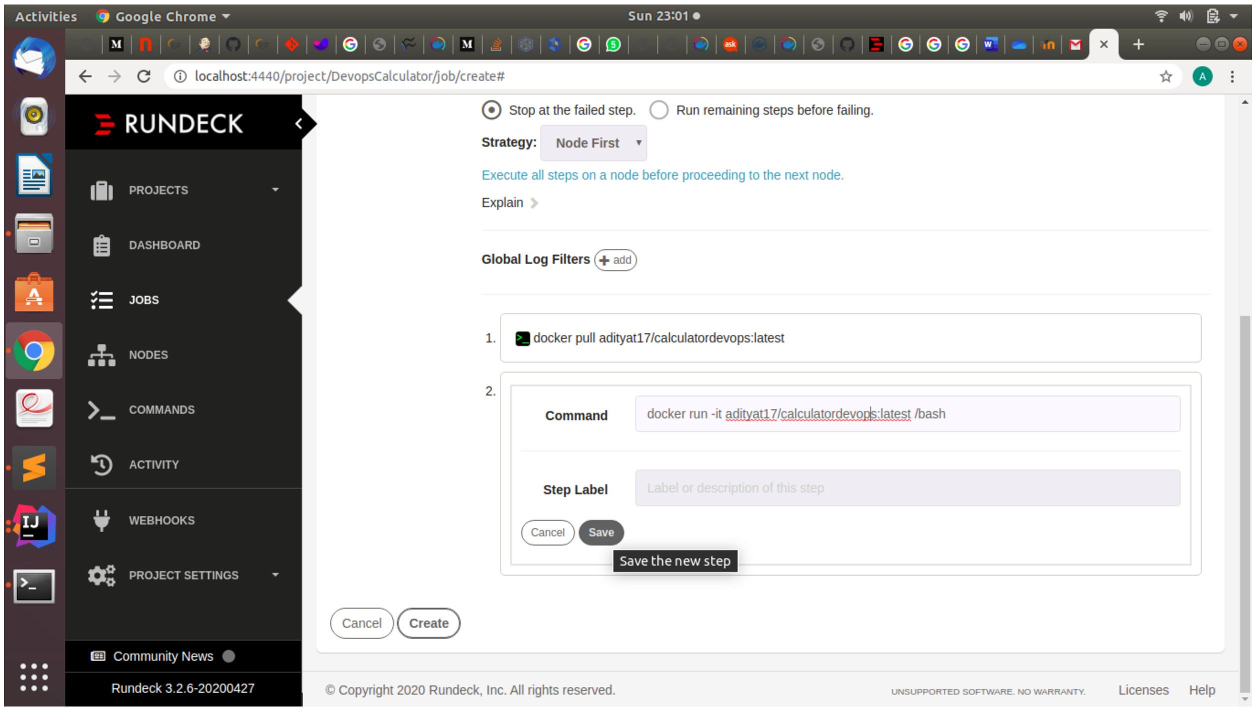
Execute all steps on a node before proceeding to the next node.

Global Log Filters

1. Command: `docker pull adityat17/calculatordevops:latest`

Step Label: Label or description of this step

No Workflow steps



localhost:4440/project/DevopsCalculator/job/create#

[✓ Stop at the failed step.] Run remaining steps before failing.

Strategy: Node First

Execute all steps on a node before proceeding to the next node.

Global Log Filters

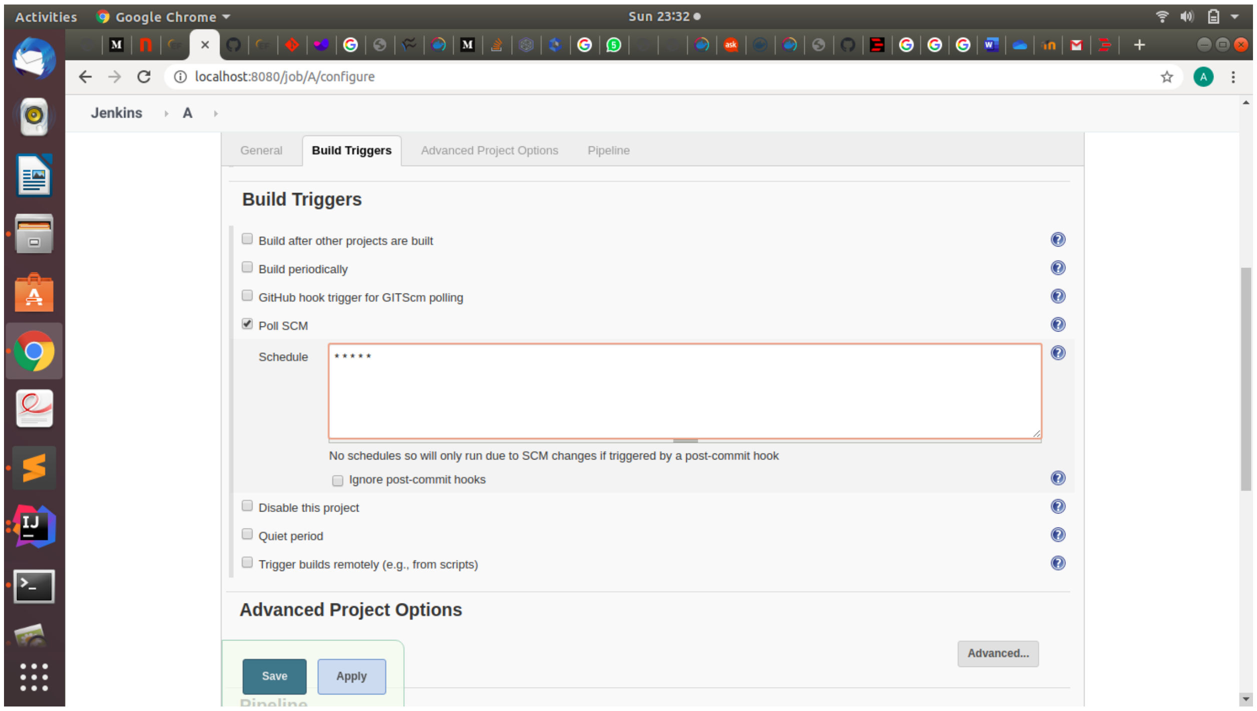
1. `docker pull adityat17/calculatordevops:latest`

2. Command: `docker run -it adityat17/calculatordevops:latest /bash`

Step Label: Label or description of this step

7. Create Jenkins Pipeline

Jenkins -> New Item -> Pipeline



Activities Google Chrome Sun 23:32

localhost:8080/job/A/configure

Jenkins › A ›

General | Build Triggers | Advanced Project Options | Pipeline

Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [x] Poll SCM

Schedule

```
* * * * *
```

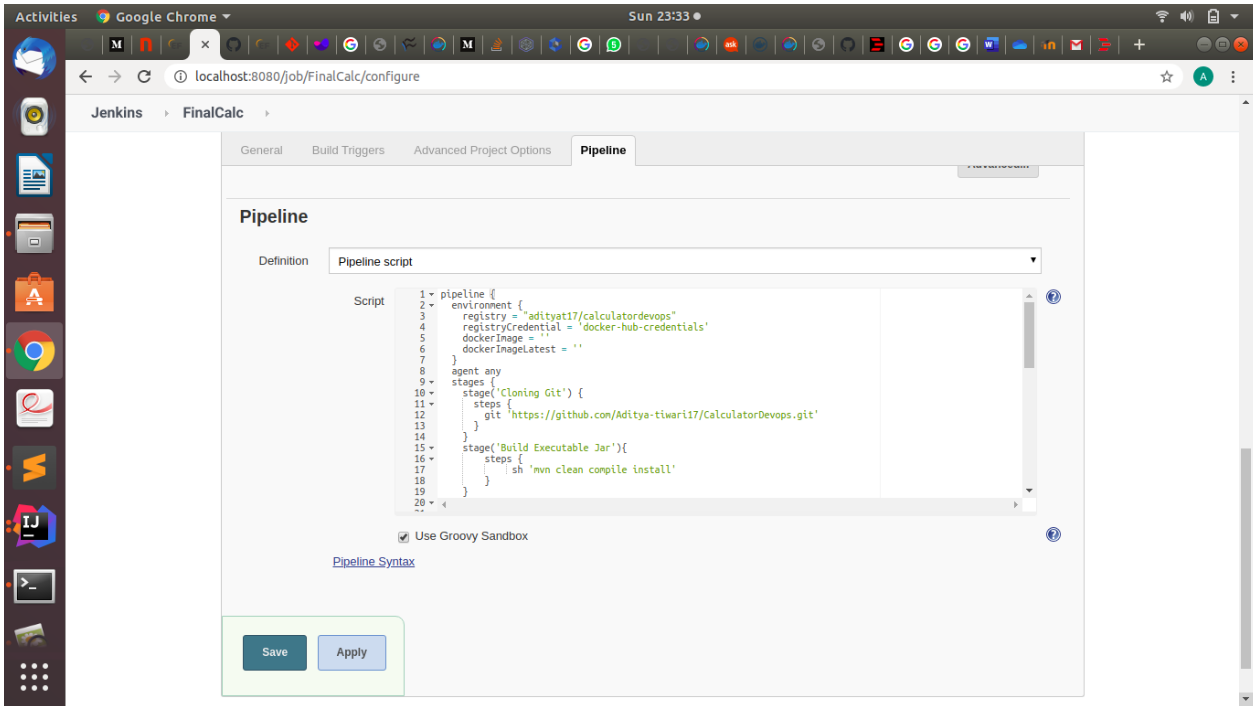
No schedules so will only run due to SCM changes if triggered by a post-commit hook

- [ ] Ignore post-commit hooks
- [ ] Disable this project
- [ ] Quiet period
- [ ] Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Advanced...

Save | Apply



Activities Google Chrome Sun 23:33

localhost:8080/job/FinalCalc/configure

Jenkins › FinalCalc ›

General | Build Triggers | Advanced Project Options | Pipeline

Pipeline

Definition: Pipeline script

Script

```
pipeline {
  environment {
    registry = "adityat17/calculatordevops"
    registryCredential = 'docker-hub-credentials'
    dockerImage = ''
    dockerImageLatest = ''
  }
  agent any
  stages {
    stage('Cloning Git') {
      steps {
        git 'https://github.com/Aditya-tiwari17/CalculatorDevops.git'
      }
    }
    stage('Build Executable Jar'){
        steps {
            sh 'mvn clean compile install'
        }
    }
```

- [x] Use Groovy Sandbox

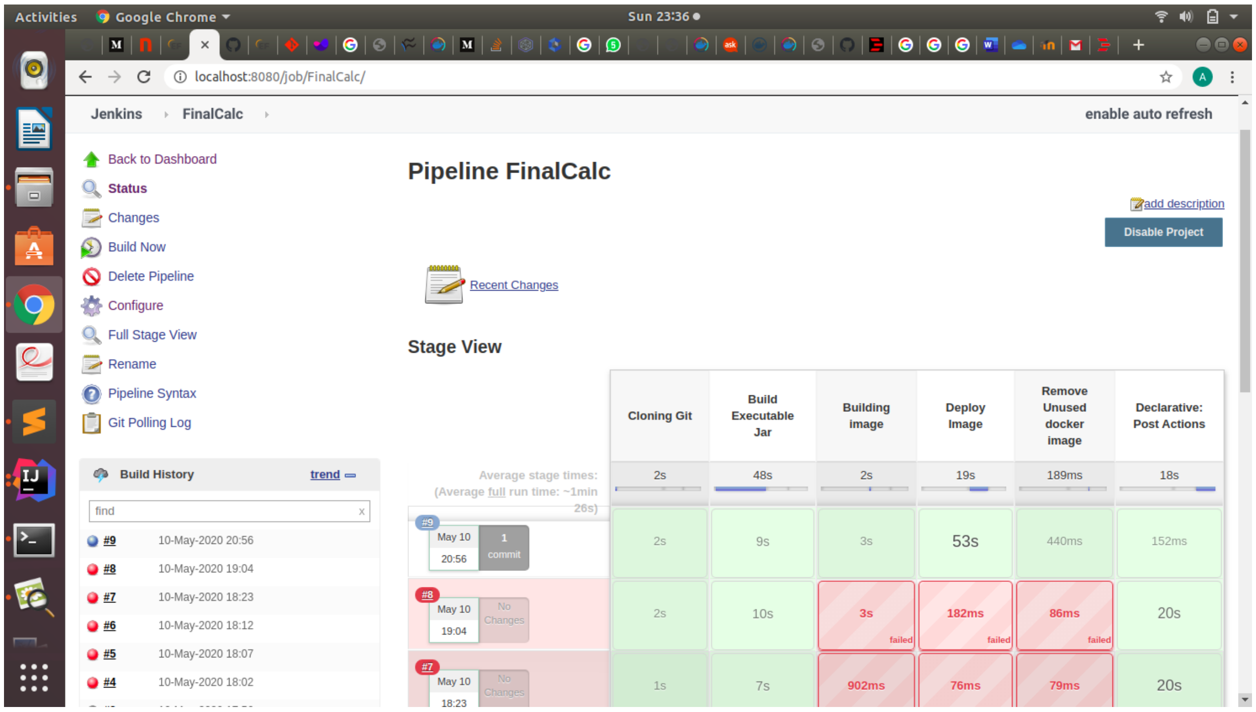
Pipeline Syntax

Save | Apply

Pipeline Script

```
pipeline {
 environment {
  registry = "adityat17/calculatordevops"
  registryCredential = 'docker-hub-credentials'
  dockerImage = ''
  dockerImageLatest = ''
 }
 agent any
 stages {
  stage('Cloning Git') {
   steps {
    git 'https://github.com/Aditya-tiwari17/CalculatorDevops.git'
   }
  }
  stage('Build Executable Jar'){
    steps {
       sh 'mvn clean compile install'
    }
  }
  stage('Building image') {
   steps{
    script {
     dockerImage = docker.build registry + ":$BUILD_NUMBER"
     dockerImageLatest = docker.build registry + ":latest"
    }
   }
  }
  stage('Deploy Image') {
   steps{
    script {
     docker.withRegistry( '', registryCredential ) {
      dockerImage.push()
      dockerImageLatest.push()
     }
    }
   }
  }
  stage('Remove Unused docker image') {
      steps{
        sh "docker rmi $registry:$BUILD_NUMBER"
      }
    }

  }
  post {
      always{
        emailext body: "Dear Sir/Mam, <br/><br/>
${currentBuild.currentResult}: Job ${env.JOB_NAME} build
${env.BUILD_NUMBER}  <br/>  More info at: ${env.BUILD_URL}  <br/><br/>
THIS IS A SYSTEM GENERATED EMAIL. PLEASE DO NOT REPLY.",
                recipientProviders: [[$class: 'DevelopersRecipientProvider'], [$class:
'RequesterRecipientProvider']],
                subject: "Jenkins Build ${currentBuild.currentResult}: Job
${env.JOB_NAME}"

        }
    }
}
```

8. Build Pipeline and all your hardwork create beautiful green images on screen



Pipeline FinalCalc

Stage View

	Cloning Git	Build Executable Jar	Building image	Deploy Image	Remove Unused docker image	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 26s)	2s	48s	2s	19s	189ms	18s
#9 May 10 20:56 1 commit	2s	9s	3s	53s	440ms	152ms
#8 May 10 19:04 No Changes	2s	10s	3s failed	182ms failed	86ms failed	20s
#7 May 10 18:23 No Changes	1s	7s	902ms	76ms	79ms	20s

#9	10-May-2020 20:56
#8	10-May-2020 19:04
#7	10-May-2020 18:23
#6	10-May-2020 18:12
#5	10-May-2020 18:07
#4	10-May-2020 18:02