

# **DevOps Project**

## **MoneyMetrics – The Financial Calculator**

**Name : Dhayanidhi S**

**Reg no : 23BIT0214**

**Email : dhayaaisro09@gmail.com**

### **Project Statement :**

**'Create a complete DevOps CI/CD Pipeline with Jenkins for a Use Case (different from the Calculator application) by using GIT, Maven, Puppet/Ansible/Terraform, Docker, JUnit, Graphite, and Grafana. You can add more tools if you want.'**

### **Overview :**

This project automates the complete software delivery lifecycle—including build, test, dockerization, deployment, monitoring, and rollback—for a CLI-based Java Financial Calculator. The pipeline is triggered automatically using SCM polling in Jenkins, which checks for new commits every 2 minutes and initiates a build only if changes are detected. Jenkins orchestrates the entire process, while Ansible executes modular roles defined in a centralized playbook to carry out tasks such as Maven-based compilation, JUnit testing, Docker image creation, and deployment to a Kubernetes (Minikube) cluster. Real-time monitoring is implemented using Collectd (for data collection), Graphite (for time-series storage), and Grafana (for visualization dashboards). Additionally, an automatic rollback mechanism is built into the pipeline to revert to the last successful version if any stage fails. This tightly integrated setup ensures fast, reliable, and fully automated software delivery with minimal manual intervention.

---

 **Table of Contents**

---

S.No.	Topic	Page No.
1	About Project and Its Components	3
2	Use Case	7
3	Project	
3.1	Creating Project Repository	8
3.2	Setting Up Maven Project	9
3.3	Docker Setup and Dockerization	16
3.4	JUnit Testing	18
3.5	Deploying on Kubernetes with Minikube	20
3.6	Monitoring Using Grafana and Graphite	25
3.7	Ansible Orchestration and Automation	28
3.8	CI/CD Using Jenkins	32
3.8.1	Freestyle Project	34
3.8.2	Running Only Ansible Playbook in Jenkins	38
3.8.3	Pipeline with Ansible Integration	40
3.9	Git Polling and Build Trigger Demonstration	44
3.10	Updating Playbook and Roles for Pipeline Functionality	45
3.11	Rollback Implementation	46
3.12	Migration from Local Git Repository to GitHub	48
4	Conclusion	49
5	Codes ( Github Link )	49

---

# Project Components

## 1. Java Financial Calculator

- A CLI-based application developed in Java.
- Accepts user input via terminal.
- Computes interest-related financial values.

## 2. JUnit Test Cases

- Unit tests verify all financial functions.
- Test failure is **intentionally triggered** to simulate rollback.

## 3. Maven Build & Test

- Jenkins runs

```
mvn clean package
```
  - If any JUnit test fails, the pipeline does **not stop immediately**. Instead, Ansible marks a flag for rollback.
- 

## Docker Structure

- The Dockerfile builds a Java app image:

Dockerfile

- Two image tags maintained:
    - latest
    - previous (used for rollback)
- 

## Jenkins CI/CD Pipeline Stages

### 1. SCM Checkout

- Pulls the latest code from your Git repo (local path or GitHub).
- SCM Polling or Webhook triggers pipeline automatically on code push.

### 2. Build

- Compiles code using mvn package.

### 3. Test

- Runs JUnit tests.
- If any test fails, Ansible creates a rollback trigger file.

#### 4. Dockerize

- Copies Dockerfile to workspace.
- Builds a Docker image tagged with build number (e.g., financial-calc:6) and latest.
- Tags previous image as previous for rollback.

#### 5. Deploy

- Removes any running financial-calc container.
- Runs the new image.
- Waits for 5 seconds to allow startup.

#### 6. Monitor

- Starts Grafana + Graphite stack using Docker Compose.
- Visualization of CPU, memory, app logs, etc.

#### 7. Rollback (if needed)

- If rollback\_trigger file is detected, rollback role is executed.
- Removes broken container.
- Replaces with image tagged as previous.

---

## Ansible Roles

The project has modular Ansible roles:

Role	Purpose
build	Maven build and packaging
test	Run JUnit tests and create rollback trigger on failure
dockerize	Build Docker image and tag it
deploy	Remove old container and deploy new one
monitor	Start Grafana and Graphite with Docker Compose
rollback	Check for test failure and perform image rollback

All roles are included via site.yml master playbook with tag-based execution.

---

## Monitoring Stack

Tools:

- **Collectd** – Gathers metrics.

- **Graphite** – Time-series database.
- **Grafana** – Visual dashboards.

#### Setup:

- Runs via Docker Compose (monitor/docker-compose.yml).
  - Started through Ansible during the monitor role.
- 

#### Rollback Workflow

Condition	Action
Tests and Build pass	Continue with deployment
Tests or Build fail	- Create rollback_trigger file - Rollback role triggers
Rollback	Remove new image run previously working image

Rollback is fully automated and **resilient**.

---

#### Permissions & Ownership

- Git repo is made a safe directory using:  

```
git config --global --add safe.directory /path/to/repo
```
  - Ownership is managed so both jenkins and dhayanidhi-s users can access files and Docker socket.
- 

#### SCM Polling vs Webhook

- SCM Polling is configured with \* \* \* \* \* to check every minute.
  - You can eliminate polling delay by using a **GitHub webhook** to notify Jenkins instantly on push.
- 

#### Success Criteria

- Jenkins automatically detects code push.
  - If build or test fails, rollback is successful.
  - Docker container always reflects last known working build.
  - Metrics are viewable in Grafana dashboard.
-

## Directory Structure (Simplified)

```
project_root/
  └── ansible_project/
      ├── roles/
      │   ├── build/
      │   ├── test/
      │   ├── dockerize/
      │   ├── deploy/
      │   ├── monitor/
      │   └── rollback/
      ├── inventory
      └── site.yml
  └── Dockerfile
  └── src/
      └── target/
          └── Jenkinsfile
  └── Jar File
  └── pom.xml
```

---

## Learning Outcomes

- Understand CI/CD fundamentals using Jenkins.
  - Build Ansible playbooks and roles for automation.
  - Learn Docker image creation and container deployment.
  - Automate rollback mechanisms using Ansible.
  - Integrate monitoring into a DevOps pipeline using Grafana.
  - Hands-on experience in connecting all DevOps stages into a seamless workflow.
- 

## Possible Future Enhancements

- Convert Java CLI app into a REST API using Spring Boot.
  - Use GitHub Webhooks instead of SCM polling for real-time trigger.
  - Integrate Slack/Email notifications for pipeline events.
  - Add database integration for persistent data storage.
  - Deploy in Kubernetes (Minikube or cloud-managed) for scalability
-

## USE CASE

### The Financial Calculator Project

The **Financial Calculator** is a comprehensive Java-based application designed to perform a wide range of financial computations that are commonly needed by individuals, accountants, businesses, students, and financial analysts. It serves as a powerful command-line tool that simplifies complex mathematical financial operations by accepting user inputs and providing accurate results within seconds. The calculator is built with clean, modular Java code and offers a reliable, easy-to-use interface, making it suitable for both educational purposes and real-world financial decision-making.

#### Key Functionalities (Performs 10+ Operations):

1. **Simple Interest (SI):** Calculates interest based on principal, rate, and time.
2. **Compound Interest (CI):** Computes interest on principal + accumulated interest over time.
3. **EMI (Equated Monthly Installment):** Calculates fixed monthly payment for a loan.
4. **Future Value (FV):** Determines the value of an investment after a period at a fixed interest rate.
5. **Present Value (PV):** Calculates today's value of a future amount of money.
6. **Loan Affordability Check:** Computes maximum loan based on EMI and interest rate.
7. **Savings Growth Estimation:** Predicts savings over time with periodic deposits.
8. **Investment Return Analysis:** Calculates returns from various investment scenarios.
9. **Break-even Point Estimation:** Determines when an investment will break even.
10. **Interest Rate Conversion:** Converts annual interest rates to monthly or daily and vice versa.

#### Real-World Use Cases & Benefits

This calculator is especially useful for **bank employees, loan officers, investors, financial advisors, students studying commerce or finance, and individuals managing personal finances**. It allows quick assessments of loans, savings, and investments, making budgeting and financial planning more efficient. For example, a student can estimate how much they'll owe after taking a student loan, or a home buyer can compute EMI options before approaching a bank. Businesses can evaluate investment proposals or compare loan options to select the most cost-effective one.

The Financial Calculator eliminates manual errors, saves time, and promotes financial literacy by helping users understand how financial parameters interact — making it a vital tool in day-to-day financial decision-making.

## Creating project repository

Install GIT client and server on the same host .

**ssh git@localhost** – Establishes a secure SSH connection to the local Git server for performing Git operations.

**git init --bare project\_repository.git** – Creates a bare repository to act as the central remote for code versioning and CI/CD integration.

**Bare repositories** store only version history and are ideal for collaborative or automated workflows (no working directory).

**git clone git@localhost:project\_repository.git** – Clones the bare repo into a working directory for development or Jenkins builds.

This setup is essential for pushing code, polling for changes, and triggering the DevOps pipeline automatically

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ ssh git@localhost
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.11.0-26-generic x86_64)

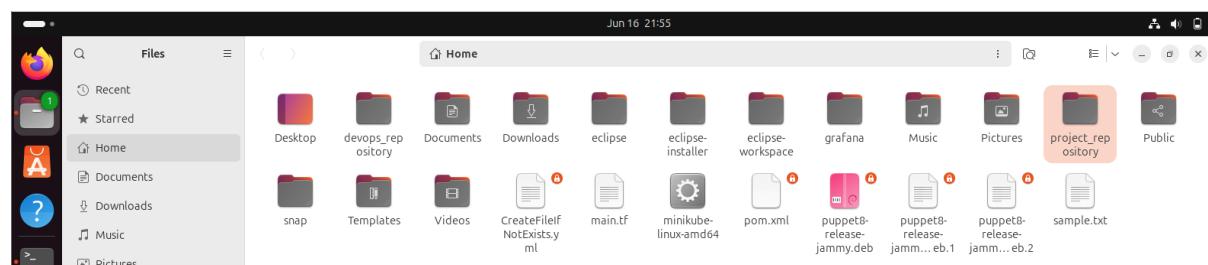
 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

12 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Mon Jun 16 21:50:28 2025 from 127.0.0.1
git@dhyanidhi-s-VirtualBox:~$ git init --bare project_repository.git
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/git/project_repository.git/
git@dhyanidhi-s-VirtualBox:~$ exit
logout
Connection to localhost closed.
dhayanidhi-s@dhyanidhi-s-VirtualBox:~$ git clone git@admin:devops_repository.git
dhayanidhi-s@dhyanidhi-s-VirtualBox:~$ git clone git@localhost:project_repository.git
Cloning into 'project_repository'...
warning: You appear to have cloned an empty repository.
dhayanidhi-s@dhyanidhi-s-VirtualBox:~$
```



Project\_repository created .

# Setting Up Maven Project

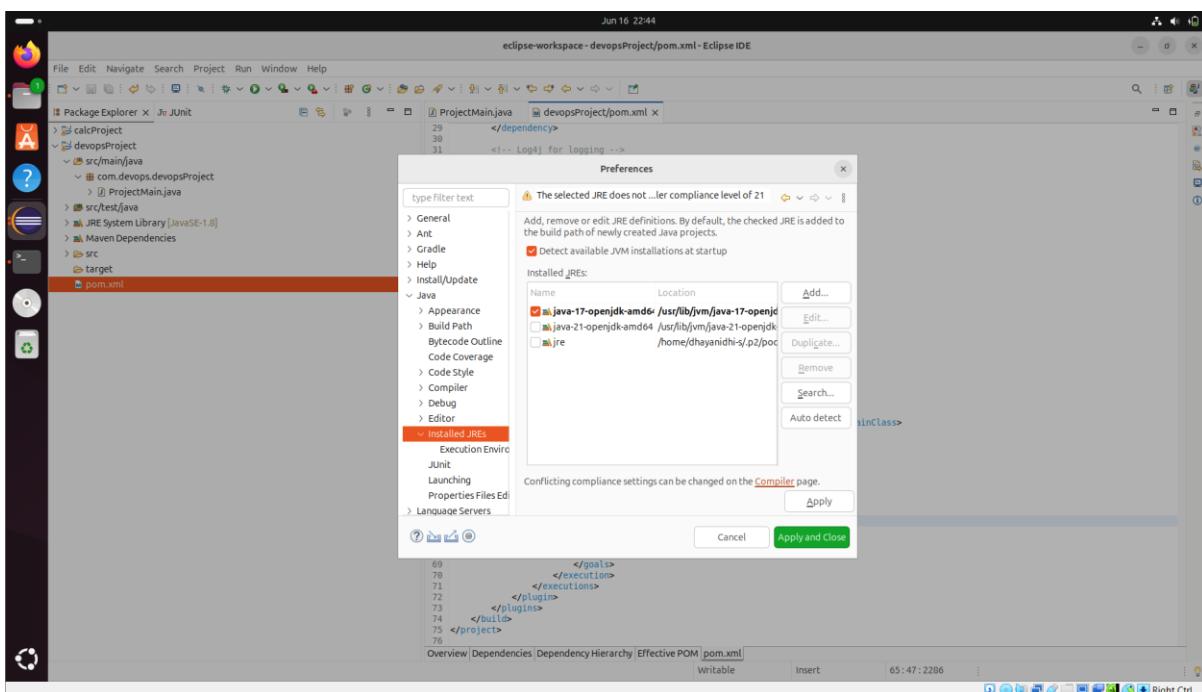
Creating a modular Maven project that will later be used in the CI/CD pipeline.

## 1. Installed Java JDK (Prerequisite for Eclipse)

## 2. Downloaded and Installed Eclipse IDE

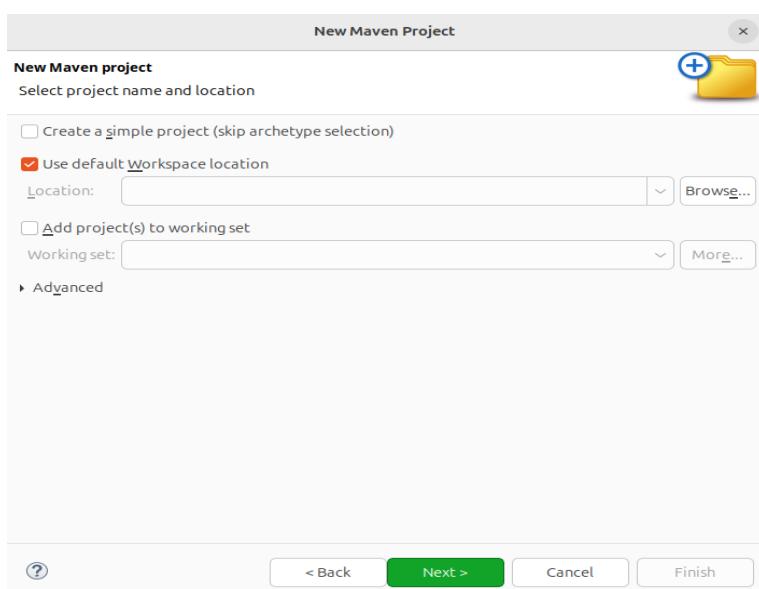
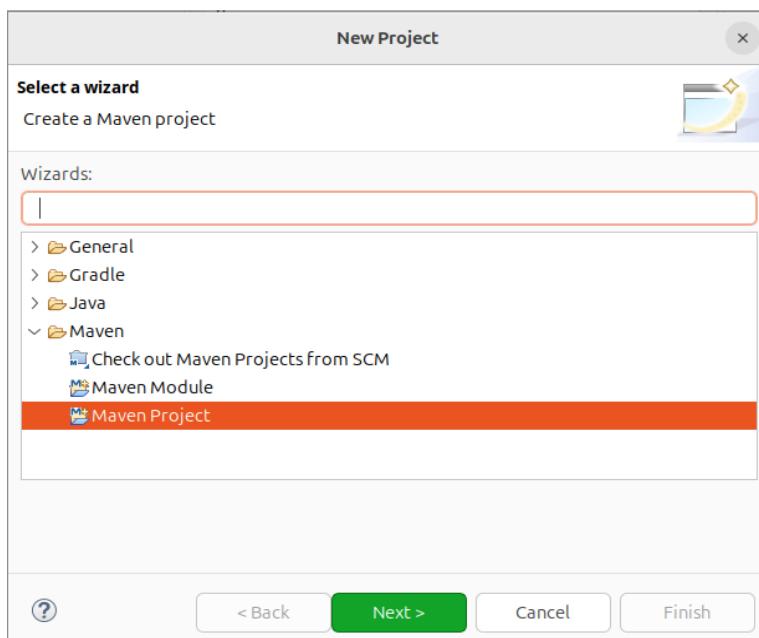
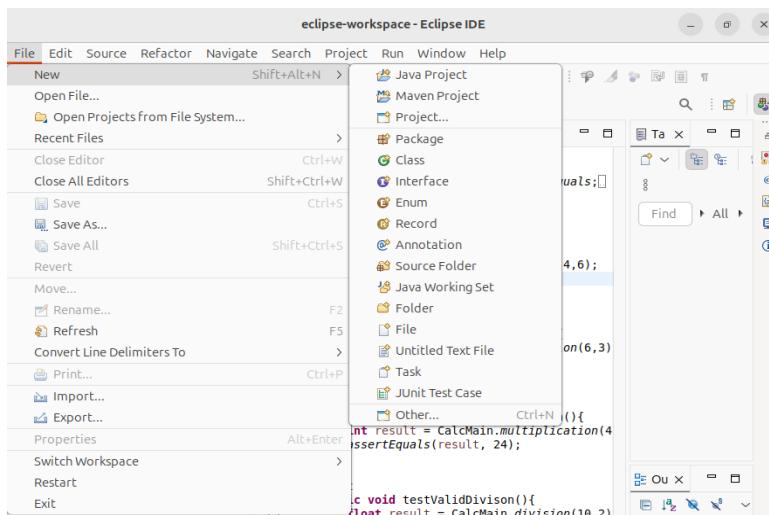
## 3. Configuring Eclipse Environment

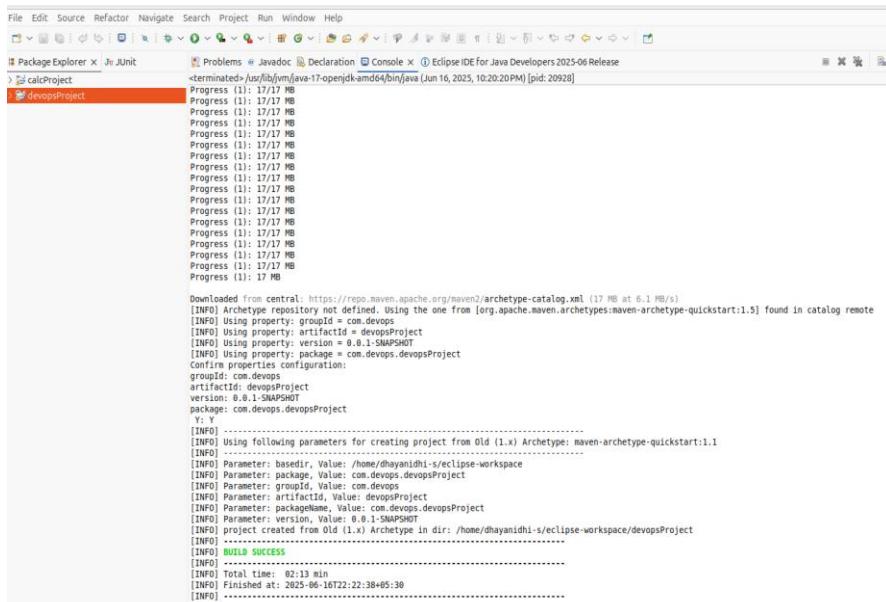
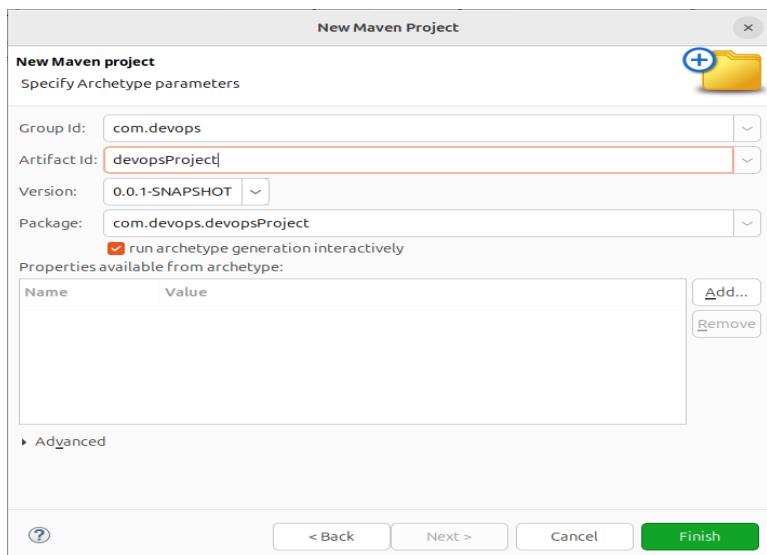
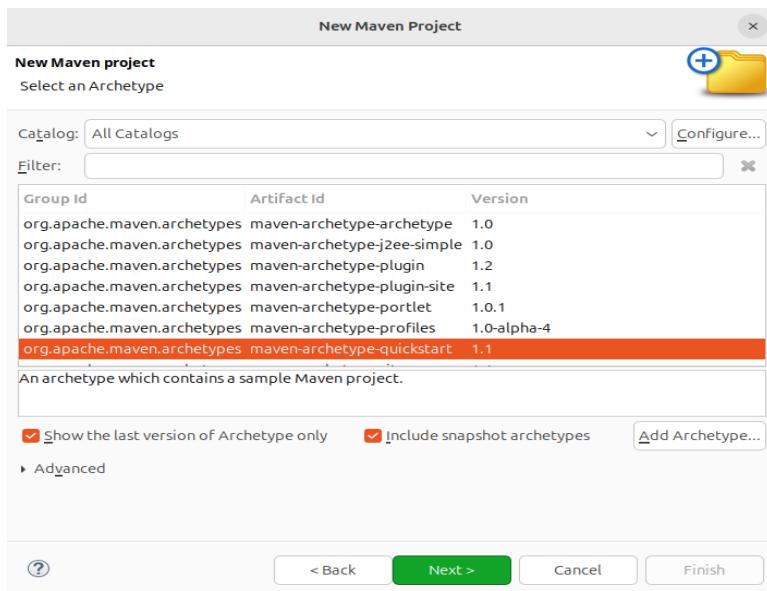
- Ensuring Eclipse is using the correct JDK via:  
Window → Preferences → Java → Installed JREs
- Selecting the path to **OpenJDK 17**.



## 4. Creating a Maven Project in Eclipse

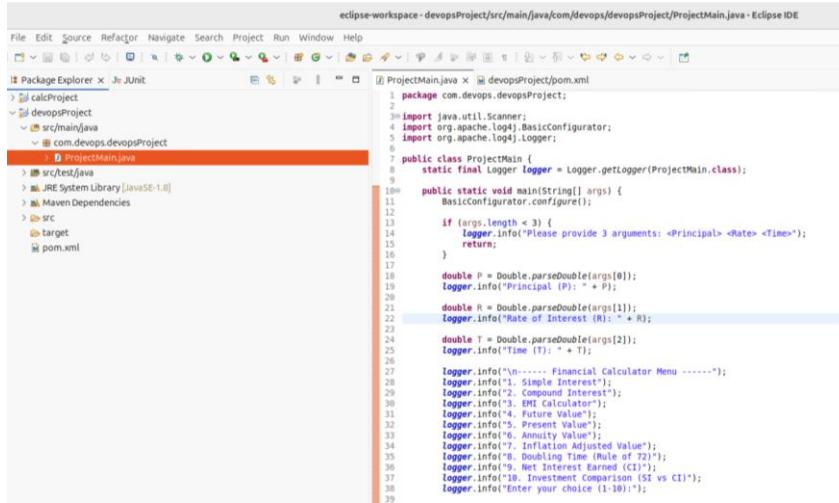
- File → New → Project → Maven → Maven Project
- Choose **archetype**, like maven-archetype-quickstart
- Fill in:
  - Group ID:** com.devops
  - Artifact ID:** devopsProject
- Click **Finish** to create the project structure.





## 5. Developing Financial Calculator App

- Inside src/main/java/com.devops.devopsProject, create ProjectMain.java with logic for:
  - Simple Interest , Compound Interest , EMI , Net Interest Earned and more as needed.
  - Accept command-line arguments and log output using **Log4j**
  - **Used java as language**



```
eclipse-workspace - devopsProject/src/main/java/com/devops/devopsProject/ProjectMain.java - Eclipse IDE

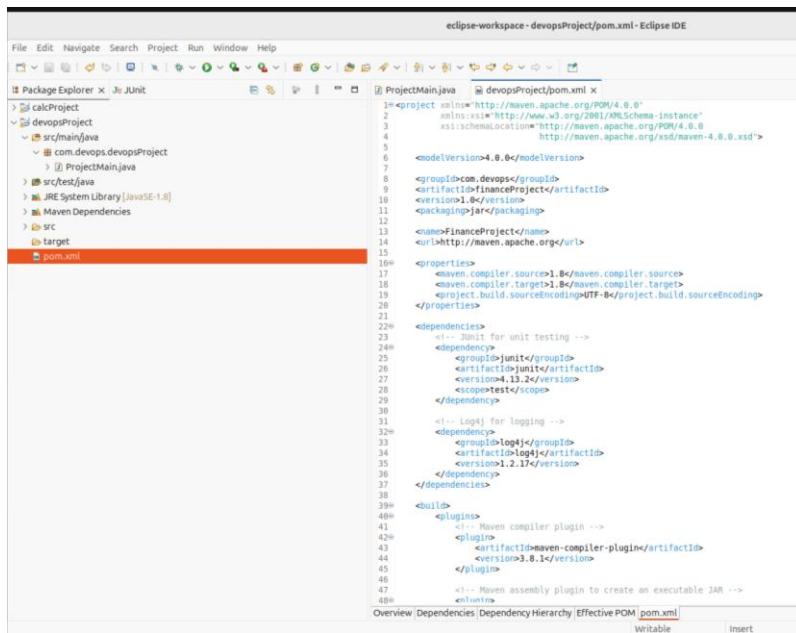
File Edit Source Refactor Navigate Search Project Run Window Help
File Project Explorer JUnit ProjectMain.java devopsProject/pom.xml
src/test/java JRE System Library [JavaSE-1.8]
src/main/java Maven Dependencies
src pom.xml

1 package com.devops.devopsProject;
2
3 import java.util.Scanner;
4 import org.apache.log4j.BasicConfigurator;
5 import org.apache.log4j.Logger;
6
7 public class ProjectMain {
8     static final Logger logger = Logger.getLogger(ProjectMain.class);
9
10    public static void main(String[] args) {
11        BasicConfigurator.configure();
12
13        if (args.length > 3) {
14            logger.info("Please provide 3 arguments: <Principal> <Rate> <Time>");
15            return;
16        }
17
18        double P = Double.parseDouble(args[0]);
19        logger.info("Principal (P): " + P);
20
21        double R = Double.parseDouble(args[1]);
22        logger.info("Rate of Interest (R): " + R);
23
24        double T = Double.parseDouble(args[2]);
25        logger.info("Time (T): " + T);
26
27        logger.info("\n----- Financial Calculator Menu -----");
28        logger.info("1. Simple Interest");
29        logger.info("2. Compound Interest");
30        logger.info("3. Future Value");
31        logger.info("4. Present Value");
32        logger.info("5. Future Value");
33        logger.info("6. Inflation Adjusted Value");
34        logger.info("7. Doubling Time (Rule of 72)");
35        logger.info("8. Net Interest Earned (CII)");
36        logger.info("9. Investment Comparison (SI vs CII)");
37        logger.info("10. Investment Comparison (SI vs SI)");
38
39        logger.info("Enter your choice (1-10):");
40    }
41}
```

## 6. Configuring pom.xml

Add Required Maven Dependencies which Include:

- junit:junit:4.12 for testing
- log4j:log4j:1.2.17 for logging
- maven-assembly-plugin for creating a JAR with dependencies



```
eclipse-workspace - devopsProject/pom.xml - Eclipse IDE

File Edit Navigate Search Project Run Window Help
File Project Explorer JUnit ProjectMain.java devopsProject/pom.xml
src/test/java JRE System Library [JavaSE-1.8]
src/main/java Maven Dependencies
src pom.xml

<project xmlns="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                           http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.devops</groupId>
    <artifactId>FinanceProject</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <name>FinanceProject</name>
    <url>http://maven.apache.org</url>
    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <!-- JUnit for unit testing -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
        <!-- Log4j for logging -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                </plugin>
            <!-- Maven assembly plugin to create an executable JAR -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-plugin</artifactId>
                <version>3.2.0</version>
                </plugin>
        </plugins>
    </build>

```

## 7. Building and Package the App

- Configure Build path
- Right-click project → Run As → Maven clean
- Then Run As → Maven install
- This generates a runnable JAR file in the target/ directory.

The screenshot shows two instances of the Eclipse IDE interface. The top window displays the 'Package Explorer' view with a project named 'devopsProject'. A context menu is open over the project node, with the 'Build Path' option highlighted. The bottom window also shows the 'Package Explorer' view with the same project. A 'Properties' dialog is open for 'devopsProject', specifically for the 'Java Build Path' tab. This dialog lists 'Source', 'Libraries', 'Order and Export', and 'Module Dependencies'. Under 'Libraries', 'JRE System Library [JavaSE-1.8]' is selected. On the right side of the dialog, there are buttons for 'Add JARs...', 'Add External JARs...', 'Add Variable...', 'Add Library...', 'Add Class Folder...', 'Add External Class Folder...', 'Edit...', 'Remove', and 'Migrate JAR File...'. At the bottom of the dialog are 'Apply' and 'Apply and Close' buttons. The code editor at the bottom of both windows shows a Java file named 'ProjectMain.java' with the following content:

```
1 package com.devops.devopsProject;
2
3 import java.util.Scanner;
4 import org.apache.log4j.BasicConfigurator;
5 import org.apache.log4j.Logger;
6
7 public class ProjectMain {
8     static final Logger logger = Logger.getLogger(ProjectMain.class);
9
10    public static void main(String[] args) {
11        BasicConfigurator.configure();
12
13        if (args.length < 3) {
14            logger.info("Please provide 3 arguments: <Principal> <Rate> <Time>");
15            return;
16        }
17
18        double P = Double.parseDouble(args[0]);
19        double R = Double.parseDouble(args[1]);
20        double T = Double.parseDouble(args[2]);
21
22        logger.info("----- Financial Calculator Menu -----");
23        logger.info("1. Simple Interest");
24        logger.info("2. Compound Interest");
25        logger.info("3. EMI Calculator");
26        logger.info("4. Future Value");
27        logger.info("5. Present Value");
28        logger.info("6. Annuity Value");
29        logger.info("7. Inflation Adjusted Value");
30        logger.info("8. Doubling Time (Rule of 72)");
31        logger.info("9. Net Interest Earned (CI)");
32        logger.info("10. Investment Comparison (SI vs CI)");
33
34        Scanner sc = new Scanner(System.in);
35        int choice = sc.nextInt();
36
37        switch (choice) {
38            case 1:
39                double si = (P * R * T) / 100;
40                logger.info("Simple Interest: " + String.format("%.2f", si));
41                break;
42            case 2:
43                double ci = P * Math.pow(1 + (R / 100), T);
44                logger.info("Compound Interest: " + String.format("%.2f", ci));
45                break;
46            case 3:
47                double emi = P * (Math.pow((1 + (R / 100)), T)) / ((Math.pow((1 + (R / 100)), T)) - 1);
48                logger.info("EMI: " + String.format("%.2f", emi));
49                break;
50            case 4:
51                double fv = P * Math.pow(1 + (R / 100), T);
52                logger.info("Future Value: " + String.format("%.2f", fv));
53                break;
54            case 5:
55                double pv = P / Math.pow(1 + (R / 100), T);
56                logger.info("Present Value: " + String.format("%.2f", pv));
57                break;
58            case 6:
59                double annuity = P * (Math.pow((1 + (R / 100)), T)) / ((Math.pow((1 + (R / 100)), T)) - 1);
60                logger.info("Annuity Value: " + String.format("%.2f", annuity));
61                break;
62            case 7:
63                double inflation = P * Math.pow(1 + (R / 100), T);
64                logger.info("Inflation Adjusted Value: " + String.format("%.2f", inflation));
65                break;
66            case 8:
67                double doublingTime = Math.log(2) / (R / 100);
68                logger.info("Doubling Time (Rule of 72): " + String.format("%.2f", doublingTime));
69                break;
70            case 9:
71                double netInterest = P * Math.pow(1 + (R / 100), T) - P;
72                logger.info("Net Interest Earned (CI): " + String.format("%.2f", netInterest));
73                break;
74            case 10:
75                double investmentComparison = (P * Math.pow(1 + (R / 100), T)) / P;
76                logger.info("Investment Comparison (SI vs CI): " + String.format("%.2f", investmentComparison));
77                break;
78            default:
79                logger.info("Enter your choice (1-10):");
80        }
81    }
82}
```

eclipse-workspace - devopsProject/src/main/java/com/devops/devopsProject/ProjectMain.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
devopsProject
src/main
com
src/test
JRE System Library [JavaSE-1.8]
src
target
pom.xml
New Go Into Open in New Window Open Type Hierarchy F4 Shift+Alt+W
Copy Ctrl+C Copy Qualified Name Ctrl+V Paste Ctrl+V Delete Remove from Context
Build Path Shift+Alt+S
Source Shift+Alt+T Refactor Shift+Alt+R Import...
Export...
Refresh F5 Close Project Close Unrelated Project Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Maven
Tgam
Compare With GitHub
Configure
Properties Alt+Enter
1 package com.devops.devopsProject;
2 import java.util.Scanner;
3 import org.apache.log4j.BasicConfigurator;
4 import org.apache.log4j.Logger;
5
6 public class ProjectMain {
7     static final Logger logger = Logger.getLogger(ProjectMain.class);
8
9     public static void main(String[] args) {
10         BasicConfigurator.configure();
11
12         if (args.length > 3) {
13             logger.info("Please provide 3 arguments: <Principal> <Rate> <Time>");
14             return;
15         }
16
17         double P = Double.parseDouble(args[0]);
18         logger.info("Principal (P): " + P);
19
20         double R = Double.parseDouble(args[1]);
21         logger.info("Rate of Interest (R): " + R);
22
23         double T = Double.parseDouble(args[2]);
24         logger.info("Time (T): " + T);
25
26         logger.info("\n----- Financial Calculator Menu -----");
27         logger.info("1. Simple Interest");
28         logger.info("2. Compound Interest");
29         logger.info("3. Future Value");
30         logger.info("4. Present Value");
31         logger.info("5. Present Value");
32         logger.info("6. Annuity Value");
33         logger.info("7. Inflation Adjusted Value");
34         logger.info("8. Net Present Value (Rule of 72)");
35         logger.info("9. Net Interest Earned (CI)");
36         logger.info("10. Investment Comparison (SI vs CI)");
37         logger.info("Enter your choice (1-10):");
38
39         Scanner sc = new Scanner(System.in);
40         int choice = sc.nextInt();
41
42         switch (choice) {
43             case 1:
44                 double SI = (P * R * T) / 100;
45                 logger.info("Simple Interest: " + String.format("%.2f", SI));
46                 break;
47
48             case 2:
49
50             case 3:
51
52             case 4:
53
54             case 5:
55
56             case 6:
57
58             case 7:
59
60             case 8:
61
62             case 9:
63
64             case 10:
65
66             default:
67                 logger.info("Invalid choice");
68         }
69     }
70 }

```

eclipse-workspace - devopsProject/src/main/java/com/devops/devopsProject/ProjectMain.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
devopsProject
src/main/java
com.devops.devopsProject
ProjectMain.java
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
Problems Javadoc Declaration Console Eclipse IDE for Java Developers 2025-06 Release
terminated: /usr/lib/jvm/java-17-openjdk-amd64/bin/java (Jun 16, 2025, 10:48:36 PM) [pid: 30550]
[INFO] Scanning for projects...
[INFO] ...
< com.devops:financeProject 1.0
[INFO]   from pom.xml
[INFO] ...
[INFO] [jar]
[INFO] ...
< com.devops:financeProject 1.0
[INFO]   from pom.xml
[INFO] ...
[INFO] Deleting /home/dhayaniidhi-s/eclipse-workspace/devopsProject/target
[INFO] BUILD SUCCESS
[INFO] ...
[INFO]   time: 0.261
[INFO] Finished at: 2025-06-16T22:48:38+05:30
[INFO] ...

```

Maven clean completed successfully

eclipse-workspace - devopsProject/src/main/java/com/devops/devopsProject/ProjectMain.java - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit
devopsProject
src/main/java
com.devops.devopsProject
ProjectMain.java
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
Problems Javadoc Declaration Console Eclipse IDE for Java Developers 2025-06 Release
terminated: /usr/lib/jvm/java-17-openjdk-amd64/bin/java (Jun 16, 2025, 10:49:29 PM) [pid: 30553]
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/felix/org.apache.felix.bundlerepository/1.6.2/org.apache.felix.bundlerepository-1.6.2.jar (151 kB at 1)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/wapache/maven-profile/2.0.7/maven-profile-2.0.7.jar
[INFO] Downloading from central: https://repo.maven.apache.org/nexus2/0.4/esynmock-2.4.jar (81 kB at 811 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/nexus2/0.4/esynmock-2.4.jar (81 kB at 811 kB/s)
[INFO] ...
[INFO] [jar]
[INFO] ...
< com.devops:financeProject 1.0
[INFO]   from pom.xml
[INFO] ...
[INFO] [clean:3.2.0:clean (default-clean) @ financeProject]
[INFO] ...
[INFO] Deleting /home/dhayaniidhi-s/eclipse-workspace/devopsProject/target
[INFO] BUILD SUCCESS
[INFO] ...
[INFO]   time: 0.261
[INFO] Finished at: 2025-06-16T22:48:38+05:30
[INFO] ...

```

Maven install completed successfully

```

eclipse-workspace - devopsProject/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Window Help
File Problems Declaration Console Eclipse IDE for Java Developers 2025-06 Release
<terminated> /usr/lib/jvm/java-17-openjdk-amd64/bin/java [Jun 16, 2025, 10:56:39 PM] [pid: 33624]
[INFO] from pom.xml
[INFO] .....[ jar ].....
[INFO] --- resources:3.3.1:resources (default-resources) @ devopsProject ---
[INFO] skip non existing resourceDirectory /home/dhayanidhi-s/eclipse-workspace/devopsProject/src/main/resources
[INFO] ...
[INFO] ... compiler:3.8.1:compile (default-compile) @ devopsProject ...
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/classes
[INFO] ...
[INFO] --- resources:3.3.1:testResources (default-testResources) @ devopsProject ---
[INFO] skip non existing resourceDirectory /home/dhayanidhi-s/eclipse-workspace/devopsProject/src/test/resources
[INFO] ...
[INFO] ... compiler:3.8.1:testCompile (default-testCompile) @ devopsProject ...
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/test-classes
[INFO] ...
[INFO] --- surefire:3.2.5:test (default-test) @ devopsProject ...
[INFO] Using auto detected provider org.apache.maven.surefire.junit4.JUnit4Provider
[INFO] ...
[INFO] .....
[INFO] T E S T S
[INFO] .....
[INFO] Running com.devops.devopsProject.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.038 s -- in com.devops.devopsProject.AppTest
[INFO] Results:
[INFO] ...
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] ...
[INFO] --- jar:3.4.1:jar (default-jar) @ devopsProject ...
[INFO] Building jar: /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/devopsProject-1.0.jar
[INFO] ...
[INFO] --- assembly:3.3.0:single (make-assembly) @ devopsProject ...
[INFO] Building jar: /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/devopsProject-1.0-jar-with-dependencies.jar
[INFO] ...
[INFO] --- install:3.1:install (default-install) @ devopsProject ...
[INFO] Installing /home/dhayanidhi-s/eclipse-workspace/devopsProject/pom.xml to /home/dhayanidhi-s/.m2/repository/com/devops/devopsProject/1.0/devopsProject-1.0.pom
[INFO] Installing /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/devopsProject-1.0.jar to /home/dhayanidhi-s/.m2/repository/com/devops/devopsProject/1.0/devopsProject-1.0.jar
[INFO] Installing /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/devopsProject-1.0-jar-with-dependencies.jar to /home/dhayanidhi-s/.m2/repository/com/devops/devopsProject/1.0/devopsProject-1.0-jar-with-dependencies.jar
[INFO] .....
[INFO] BUILD SUCCESS
[INFO] .....
[INFO] Total time: 3.393 s
[INFO] Finished at: 2025-06-16T22:56:43+05:30
[INFO] .....

```

Home / eclipse-workspace / devopsProject / target

archive-tmp classes generated-sources generated-test-sources maven-archiver maven-status surefire-reports test-classes devopsProject-1.0.jar devopsProject-1.0-jar-with-dependencies.jar

**Runnable JAR file is successfully created on the target directory .**

## 8. Run the Executable JAR

Test the app by running the executable JAR :

```
java -jar target/finance-calc-1.0-jar-with-dependencies.jar 10000 7.5 3 1
```

```

dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ java -jar /home/dhayanidhi-s/eclipse-workspace/devopsProject/target/devopsProject-1.0-jar-with-dependencies.jar "10000" "7.5" "3" "1"
0 [main] INFO com.devops.devopsProject.ProjectMain - Principal (P): 10000.0
0 [main] INFO com.devops.devopsProject.ProjectMain - Rate of Interest (R): 7.5
0 [main] INFO com.devops.devopsProject.ProjectMain - Time (T): 3.0
----- Financial Calculator Menu -----
0 [main] INFO com.devops.devopsProject.ProjectMain - 1. Simple Interest
0 [main] INFO com.devops.devopsProject.ProjectMain - 2. Compound Interest
1 [main] INFO com.devops.devopsProject.ProjectMain - 3. EMI Calculator
1 [main] INFO com.devops.devopsProject.ProjectMain - 4. Future Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 5. Present Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 6. Annuity Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 7. Inflation Adjusted Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 8. Doubling Time (Rule of 72)
1 [main] INFO com.devops.devopsProject.ProjectMain - 9. Net Interest Earned (CI)
1 [main] INFO com.devops.devopsProject.ProjectMain - 10. Investment Comparison (SI vs CI)
1 [main] INFO com.devops.devopsProject.ProjectMain - Enter your choice (1-10):
38 [main] INFO com.devops.devopsProject.ProjectMain - Simple Interest: ₹2250.00
dhayanidhi-s@dhyanidhi-s-VirtualBox:~$ sudo docker build -t devops_project:1.0 .
[+] Building 0.2s (1/1) FINISHED
      docker :default

```

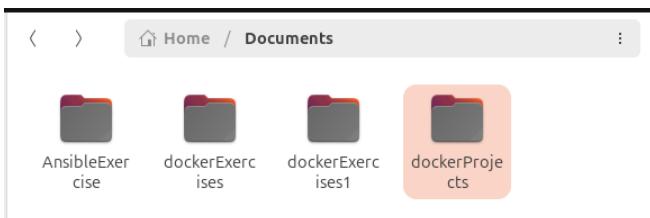
(Where 10000 7.5 3 2 are input arguments for operations)

# Docker Setup & Dockerization of Java Financial Calculator

## 1. Install Docker on Ubuntu

## 2. Add User to Docker Group

## 3. Create the dockerProjects Directory

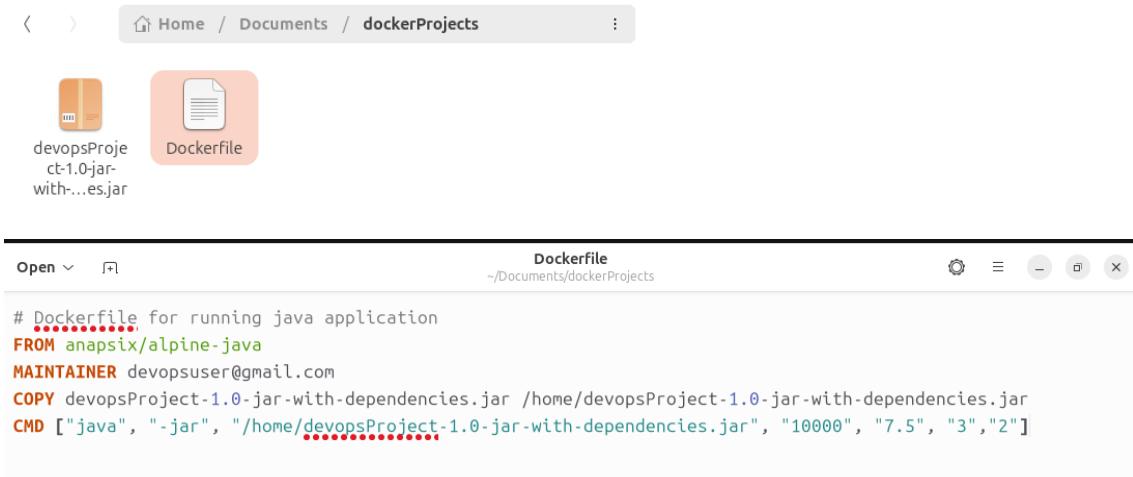


Created a dedicated folder for all Docker-related files, keeping things organized.

## 4. Copy the JAR File into the Directory

Copied the compiled executable JAR (from the Maven project) into the Docker workspace for image creation.

## 5. Create the Dockerfile



```
# Dockerfile for running java application
FROM anapsix/alpine-java
MAINTAINER devopsuser@gmail.com
COPY devopsProject-1.0-jar-with-dependencies.jar /home/devopsProject-1.0-jar-with-dependencies.jar
CMD ["java", "-jar", "/home/devopsProject-1.0-jar-with-dependencies.jar", "10000", "7.5", "3","2"]
```

Defines the Java environment, copies the app, and sets default execution parameters for your calculator.

## 6. Confirm Dockerfile Content

Use this to double-check the contents of your Dockerfile before building to avoid syntax or path errors.

```
cat dockerProjects/Dockerfile
```

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ cd Documents/dockerProjects
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$ cat Dockerfile
# Dockerfile for running java application
FROM anapsix/alpine-java
MAINTAINER devopsuser@gmail.com
COPY devopsProject-1.0-jar-with-dependencies.jar /home/devopsProject-1.0-jar-with-dependencies.jar
CMD ["java", "-jar", "/home/devopsProject-1.0-jar-with-dependencies.jar", "10000", "7.5", "3","2"]
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$
```

## 7. Build the Docker Image

```
cd dockerProjects
docker build -t devops_project:1.0 .
```

Builds a Docker image from the Dockerfile. -t tags it for easy identification as devops\_calc version 1.0.

---

## 8. Run the Docker Container

```
docker run devops_calc:1.0
```

Launches a container from the image and runs the financial calculator app with sample arguments (like Principal, Rate, Operation Type, and Time).

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ cd Documents/dockerProjects
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$ sudo docker build -t devops_project:1.0 .
[+] Building 3.2s (7/7) FINISHED                                            docker:default
--> [internal] load build definition from Dockerfile                      0.0s
--> => transferring dockerfile: 336B                                         0.0s
=> WARN: MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label 0.0s
--> [internal] load metadata for docker.io/anapsix/alpine-java:latest      2.6s
--> [internal] load .dockerignore                                           0.0s
--> => transferring context: 2B                                         0.0s
--> [internal] load build context                                         0.0s
--> => transferring context: 487.68kB                                     0.0s
=> CACHED [1/2] FROM docker.io/anapsix/alpine-java:latest@sha256:1d24bc352e07b84c073acfff8b 0.0s
=> [2/2] COPY devopsProject-1.0-jar-with-dependencies.jar /home/devopsProject-1.0-jar-with- 0.2s
=> exporting to image                                                 0.1s
=> => exporting layers                                              0.1s
=> => writing image sha256:1479186483db6cff96dfc119b033c0ddbcda1c579ace47b8b27b54310d749762 0.0s
=> => naming to docker.io/library/devops_project:1.0                      0.0s

1 warning found (use docker --debug to expand):
- MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label (line 3)
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$ sudo docker run devops_project:1.0
0 [main] INFO com.devops.devopsProject.ProjectMain - Principal (P): 10000.0
1 [main] INFO com.devops.devopsProject.ProjectMain - Rate of Interest (R): 7.5
1 [main] INFO com.devops.devopsProject.ProjectMain - Time (T): 3.0
1 [main] INFO com.devops.devopsProject.ProjectMain -
----- Financial Calculator Menu -----
1 [main] INFO com.devops.devopsProject.ProjectMain - 1. Simple Interest
1 [main] INFO com.devops.devopsProject.ProjectMain - 2. Compound Interest
1 [main] INFO com.devops.devopsProject.ProjectMain - 3. EMI Calculator
1 [main] INFO com.devops.devopsProject.ProjectMain - 4. Future Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 5. Present Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 6. Annuity Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 7. Inflation Adjusted Value
1 [main] INFO com.devops.devopsProject.ProjectMain - 8. Doubling Time (Rule of 72)
1 [main] INFO com.devops.devopsProject.ProjectMain - 9. Net Interest Earned (CI)
1 [main] INFO com.devops.devopsProject.ProjectMain - 10. Investment Comparison (SI vs CI)
1 [main] INFO com.devops.devopsProject.ProjectMain - Enter your choice (1-10):
54 [main] INFO com.devops.devopsProject.ProjectMain - Compound Interest: ₹2422.97
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$
```

Runs financial calculator JAR and prints the result for the provided inputs

## JUnit Testing for Java Financial Calculator

### 1. Add JUnit Dependency in pom.xml

Open pom.xml file and ensure the following snippet is added inside the <dependencies> section:

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

This imports the JUnit 4.12 framework, allowing you to write and run test cases for all calculator methods.

### 2. Create Test Class CalcTest.java

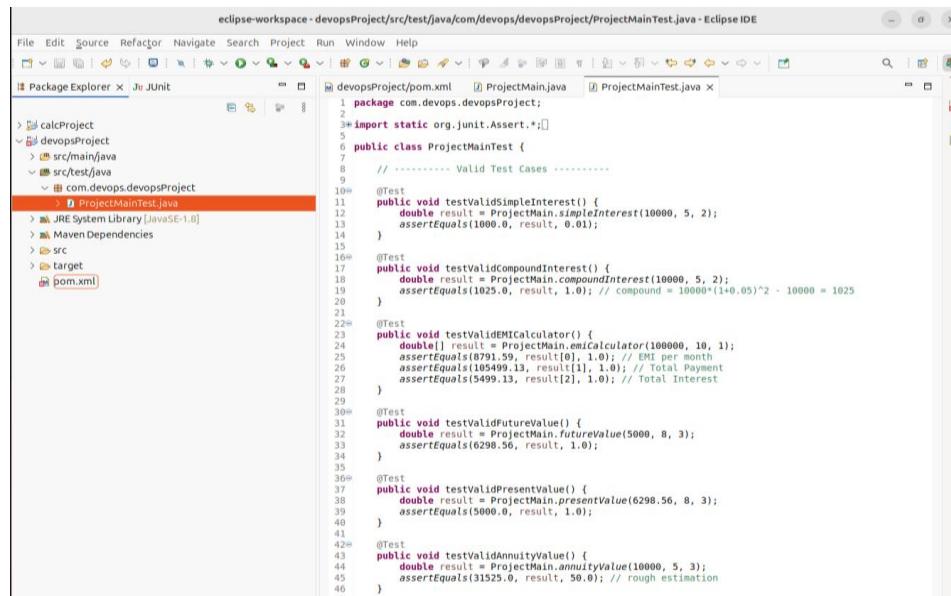
Inside your Maven project, navigate to:

src/test/java → com.devops.calcProject

Then, create a new Java class named ProjectMainTest.java. This folder is dedicated for unit testing in Maven-based projects.

### 3. Write JUnit Test Cases

Paste JUnit test code into ProjectMainTest.java.



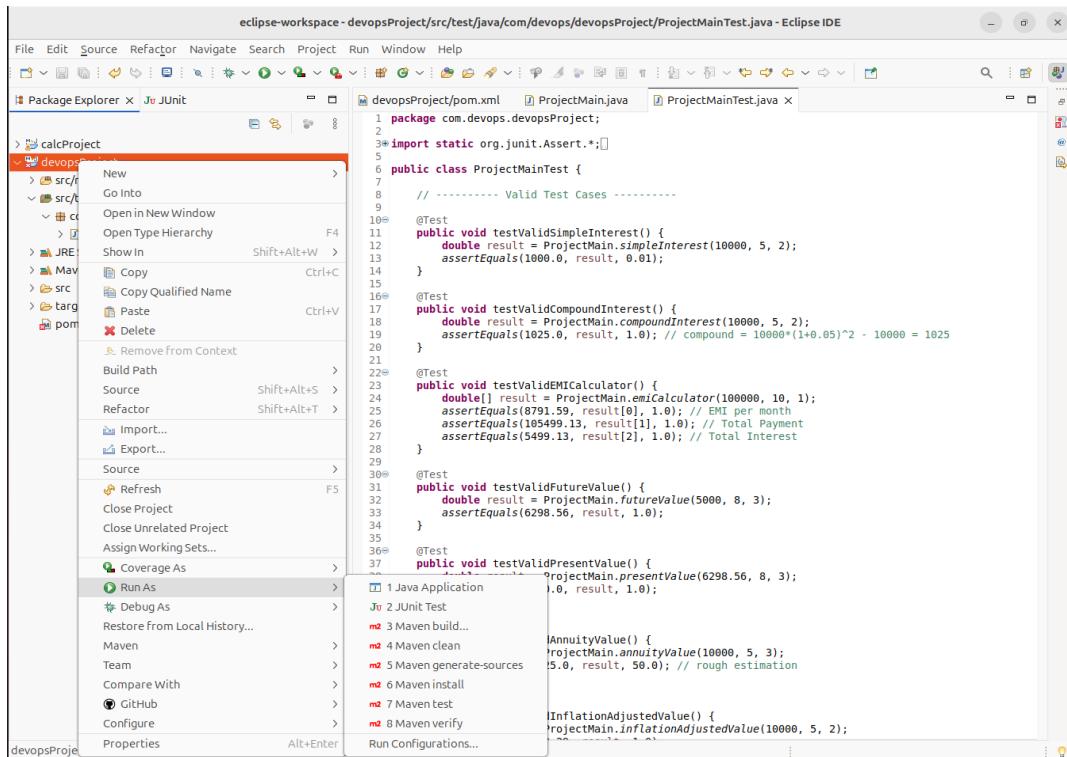
```
eclipse-workspace - devopsProject/src/test/java/com/devops/devopsProject/ProjectMainTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer JUnit devopsProject/pom.xml ProjectMain.java ProjectMainTest.java
calcProject devopsProject
src/main
src/test/java
com.devops.devopsProject
ProjectMainTest.java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml

1 package com.devops.devopsProject;
2
3 import static org.junit.Assert.*;
4
5 public class ProjectMainTest {
6
7     // ----- Valid Test Cases -----
8
9     @Test
10    public void testValidSimpleInterest() {
11        double result = ProjectMain.simpleInterest(10000, 5, 2);
12        assertEquals(1000.0, result, 0.01);
13    }
14
15    @Test
16    public void testValidCompoundInterest() {
17        double result = ProjectMain.compoundInterest(10000, 5, 2);
18        assertEquals(1025.0, result, 1.0); // compound = 10000*(1+0.05)^2 - 10000 = 1025
19    }
20
21    @Test
22    public void testValidEMICalculator() {
23        double[] result = ProjectMain.emiCalculator(100000, 10, 1);
24        assertEquals(8791.59, result[0], 1.0); // EMI per month
25        assertEquals(105499.13, result[1], 1.0); // Total Payment
26        assertEquals(15499.13, result[2], 1.0); // Total Interest
27    }
28
29    @Test
30    public void testValidFutureValue() {
31        double result = ProjectMain.futureValue(5000, 8, 3);
32        assertEquals(6298.56, result, 1.0);
33    }
34
35    @Test
36    public void testValidPresentValue() {
37        double result = ProjectMain.presentValue(6298.56, 8, 3);
38        assertEquals(5000.0, result, 1.0);
39    }
40
41    @Test
42    public void testValidAnnuityValue() {
43        double result = ProjectMain.annuityValue(10000, 5, 3);
44        assertEquals(31525.0, result, 50.0); // rough estimation
45    }
46}
```

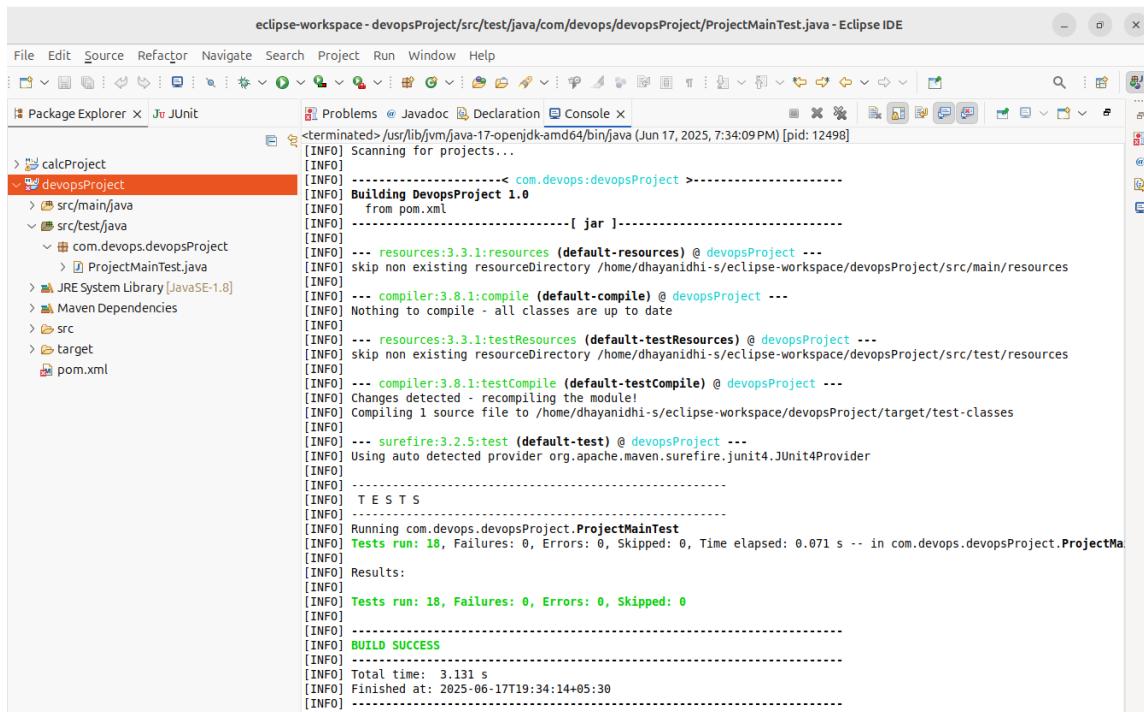
These test both **valid** and **invalid** cases for your calculator methods, ensuring correctness and catching bugs.

## 4. Run All Tests via Maven

Right-click on your project → Run As → Maven test



This command automatically finds and runs all test files under src/test/java and shows results in the console.



## 5. Run JUnit Test Class Individually (Optional)

To run just ProjectMainTest.java:

- Right-click on the class file

- Click **Run As → JUnit Test**

This displays a graphical green/red bar interface indicating which tests passed or failed.

The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - devopsProject/src/test/java/com/devops/devopsProject/ProjectMainTest.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left side features the Package Explorer view with "ProjectMainTest" selected, showing "Runs: 18/18", "Errors: 0", and "Failures: 0". The center contains the JUnit View, which lists 18 test cases under "com.devops.devopsProject.ProjectMainTest [Runner: JUnit]". All test cases are marked with a green icon and have a duration of "0.000 s". The right side is the code editor showing the Java code for ProjectMainTest:

```
1 package com.devops.devopsProject;
2 import static org.junit.Assert.*;
3
4 public class ProjectMainTest {
5     // ----- Valid Test Cases -----
6     @Test
7     public void testValidSimpleInterest() {
8         double result = ProjectMain.simpleInterest(10000, 5, 2);
9         assertEquals(1000.0, result, 0.01);
10    }
11
12    @Test
13    public void testValidCompoundInterest() {
14        double result = ProjectMain.compoundInterest(10000, 5, 2);
15        assertEquals(1025.0, result, 1.0); // compound = 10000*(1+0.05)^2 - 10000 = 1025
16    }
17
18    @Test
19    public void testValidEMICalculator() {
20        double[] result = ProjectMain.emiCalculator(100000, 10, 1);
21        assertEquals(8791.59, result[0], 1.0); // EMI per month
22        assertEquals(105499.13, result[1], 1.0); // Total Payment
23        assertEquals(5499.13, result[2], 1.0); // Total Interest
24    }
25
26    @Test
27    public void testValidFutureValue() {
28        double result = ProjectMain.futureValue(5000, 8, 3);
29        assertEquals(6298.56, result, 1.0);
30    }
31
32    @Test
33    public void testValidPresentValue() {
34        double result = ProjectMain.presentValue(6298.56, 8, 3);
35        assertEquals(5000.0, result, 1.0);
36    }
37
38    @Test
39    public void testValidAnnuityValue() {
40        double result = ProjectMain.annuityValue(10000, 5, 3);
41        assertEquals(31525.0, result, 50.0); // rough estimation
42    }
43
44    @Test
45    public void testValidInflationAdjustedValue() {
46        double result = ProjectMain.inflationAdjustedValue(10000, 5, 2);
47    }
48}
```

All Test Cases were successful .



## Deploying Financial Calculator on Kubernetes with Minikube

---



### 1. Install Kubernetes (Minikube)

Minikube helps you create a lightweight Kubernetes cluster locally.  
Set up Minikube on your Ubuntu system.



### 2. Start Minikube and Connect Docker Daemon

To work with Minikube's internal Docker environment:

```
minikube start  
eval $(minikube docker-env)
```

This ensures Docker images built will be available to your Kubernetes pods.

The screenshot shows a terminal window with the following text:

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$ minikube start  
minikube v1.36.0 on Ubuntu 24.04 (vbox/amd64)  
Using the docker driver based on existing profile  
Starting "minikube" primary control-plane node in "minikube" cluster  
Pulling base image v0.0.47 ...  
Restaring existing docker container for "minikube" ...  
Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...  
Verifying Kubernetes components...  
  Using image docker.io/kubernetesui/metrics-scraper:v1.0.8  
  Using image docker.io/kubernetesui/dashboard:v2.7.0  
  Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.5.3  
  Using image registry.k8s.io/ingress-nginx/controller:v1.12.2  
  Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.5.3  
  Using image gcr.io/k8s-minikube/storage-provisioner:v5  
  Using image gcr.io/k8s-minikube/kube-registry-proxy:0.0.9  
  Using image docker.io/registry:3.0.0  
Verifying ingress addon...  
Some dashboard features require the metrics-server addon. To enable all features please run:  
  minikube addons enable metrics-server  
Verifying registry addon...  
Enabled addons: default-storageclass, storage-provisioner, dashboard, ingress, registry  
! /usr/local/bin/kubectl is version 1.31.0, which may have incompatibilities with Kubernetes 1.33.1.  
  ■ Want kubectl v1.33.1? Try 'minikube kubectl -- get pods -A'  
  ■ Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ eval $(minikube docker-env)  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ cd Documents/dockerProjects  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/Documents/dockerProjects$ docker build -t devops_project:1.0 .  
[+] Building 2.6s (7/7) FINISHED  
          docker:default  
=> [internal] load build definition from Dockerfile           0.1s  
=> => transferring dockerfile: 336B                           0.0s  
=> WARN: MaintainerDeprecated: Maintainer instruction is deprecated in f  0.1s  
=> [internal] load metadata for docker.io/anapsix/alpine-java:latest   2.1s  
=> [internal] load .dockerrcignore                            0.0s  
=> => transferring context: 2B                                0.0s  
=> [internal] load build context                          0.1s  
=> => transferring context: 487.68kB                         0.0s  
=> CACHED [1/2] FROM docker.io/anapsix/alpine-java:latest@sha256:1d24bc3  0.0s  
=> [2/2] COPY devopsProject-1.0-jar-with-dependencies.jar /home/devopsPr  0.1s  
=> exporting to image                                     0.1s  
=> exporting layers                                      0.0s  
=> writing manifest                                         0.0s
```



### 3. Build the Docker Image Inside Minikube

Navigate to Dockerfile directory and build the image:

```
docker build -t devops_project:1.0 .
```

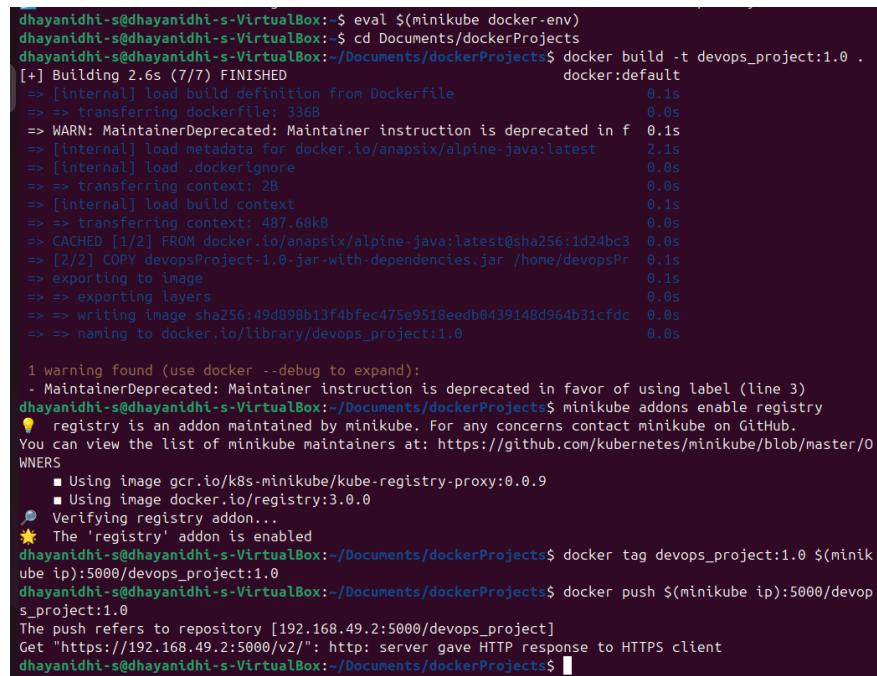
This creates the devops\_project:1.0 image directly inside Minikube's Docker engine.

## 📦 4. Enable Minikube Registry & Push Image

If you prefer pushing to Minikube's internal registry:

```
minikube addons enable registry
docker tag devops_project:1.0 $(minikube ip):5000/devops_project:1.0
docker push $(minikube ip):5000/devops_project:1.0
```

⚠️ Skip if image was already built with eval \$(minikube docker-env).



```
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~ eval $(minikube docker-env)
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~ cd Documents/dockerProjects
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~/Documents/dockerProjects$ docker build -t devops_project:1.0 .
[+] Building 2.6s (7/7) FINISHED
   docker:default
=> [internal] load build definition from Dockerfile          0.1s
=> => transferring dockerfile: 336B                          0.0s
=> WARN: MaintainerDeprecated: Maintainer instruction is deprecated in f 0.1s
=> [internal] load metadata for docker.io/anapsix/alpine-java:latest 2.1s
=> => [internal] load .dockerrcignore                      0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load build context                         0.1s
=> => transferring context: 487.68kB                     0.0s
=> CACHED [1/2] FROM docker.io/anapsix/alpine-java:latest@sha256:1d24bc3 0.0s
=> [2/2] COPY devopsProject-1.0-jar-with-dependencies.jar /home/devopsPr 0.1s
=> exporting to image                                     0.1s
=> => exporting layers                                    0.0s
=> => writing image sha256:49d898b13f4bfec475e9518edb0439148d964b31cfdc 0.0s
=> => naming to docker.io/library/devops_project:1.0      0.0s

1 warning found (use docker --debug to expand):
- MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label (line 3)
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~/Documents/dockerProjects$ minikube addons enable registry
💡 registry is an addon maintained by minikube. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  ■ Using image gcr.io/k8s-minikube/kube-registry-proxy:0.0.9
  ■ Using image docker.io/registry:3.0.0
🔍 Verifying registry addon...
🌟 The 'registry' addon is enabled
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~/Documents/dockerProjects$ docker tag devops_project:1.0 $(minikube ip):5000/devops_project:1.0
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~/Documents/dockerProjects$ docker push $(minikube ip):5000/devops_project:1.0
The push refers to repository [192.168.49.2:5000/devops_project]
Get "https://192.168.49.2:5000/v2/": http: server gave HTTP response to HTTPS client
dhayanidhi-s@dhayanidhi-s-VirtualBox: ~/Documents/dockerProjects$
```

## ✍️ 5. Create Kubernetes Deployment YAML File

Create the deployment file:

```
sudo nano calculator-deployment.yaml
```



```
GNU nano 7.2                                         project-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: project-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: moneymetrics
  template:
    metadata:
      labels:
        app: moneymetrics
    spec:
      containers:
        - name: moneymetrics  # ✅ all lowercase
          image: devops_project:1.0
          imagePullPolicy: Never
          ports:
            - containerPort: 8080
```

Home / Documents / dockerProjects

devopsProject-1.0-jar-with...es.jar Dockerfile project-deployment.yaml

This configures a pod to run the Dockerized calculator app.

## 6. Apply Deployment and Expose Service

Run the following commands to deploy and expose the app:

```
kubectl apply -f calculator-deployment.yaml  
kubectl expose deployment calculator-deployment --type=NodePort --port=8080
```

This makes the app available over a NodePort in your Minikube cluster.

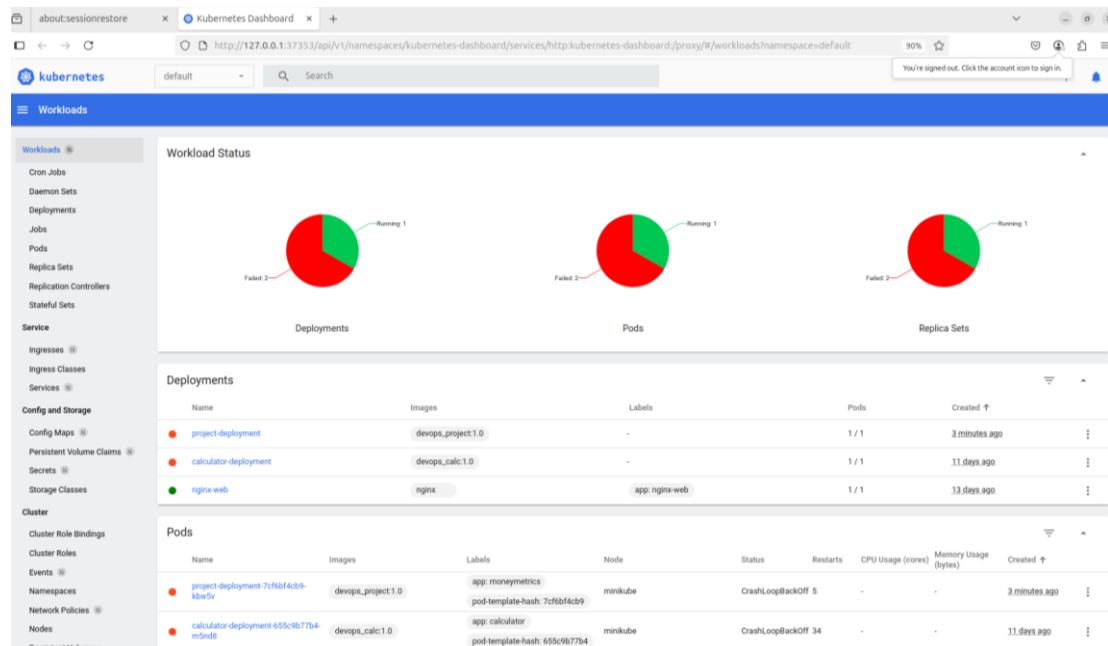
---

## 7. Launch Dashboard and Monitor Logs

Start the Minikube dashboard to visually monitor pods and logs:

```
minikube dashboard
```

```
The push refers to repository [192.168.49.2:5000/devops_project]  
Get "https://192.168.49.2:5000/v2/: http server gave HTTP response to HTTPS client  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ sudo nano project-deployment.yaml  
[sudo] password for dhayanidhi:  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ kubectl apply -f project-deployment.yaml  
error: error when retrieving current configuration of:  
Resource: "apps/v1, Resource=deployments", GroupVersionKind: "apps/v1, Kind=Deployment"  
Name: "", Namespace: "default"  
from server for "project-deployment.yaml": resource name may not be empty  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ sudo nano project-deployment.yaml  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ kubectl apply -f project-deployment.yaml  
The Deployment "project-deployment" is invalid: spec.template.spec.containers[0].name: Invalid value: "MoneyMetrics": a lowercase RFC 1123 label must consist of lower case alphanumeric characters or '-', and must start and end with an alphanumeric character (e.g. 'my-name', or 'i23-abc', regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?')  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ sudo nano project-deployment.yaml  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ kubectl apply -f project-deployment.yaml  
deployment.apps/project-deployment created  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ sudo nano project-deployment.yaml  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ kubectl expose deployment project-deployment --type=NodePort --port=8080  
  
service/project-deployment exposed  
dhayanidhi@dhayanidhi-VirtualBox:~/Documents/dockerProjects$ minikube dashboard  
🟡 Verifying dashboard health ...  
Launching proxy...  
🟡 Verifying proxy health ...  
🔗 Opening http://127.0.0.1:37353/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard/proxy/ in your default browser...  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/local/share/doc /usr/local/share/doc none bind,ro 0 0): cannot open directory "/var/lib/local/share": permission denied  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/gimp/2.0/help /usr/share/gimp/2.0/help none bind,ro 0 0): cannot write to "/var/lib/snapd/hostfs/usr/share/gimp/2.0/help" because it would affect the host in "/var/lib/snapd"  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/gtk-doc /usr/share/gtk-doc none bind,ro 0 0): cannot write to "/var/lib/snapd/hostfs/usr/share/gtk-doc" because it would affect the host in "/var/lib/snapd"  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/javascript /usr/share/javascript none bind,ro 0 0): cannot write to "/var/lib/snapd/hostfs/usr/share/javascript" because it would affect the host in "/var/lib/snapd"  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/libreoffice/help /usr/share/libreoffice/help none bind,ro 0 0): cannot write to "/var/lib/snapd/hostfs/usr/share/libreoffice/help" because it would affect the host in "/var/lib/snapd"  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/sphinx_rtd_theme /usr/share/sphinx_rtd_theme none bind,ro 0 0): cannot write to "/var/lib/snapd/hostfs/usr/share/sphinx_rtd_theme" because it would affect the host in "/var/lib/snapd"  
update_pp:85: cannot change mount namespace according to change mount (/var/lib/snapd/hostfs/usr/share/xubuntu-docs /usr/share/xubuntu-docs none bind,ro 0 0): cannot write to "/var/lib/snapd/hostfs/usr/share/xubuntu-docs" because it would affect the host in "/var/lib/snapd"  
Ctk-Message: gtk-bridge:Gtk-Message: 2023-01-10T11:45:53.580Z: Not loading module "atk-bridge": The functionality is provided by GTK natively. Please try to not load it.
```

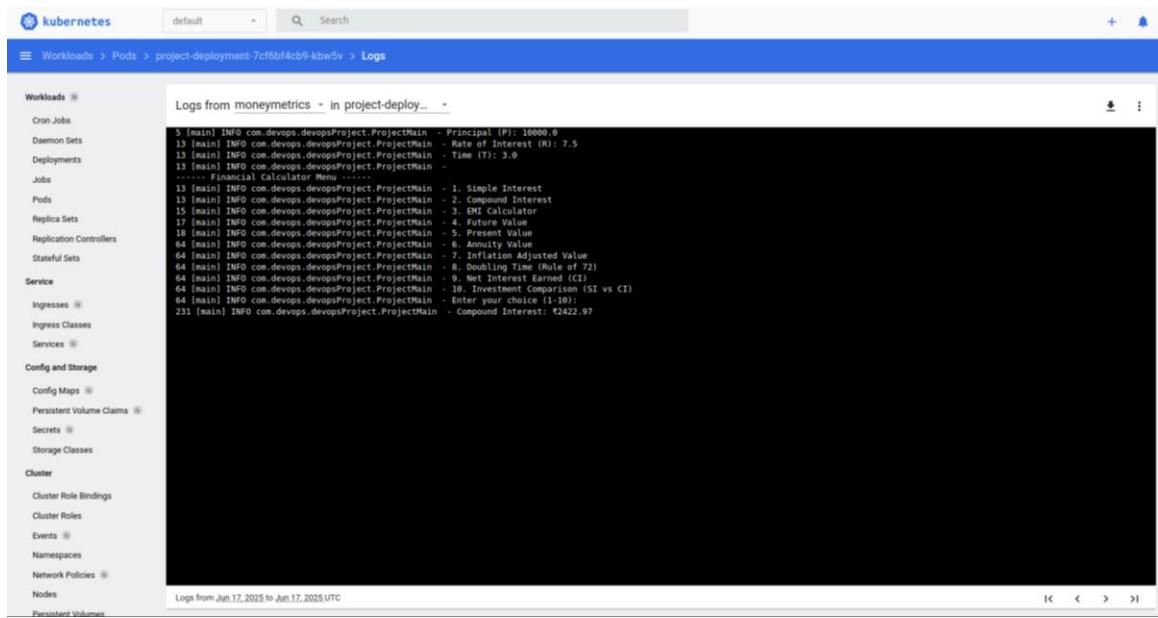


Name	Images	Labels	Pods	Created
project-deployment	devops_project:1.0	-	1/1	3 minutes ago
calculator-deployment	devops_calc:1.0	-	1/1	11 days ago
nginx-web	nginx	app:nginx-web	1/1	13 days ago

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
project-deployment-7cfbf4cb9-lkbwv	devops_project:1.0	app:moniemetrics pod-template-hash:7cfbf4cb9	minikube	CrashLoopBackOff	5	-	-	3 minutes ago
calculator-deployment-655c9b77b4-mndbb	devops_calc:1.0	app:calculator pod-template-hash:655c9b77b4	minikube	CrashLoopBackOff	34	-	-	11 days ago

Go to Pods → calculator-deployment → Logs to verify the container output.



The screenshot shows the Kubernetes UI interface. At the top, there's a navigation bar with the Kubernetes logo, a dropdown menu set to 'default', a search bar with placeholder text 'Search', and a '+' button. Below the navigation bar, the path 'Workloads > Pods > project-deployment-7cf6bf74cb9-kbw5v > Logs' is displayed. On the left side, there's a sidebar with several sections: Workloads (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets), Service (Ingresses, Ingress Classes, Services), and Config and Storage (Config Maps, Persistent Volume Claims, Secrets, Storage Classes). The main content area is titled 'Logs from moneymetrics in project-deploy...' and contains the following log entries:

```
0 [main] INFO com.devops.devopsProject.ProjectMain Principal (P): 10000.0
13 [main] INFO com.devops.devopsProject.ProjectMain Rate of Interest (R): 7.5
13 [main] INFO com.devops.devopsProject.ProjectMain - Time (T): 3.0
13 [main] INFO com.devops.devopsProject.ProjectMain -
..... Financial Calculator Menu .....
13 [main] INFO com.devops.devopsProject.ProjectMain - 1. Simple Interest
13 [main] INFO com.devops.devopsProject.ProjectMain - 2. Compound Interest
15 [main] INFO com.devops.devopsProject.ProjectMain - 3. EMI Calculator
17 [main] INFO com.devops.devopsProject.ProjectMain - 4. Future Value
18 [main] INFO com.devops.devopsProject.ProjectMain - 5. Present Value
60 [main] INFO com.devops.devopsProject.ProjectMain - 6. Annuity Value
64 [main] INFO com.devops.devopsProject.ProjectMain - 7. Bond Price and Yield Adjusted Value
64 [main] INFO com.devops.devopsProject.ProjectMain - 8. Doubling Time (Rule of 72)
64 [main] INFO com.devops.devopsProject.ProjectMain - 9. Net Interest Earned (CI)
64 [main] INFO com.devops.devopsProject.ProjectMain - 10. Investment Comparison (SI vs CI)
64 [main] INFO com.devops.devopsProject.ProjectMain - Enter your choice (1-10):
131 [main] INFO com.devops.devopsProject.ProjectMain - Compound Interest: 12422.97
```

At the bottom of the log area, it says 'Logs from Jun 17, 2023 to Jun 17, 2023 UTC'. To the right of the log area, there are navigation icons for left, right, and first/last pages.



## Monitoring using Grafana and Graphite

This stage integrates **Graphite** (for storing system metrics) and **Grafana** (for beautiful visual dashboards) to monitor your **Java-based Financial Calculator application**.

---



### 1. Install Docker and Docker Compose ( and Nagios )

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ sudo curl -L "https://github.com/docker/compose/releases/download/v2.37.0/docker-compose-$(uname -s)-$(uname -m)" \
> -o /usr/local/bin/docker-compose && sudo chmod +x /usr/local/bin/docker-compose
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total Spent   Left Speed
0       0      0      0      0      0      0 --:--:-- --:--:-- --:--:-- 0
100  71.3M  100  71.3M  0      0 2091k      0 0:00:34  0:00:34  --:--:-- 2631k
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ sudo chmod +x /usr/local/bin/docker-compose
```

---



### 2. Configure Docker Compose for Grafana and Graphite

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ mkdir ~/grafana && cd ~/grafana
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/grafana$ nano docker-compose.yml
```

```
GNU nano 7.2                               docker-compose.yml
Version: "3"
services:
  grafana:
    image: grafana/grafana
    container_name: grafana
    restart: always
    ports:
      - 3000:3000
    networks:
      - grafana-net
    volumes:
      - grafana-volume:/var/lib/grafana

  graphite:
    image: graphiteapp/graphite-statsd
    container_name: graphite
    restart: always
    networks:
      - grafana-net

networks:
  grafana-net:

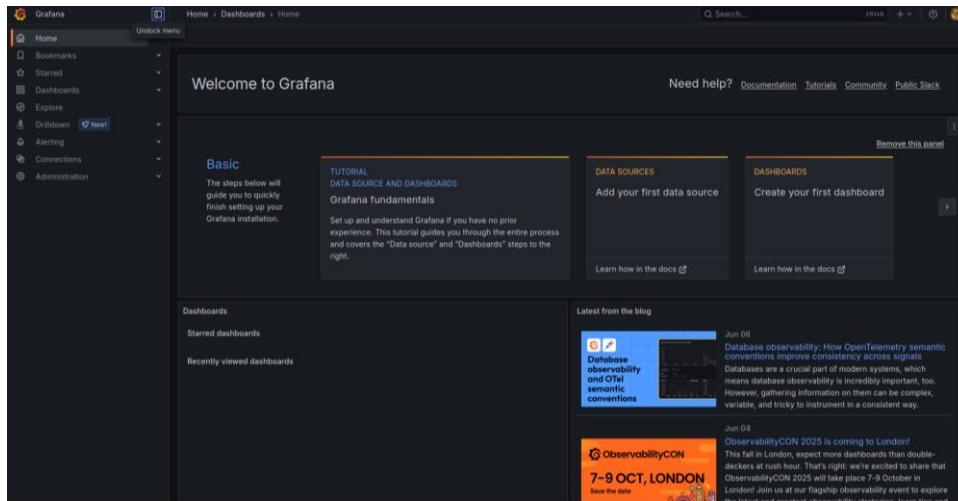
volumes:
  grafana-volume:
    external: true
```

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/grafana$ sudo docker volume create --name=grafana-volume
grafana-volume
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/grafana$ sudo docker-compose up -d
WARN[0000] /home/dhayanidhi-s/grafana/docker-compose.yml: the attribute `version` is obsolete, it will be
ignored, please remove it to avoid potential confusion
[+] Running 17/17
  ✓ graphite Pulled                                188.5s
  ✓ 0cdfa0c98ed7 Pull complete                     15.5s
  ✓ 723656d3099c Pull complete                     174.0s
  ✓ 620093e6e6f0 Pull complete                     174.2s
  ✓ fcf6a0ec82d9 Pull complete                     174.2s
  ✓ f09364cff1fd Pull complete                     183.2s
  ✓ grafana Pulled                                 128.7s
  ✓ f18232174bc9 Pull complete                     4.2s
  ✓ 65babbe3dfe5 Pull complete                     4.3s
  ✓ 651b0ba49b07 Pull complete                     4.8s
  ✓ d953cde4314b Pull complete                     7.1s
  ✓ aecd4cb03450 Pull complete                     7.2s
  ✓ 13fa68ca8757 Pull complete                     7.3s
  ✓ f836d47fdc4d Pull complete                     116.1s
  ✓ 8b5292c946e1 Pull complete                     123.3s
  ✓ 454a4350d439 Pull complete                     123.3s
  ✓ 9a8c18aee5ea Pull complete                     123.4s
[+] Running 3/3
  ✓ Network grafana_grafana-net Created           0.2s
  ✓ Container graphite Started                    1.4s
  ✓ Container grafana Started                    1.4s
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/grafana$
```

---

### 3. Access Grafana Dashboard

 Open Grafana in browser and sign in with credentials . If new both username and password are admin then set new password.

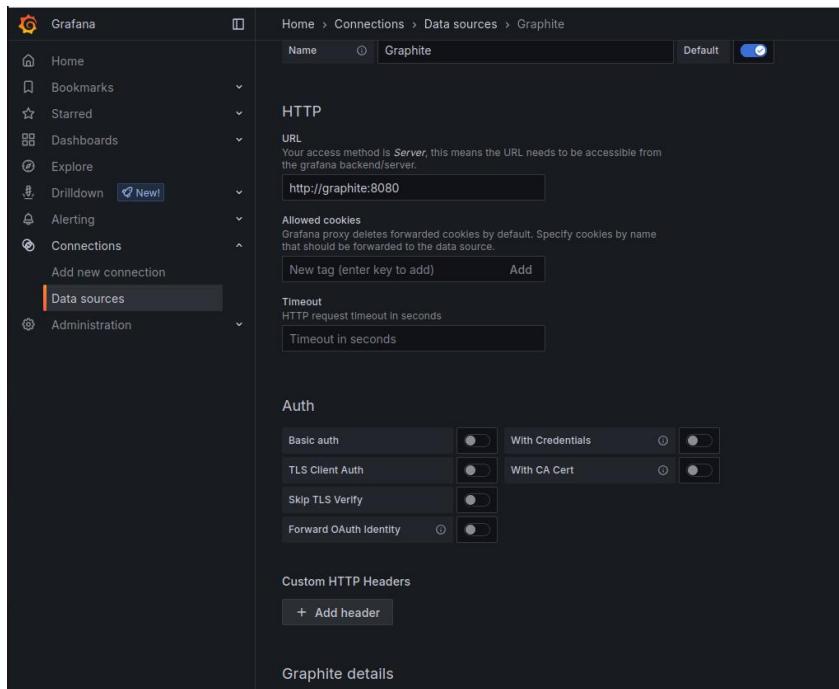


### 4. Add Graphite as Data Source in Grafana

 Navigate:

Settings → Data Sources → Add Data Source → Choose "Graphite"

 After proper Configuration Click **Save & Test**



### 5. Import Dashboard Template

 Import community dashboard:

- Click **Dashboard** → **Import**
  - Enter **Dashboard ID: 55**
  - Select **Graphite** as data source and **import**



Now see a predefined system metrics dashboard.

## 6. Install and Configure Collectd



## Install Collectd:



#### Enable essential plugins: Important



## 7. Restart Collectd



**Restart the service:**

```
sudo systemctl restart collectd  
sudo systemctl status collectd
```



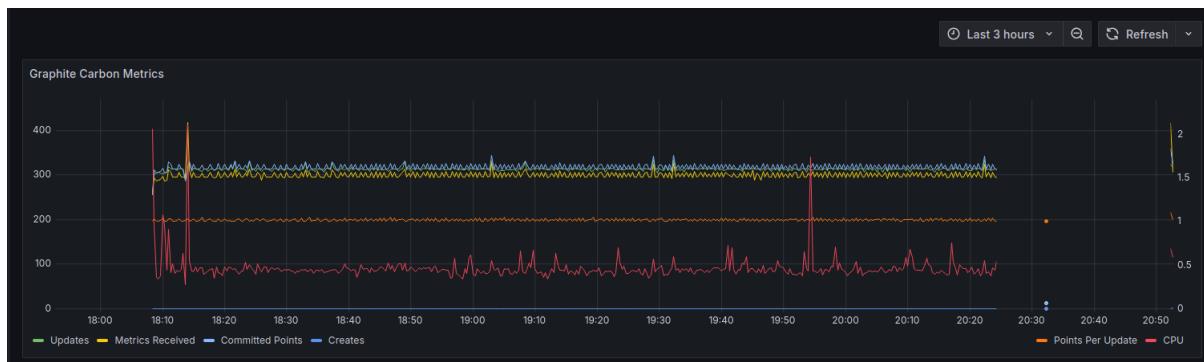
## 8. Visualize Metrics in Grafana



Metrics like:

- CPU usage
  - RAM usage
  - Disk utilization
  - Network bandwidth

will now stream live from **Collectd** → **Graphite** → **Grafana**, helping to monitor the application/server health.



# Ansible Automation Setup for devops\_Project

## 1. Install Ansible

Install Ansible using apt:

```
sudo apt update  
sudo apt install ansible -y  
ansible --version
```

Installs Ansible, a configuration management tool used to automate the entire CI/CD and monitoring pipeline.

## 2. Create Ansible Project Directory

Navigate to Git project repository and create a folder for Ansible:

Keeps the Ansible configuration separate and organized within the project repository.

## 3. Create Ansible Roles

Inside the ansible\_project, create the required Ansible roles and supporting files:

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ ls  
ansible_project      eclipse-workspace      Public  
CreateFileIfNotExists.yml  grafana          puppet8-release-jammy.deb  
Desktop              main.tf            puppet8-release-jammy.deb.1  
devops_repository     minikube-linux-amd64  puppet8-release-jammy.deb.2  
Documents             Music              sample.txt  
Downloads             Pictures           snap  
eclipse               pom.xml            Templates  
eclipse-installer     project_repository  Videos  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ cd ansible_project  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ touch inventory.ini  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano inventory.ini  
[sudo] password for dhayanidhi-s:  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ touch site.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano site.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano site.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ mkdir -p roles/{gitclone,build,test,dockerize  
,deploy,monitor}/tasks  
touch roles/gitclone/tasks/main.yml  
touch roles/build/tasks/main.yml  
touch roles/test/tasks/main.yml  
touch roles/dockerize/tasks/main.yml  
touch roles/deploy/tasks/main.yml  
touch roles/monitor/tasks/main.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano roles/gitclone/tasks/main.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano roles/build/tasks/main.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano roles/test/tasks/main.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano roles/dockerize/tasks/main.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano roles/deploy/tasks/main.yml  
  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$ sudo nano roles/monitor/tasks/main.yml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/ansible_project$
```

Each role automates a distinct step in the pipeline (e.g., Java installation, Docker build, Kubernetes deploy, etc.)

## 4. Create site.yml (Master Playbook)

Create a central playbook file to execute all roles in sequence:



```
site.yml
~/project_repository/ansible_project

---
- name: Full DevOps Pipeline
  hosts: local
  become: true

  vars:
    build_number: "{{ build_number }}"
    build_path: "{{ build_path }}"

  roles:
    - { role: build, tags: ['build'] }
    - { role: test, tags: ['test'] }
    - { role: dockerize, tags: ['dockerize'] }
    - { role: deploy, tags: ['deploy'] }
    - { role: monitor, tags: ['monitor'] }
    - { role: rollback, tags: ['rollback'] }
```

This playbook controls the execution order of all roles on your local host.

## 5. Create inventory.ini File

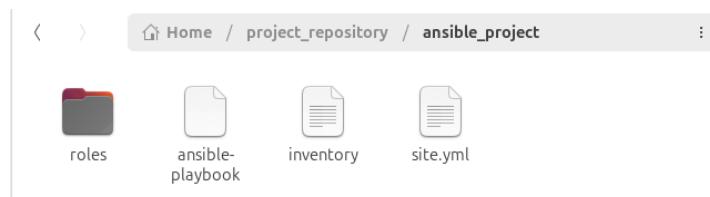
Create inventory file to define the target machine (here: localhost):



```
inventory
~/project_repository/ansible_project

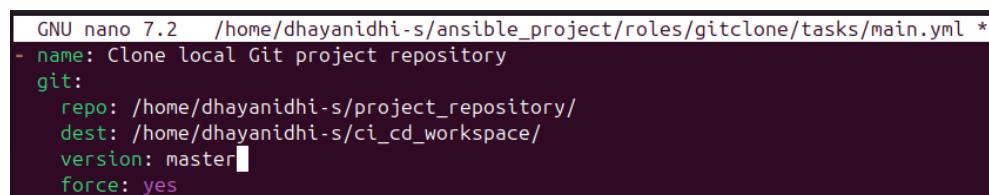
[local]
localhost ansible_connection=local
```

Specifies where the Ansible roles should run — in this case, on your own system.



## 6. Add Tasks in Each Role (main.yml)

Navigate into each role's tasks folder and add commands specific to the pipeline.



```
GNU nano 7.2  /home/dhayanidhi-s/ansible_project/roles/gitclone/tasks/main.yml *
- name: Clone local Git project repository
  git:
    repo: /home/dhayanidhi-s/project_repository/
    dest: /home/dhayanidhi-s/ci_cd_workspace/
    version: master
    force: yes
```

```
GNU nano 7.2          roles/build/tasks/main.yml *
- name: Build the Java project using Maven
  command: mvn clean package
  args:
    chdir: /home/dhayanidhi-s/ci_cd_workspace
```

```
GNU nano 7.2          roles/test/tasks/main.yml *
- name: Run JUnit tests using Maven
  command: mvn test
  args:
    chdir: /home/dhayanidhi-s/ci_cd_workspace
```

```
GNU nano 7.2          roles/deploy/tasks/main.yml *
- name: Run Docker container from built image
  docker_container:
    name: financial-calc
    image: financial-calc:v1
    state: started
    restart_policy: always
    ports:
      - "8081:8080"
```

```
GNU nano 7.2          roles/dockerize/tasks/main.yml
name: Copy Dockerfile to workspace
copy:
  src: /home/dhayanidhi-s/Documents/dockerProjects/Dockerfile
  dest: /home/dhayanidhi-s/ci_cd_workspace/Dockerfile

name: Build Docker image from project
command: docker build -t financial-calc:v1 .
args:
  chdir: /home/dhayanidhi-s/ci_cd_workspace
```

```
GNU nano 7.2          roles/monitor/tasks/main.yml *
- name: Copy Docker Compose file to monitoring folder
  copy:
    src: docker-compose.yml
    dest: /home/dhayanidhi-s/ci_cd_workspace/monitoring/docker-compose.yml

- name: Start monitoring stack with Docker Compose
  command: docker-compose up -d
  args:
    chdir: /home/dhayanidhi-s/ci_cd_workspace/monitoring
```

```
GNU nano 7.2          /roles/monitor/files/docker-compose.yml *
version: '3'

services:
  graphite:
    image: graphiteapp/graphite-statsd
    container_name: graphite
    ports:
      - "80:80"
      - "2003-2004:2003-2004"
      - "2023-2024:2023-2024"
      - "8125:8125/udp"
      - "8126:8126"
    restart: always

  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=admin
    restart: always
```

Each role automates one major DevOps phase of the project.

## 7. Run the Ansible Playbook

Now execute the entire pipeline automation:

```
bash
Copy code
ansible-playbook -i inventory.ini site.yml
```

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:/~/ansible_project$ nano roles/monitor/files/docker-compose.yml
dhayanidhi-s@dhayanidhi-s-VirtualBox:/~/ansible_project$ cd ~/ansible_project
ansible-playbook -i inventory.ini site.yml

PLAY [Full DevOps Pipeline for Project] ****
TASK [Gathering Facts] ****
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.12, but future
installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.18/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [gitclone : Clone local Git project repository] ****
ok: [localhost]

TASK [build : Build the Java project using Maven] ****
changed: [localhost]

TASK [test : Run JUnit tests using Maven] ****
changed: [localhost]

TASK [dockerize : Copy Dockerfile to workspace] ****
ok: [localhost]

TASK [dockerize : Build Docker image from project] ****
changed: [localhost]

TASK [deploy : Run Docker container from built image] ****
ok: [localhost]

TASK [monitor : Create monitoring directory inside workspace] ****
ok: [localhost]

TASK [monitor : Copy Docker Compose file to monitoring folder] ****
changed: [localhost]

TASK [monitor : Start monitoring stack with Docker Compose] ****
changed: [localhost]

PLAY RECAP ****
localhost          : ok=10   changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

dhayanidhi-s@dhayanidhi-s-VirtualBox:/~/ansible_project$
```

This will run all roles sequentially: Java setup → Maven build → Docker build → Test → Deploy → Enable Monitoring.

# CI/CD Pipeline Setup for Java Financial Calculator using Jenkins

---

## ❖ 1. Install Jenkins and set up

## ❖ 2. Install Required Jenkins Plugins

Go to <http://localhost:8080> → **Manage Jenkins** → **Manage Plugins**

- Under "Available" tab, install:
  - *Hudson Post Build Task Plugin*
  - *PostBuildScript Plugin*

These plugins allow Jenkins to run post-build tasks like executing the generated JAR file.

---

## 🌐 3. Push Your Maven Project to Git Server

Move your pom.xml and src/ folder into the Git repository, then:

```
git add .
git status
git commit -m "Added POM and src"
```

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$ cd project_repository
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$ git add .
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   devopsProject-1.0-jar-with-dependencies.jar
    new file:   pom.xml
    new file:   src/main/java/com/devops/devopsProject/ProjectMain.java
    new file:   src/test/java/com/devops/devopsProject/ProjectMainTest.java

dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$ git commit -m "Added POM and src"
[master (root-commit) 834adc8] Added POM and src
 4 files changed, 336 insertions(+)
 create mode 100644 devopsProject-1.0-jar-with-dependencies.jar
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/devops/devopsProject/ProjectMain.java
 create mode 100644 src/test/java/com/devops/devopsProject/ProjectMainTest.java
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$ git branch
* master
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$ git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (17/17), 438.13 KiB | 5.62 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To localhost:project_repository.git
 * [new branch]      master -> master
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/project_repository$
```

---

## 4. Grant Jenkins Ownership of the Repository

Ensure Jenkins can read/write to the Git repo:

```
sudo chown -R jenkins:jenkins /home/devops/devops_repository  
sudo chmod -R 755 /home/devops/devops_repository
```

```
dhayanidhi-s@dhayanidhi-s-VirtualBox:~$ cd devops_repository  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/devops_repository$ sudo chown -R jenkins:jenkins /home/dhayanidhi-s/project_repository  
[sudo] password for dhayanidhi-s:  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/devops_repository$ sudo chmod -R 755 /home/dhayanidhi-s/project_repository  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/devops_repository$ sudo vi pom.xml  
dhayanidhi-s@dhayanidhi-s-VirtualBox:~/devops_repository$
```

---

## 5. Add Required Maven Plugin to pom.xml

Edit the pom.xml file to ensure it compiles with Java 17

```
<groupId>org.apache.maven.plugins</groupId>  
<artifactId>maven-compiler-plugin</artifactId>  
<version>3.8.1</version>  
<configuration>  
<source>17</source>  
<target>17</target>  
</configuration>  
</plugin>  
</plugins>
```

---

## 6. Validate Jenkins Git Plugin

Go to: **Manage Jenkins → Script Console** and run:

```
System.properties['hudson.plugins.git.GitSCM.ALLOW_LOCAL_CHECKOUT']
```

If result ≠ true, edit:

```
sudo nano /etc/default/jenkins
```

Add:

```
JAVA_ARGS="-Dhudson.plugins.git.GitSCM.ALLOW_LOCAL_CHECKOUT=true"
```

The screenshot shows the Jenkins Script Console interface. At the top, it says "Dashboard > Manage Jenkins > Script Console". Below that is a "Build Executor Status" section with "0/2" available. The main area is titled "Script Console" and contains a text input field with the following Groovy code:

```
println(Jenkins.instance.pluginManager.plugins)
```

A note below the code states: "All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported." In the code editor, the line `1 System.getProperty("hudson.plugins.git.GitSCM.ALLOW_LOCAL_CHECKOUT")` is highlighted.

At the bottom right of the code editor is a "Run" button. Below the editor, there's a "Result" section with a "Result" button and a "Copy" icon. The result output shows "Result: true".

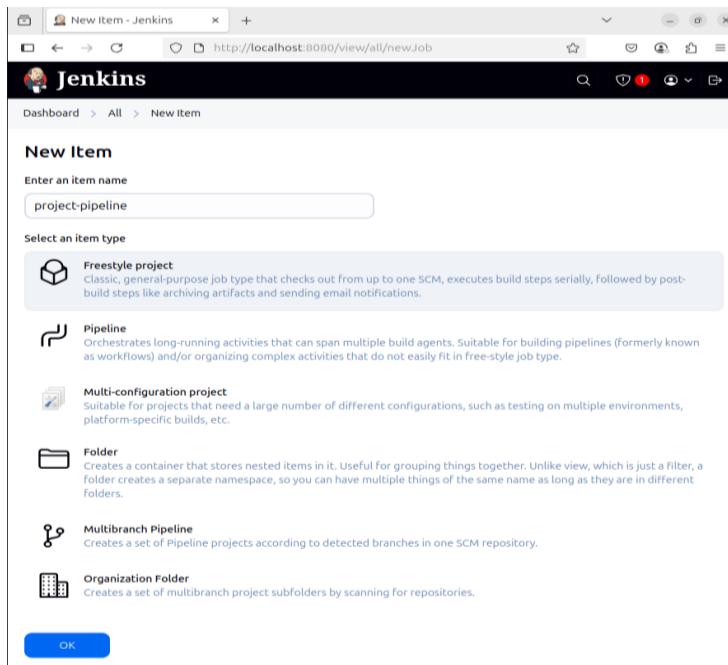
👉 In this project, we will explore both the **Freestyle Project** and the **Pipeline** methods in Jenkins to automate the build, test, and deployment processes of our Java-based Financial Calculator.

For freestyle project continue from step 7 .

## 🛠 7. Create a Freestyle Jenkins Project

From Jenkins homepage:

- Click **New Item**
- Select **Freestyle Project**
- Name it project-pipeline

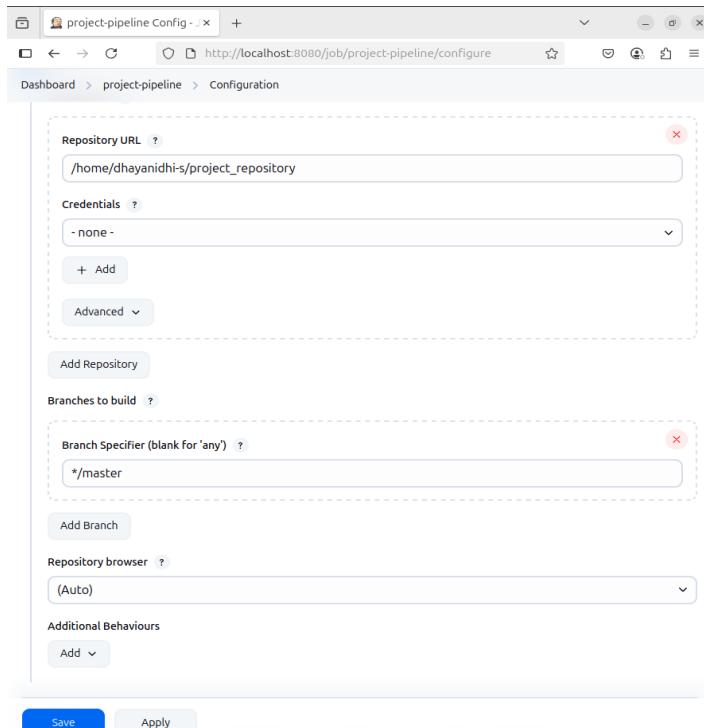


This is our CI/CD job configuration.

## 🔗 8. Connect Git Repository to Jenkins

Inside your new Jenkins project:

- Go to **Source Code Management**
- Select **Git**
- Enter local repo path or URL



Jenkins will now pull code from the Git repo.

---

## 🔨 9. Configure Build Command

- Click on **Add Build Step → Execute Shell**

Paste:

```
mvn clean package -Dmaven.compiler.source=17 -Dmaven.compiler.target=17
```

This builds the JAR using Maven inside Jenkins workspace.

To automate all the processes

---

## 🔑 10. Grant Jenkins Sudo Access Without Password

Edit the sudoers file:

```
sudo visudo
```

Add this line:

```
jenkins ALL=(ALL) NOPASSWD: ALL
```

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
devops  ALL=(ALL:ALL) ALL
Jenkins ALL=(ALL) NOPASSWD: ALL

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
```

Jenkins can now execute post-build scripts with root access.

---

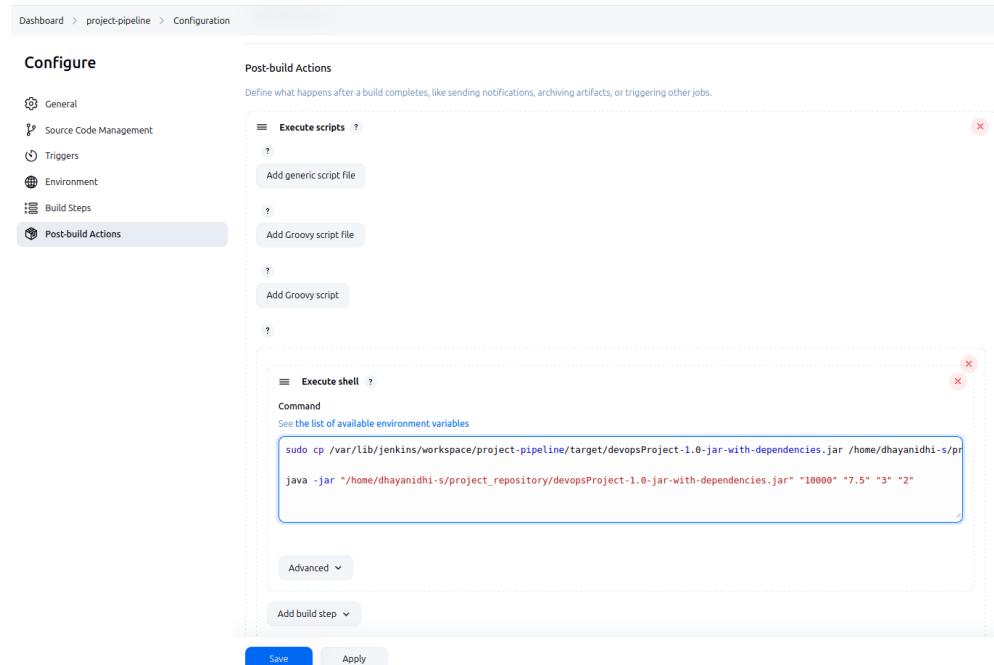
## 11. Post-Build Script to Copy & Run JAR

In **Post-build Actions** → Add:

- **Add post-build action → Execute Scripts**
- Paste:

```
sudo cp /var/lib/jenkins/workspace/project-pipeline/target/devopsProject-1.0-jar-with-dependencies.jar /home/devops/devops_repository
```

```
java -jar "/home/devops/devops_repository/devopsProject-1.0-jar-with-dependencies.jar" "10000" "7.5" "3" "2"
```



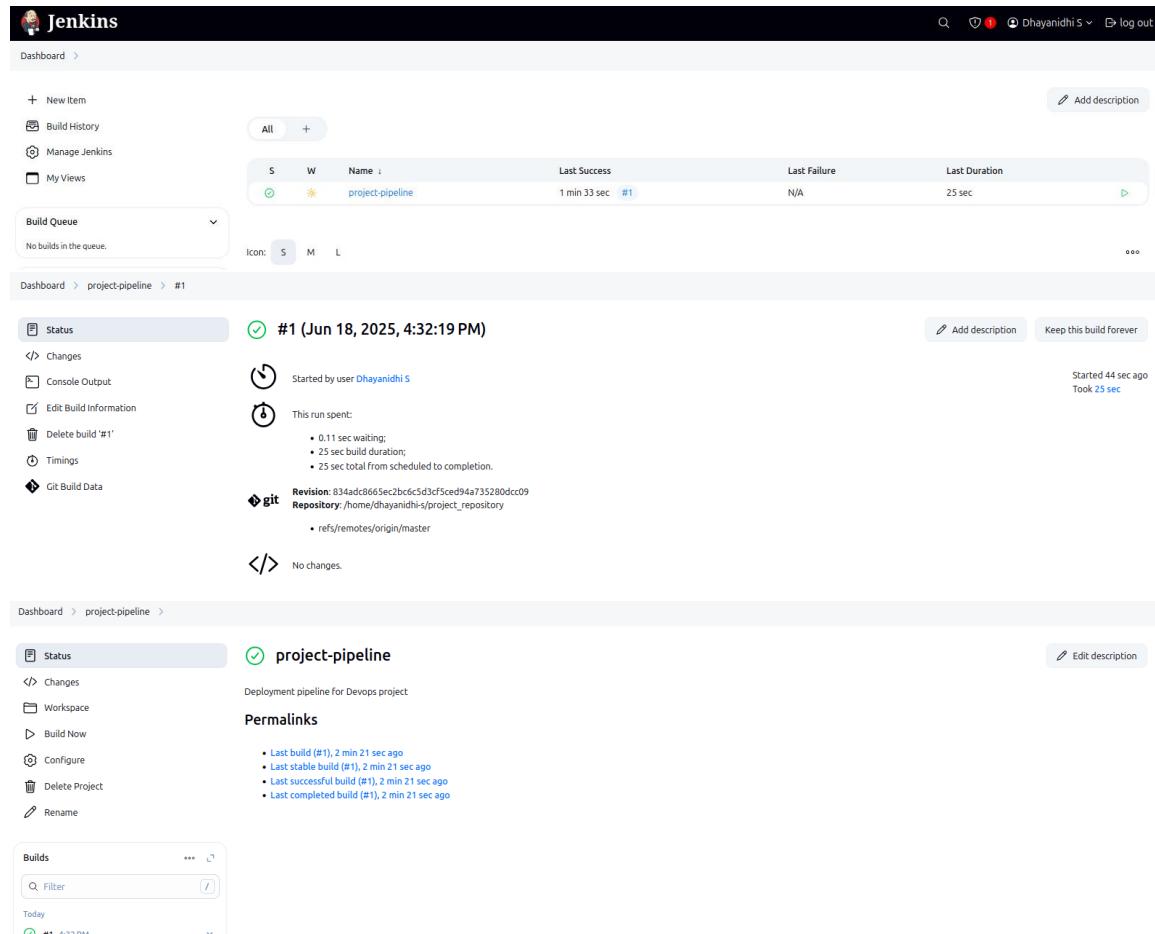
This runs the compiled JAR file after successful build.

---

## 12. Test the Jenkins Pipeline

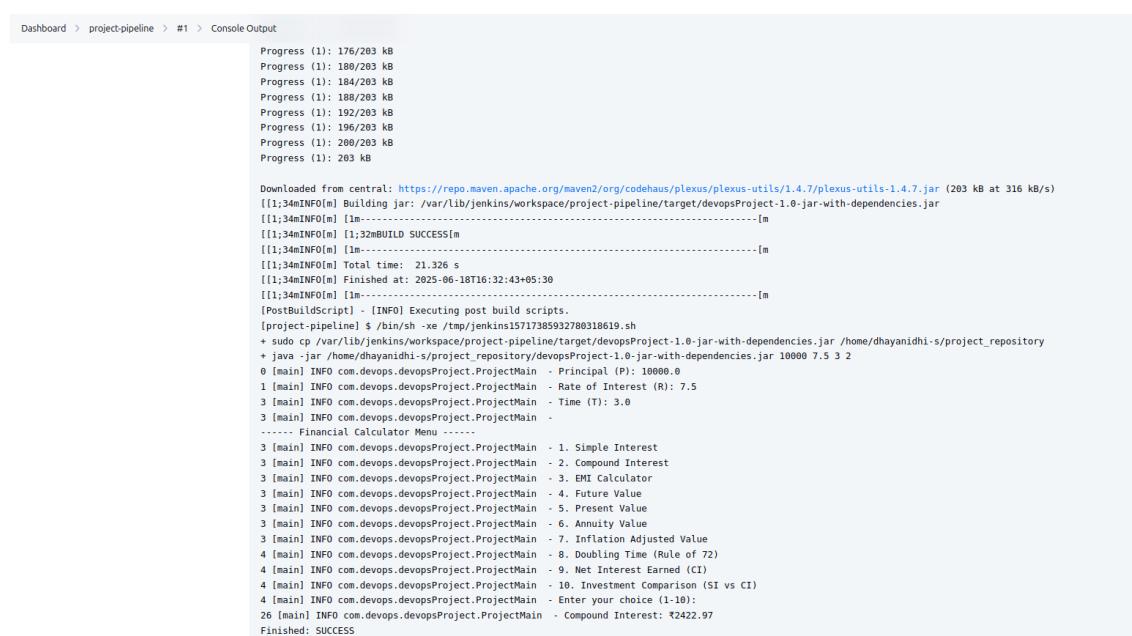
From Jenkins homepage:

→ Click **Build Now** on project



The screenshot shows the Jenkins dashboard for the 'project-pipeline' job. At the top, there's a summary table with columns for Status (S), Work (W), Name, Last Success, Last Failure, and Last Duration. The job 'project-pipeline' is listed with a green status, last success at 1 min 33 sec ago, and a duration of 25 sec. Below this is a 'Build Queue' section indicating 'No builds in the queue.' The main content area shows the build details for '#1' (Jun 18, 2025, 4:32:19 PM). It includes sections for Status (green checkmark), Changes, Console Output, Edit Build Information, Timings, and Git Build Data. The Git section shows revision 834ade8665ec2bcd5d5cf5ced94a735280dc09, repository /home/dhayanidhi-s/project\_repository, and a commit message: refs/remotes/origin/master. Below the build details is a 'Permalinks' section listing the last four builds. A 'Builds' table at the bottom shows the current build (#1) is running.

View real-time logs from **Build History** → **Console Output**.



The screenshot shows the Jenkins console output for build #1. The logs are displayed in real-time, showing the progress of the build. The output includes Maven dependency download logs, the execution of post-build scripts, and the execution of a Java application named 'devopsProject'. The application logs show various financial calculator methods being called, such as Simple Interest, Compound Interest, EMI Calculator, Future Value, Present Value, Annuity Value, Inflation Adjusted Value, Doubling Time, Net Interest Earned, Investment Comparison, and a prompt for user input. The build completed successfully at the end.

```
Progress (1): 176/203 kB
Progress (1): 180/203 kB
Progress (1): 184/203 kB
Progress (1): 188/203 kB
Progress (1): 192/203 kB
Progress (1): 196/203 kB
Progress (1): 200/203 kB
Progress (1): 203 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/1.4.7/plexus-utils-1.4.7.jar (203 kB at 316 kB/s)
[1]:34mINFO[m] Building jar: /var/lib/jenkins/workspace/project-pipeline/target/devopsProject-1.0-jar-with-dependencies.jar
[1]:34mINFO[m] 1m...
[1]:34mINFO[m] 1;32mBUILD SUCCESS[1]
[1]:34mINFO[m] 1m...
[1]:34mINFO[m] Total time: 21.326 s
[1]:34mINFO[m] Finished at: 2025-06-18T16:32:43+05:30
[1]:34mINFO[m] 1m...
[1]:34mINFO[m] [PostBuildScript] - INFO! Executing post build scripts.
[project-pipeline] $ /bin/sh -c /tmp/jenkins15717385932780318619.sh
+ sudo cp /var/lib/jenkins/workspace/project-pipeline/target/devopsProject-1.0-jar-with-dependencies.jar /home/dhayanidhi-s/project_repository
+ java -jar /home/dhayanidhi-s/project_repository/devopsProject-1.0-jar-with-dependencies.jar 10000 7.5 2
0 [main] INFO com.devops.devopsProject.ProjectMain - Principal (P): 10000.0
1 [main] INFO com.devops.devopsProject.ProjectMain - Rate of Interest (R): 7.5
2 [main] INFO com.devops.devopsProject.ProjectMain - Time (T): 3.0
3 [main] INFO com.devops.devopsProject.ProjectMain -
..... Financial Calculator Menu .....
3 [main] INFO com.devops.devopsProject.ProjectMain - 1. Simple Interest
3 [main] INFO com.devops.devopsProject.ProjectMain - 2. Compound Interest
3 [main] INFO com.devops.devopsProject.ProjectMain - 3. EMI Calculator
3 [main] INFO com.devops.devopsProject.ProjectMain - 4. Future Value
3 [main] INFO com.devops.devopsProject.ProjectMain - 5. Present Value
3 [main] INFO com.devops.devopsProject.ProjectMain - 6. Annuity Value
3 [main] INFO com.devops.devopsProject.ProjectMain - 7. Inflation Adjusted Value
4 [main] INFO com.devops.devopsProject.ProjectMain - 8. Doubling Time (Rule of 72)
4 [main] INFO com.devops.devopsProject.ProjectMain - 9. Net Interest Earned (CI)
4 [main] INFO com.devops.devopsProject.ProjectMain - 10. Investment Comparison (SI vs CI)
4 [main] INFO com.devops.devopsProject.ProjectMain - Enter your choice (1-10):
26 [main] INFO com.devops.devopsProject.ProjectMain - Compound Interest: ₹2422.97
Finished: SUCCESS
```

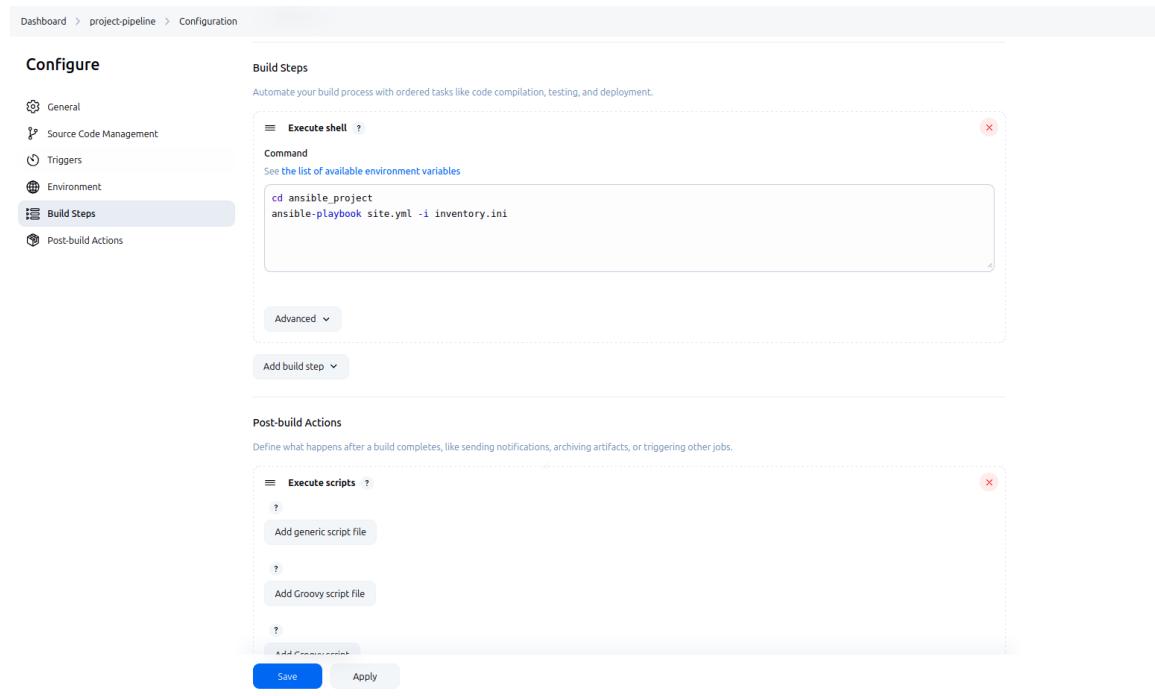
## Automating All Roles By Running Ansible Playbook In Jenkins

In the project so far, Jenkins automates key tasks like performing a maven clean, building the project, and running the generated JAR file during every build. This ensures continuous integration is in place. However, to **streamline the entire DevOps process even further**, we are now taking it a step ahead — by **triggering the Ansible playbook from Jenkins**. This playbook includes all roles such as Maven clean and build, Docker containerization, JUnit testing, Jenkins setup, Minikube deployment, and monitoring configuration using Grafana and Graphite. By doing this, Jenkins not only handles the build but also **automates the full environment setup** with a single trigger, making the pipeline **faster, consistent, and completely hands-free** for end-to-end deployment and monitoring.

### Run the entire pipeline using the Ansible playbook

 Once all roles and their tasks are defined, execute the entire pipeline:

```
ansible-playbook site.yml -i inventory.ini
```



The screenshot shows the Jenkins Pipeline Configuration page. The left sidebar has links for General, Source Code Management, Triggers, Environment, Build Steps (which is selected), and Post-build Actions. The main area is titled "Build Steps" with the sub-instruction "Automate your build process with ordered tasks like code compilation, testing, and deployment." It contains an "Execute shell" step with the command "cd ansible\_project; ansible-playbook site.yml -i inventory.ini". Below this is an "Advanced" dropdown and a "Add build step" button. The "Post-build Actions" section follows, with an "Execute scripts" step containing "Add generic script file", "Add Groovy script file", and "Add downstream". At the bottom are "Save" and "Apply" buttons.

 This will **automatically install, configure, test, deploy, monitor**, and integrate everything defined across all roles in a **single command**.

---

 **Note:** Checking the **Console Output** in Jenkins after a build helps in **identifying and resolving errors** during execution. If a failure occurs while running the Ansible playbook, it's recommended to **review and modify the main.yml file** under each role's tasks/ directory. Ensuring the tasks, paths, modules, and configurations in these files are correct is key to troubleshooting and fixing build or deployment issues efficiently. Make sure you commit and push the changes .

---

Dashboard > project-pipeline > #9 > Console Output

```
ok: [localhost]

TASK [gitclone : Clone local Git project repository] ****
changed: [localhost]

TASK [build : Ensure target directory is removed to avoid permission issues] ***
changed: [localhost]

TASK [build : Build the Java project using Maven] ****
changed: [localhost]

TASK [test : Run JUnit tests using Maven] ****
changed: [localhost]

TASK [dockerize : Copy Dockerfile to workspace] ****
ok: [localhost]

TASK [dockerize : Build Docker image from project] ****
changed: [localhost]

TASK [deploy : Run Docker container from built image] ****
changed: [localhost]

TASK [deploy : Pause to allow container startup] ****
Pausing for 5 seconds
ok: [localhost]

TASK [monitor : Create monitoring directory inside workspace] ****
ok: [localhost]

TASK [monitor : Copy Docker Compose file to monitoring folder] ****
ok: [localhost]

TASK [monitor : Start monitoring stack with Docker Compose] ****
changed: [localhost]

PLAY RECAP ****
localhost : ok=12    changed=7    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0

[PostBuildScript] - [INFO] Executing post build scripts.
Finished: SUCCESS
```

Now, we transition from using a **Freestyle Project** to a **Jenkins Pipeline** for our project. The pipeline approach provides **better automation, flexibility, and control**, allowing us to define the entire CI/CD workflow in a script (`Jenkinsfile`). This ensures a more scalable and manageable way to build, test, and deploy our financial calculator application.

# Creating Jenkins Pipeline with Ansible Integration

---

## 1. Create a Pipeline Project in Jenkins

- ◆ Go to **Jenkins Dashboard -> New Item**
- ◆ Enter a name, eg . **pipeline\_for\_project**.
- ◆ Select “**Pipeline**” as the project type.
- ◆ Click **OK** to create the pipeline project.

 A **Pipeline job** allows to define the complete build process—build, test, deploy, monitor, and rollback—as code in a Jenkinsfile.

---

## 2. Configure the Pipeline Job

After the project is created:

- ◆ Scroll down to the **Build Triggers** section and Check the box  **Poll SCM**.

### What is Poll SCM?

This tells Jenkins to **poll the Git repository** periodically to check for changes (commits, file updates). If changes are found, the pipeline build is triggered automatically.

- In the **Schedule** textbox, type:

H/2 \* \* \* \*

 This means: “Check for changes every 2 minutes, randomized per job to avoid overload.”

**H/2** → Distributes the load evenly by using a hash of the job name.

\* → Every minute/hour/day/week/month.

## Configure the Pipeline Source:

- ◆ In the **Pipeline** section below:
  - Select “**Pipeline script from SCM**”.
  - Choose **Git** under SCM.
  - In the **Repository URL**, enter the path to your local Git repository (e.g., /home/administrator/devops\_repository/).
  - Leave credentials empty (for local access).
  - Set **Script Path** as: Jenkinsfile (Assuming Jenkinsfile is located at the root of the repo.)
- ◆ Click **Save**.

Dashboard > All > pipeline\_for\_project > Configuration

**Configure**

**Triggers**

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Schedule ?

H/2 \* \* \* \*

Would last have run at Monday, June 30, 2025, 8:01:00 PM India Standard Time; would next run at Monday, June 30, 2025, 8:03:00 PM India Standard Time.

Ignore post-commit hooks ?

Trigger builds remotely (e.g., from scripts) ?

Dashboard > pipeline\_for\_project > Configuration

Define your Pipeline using Groovy directly or pull it from source control.

**Configure**

**Pipeline**

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?  
/home/dhayanshi-s/project\_repository

Credentials ?

- none -

+ Add

Advanced

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?  
\*/master

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Script Path ?  
Jenkinsfile

Lightweight checkout ?

Save Apply

💡 Jenkins will now run the Jenkinsfile from Git repository every 2 minutes (if changes are found) and follow the defined stages.

### 📝 3. Create the Jenkinsfile

Inside your Git project directory (e.g., devops\_repository), create a file named Jenkinsfile and add the following:

```

Jenkinsfile
~/project_repository

pipeline {
    agent any

    environment {
        BUILD_PATH = "${env.WORKSPACE}"
        BUILD_NUMBER = "${env.BUILD_NUMBER}"
    }

    stages {
        stage('Build') {
            steps {
                dir('ansible_project') {
                    sh 'ansible-playbook -i inventory site.yml --tags build --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER"'
                }
            }
        }

        stage('Test') {
            steps {
                dir('ansible_project') {
                    sh 'ansible-playbook -i inventory site.yml --tags test --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER"'
                }
            }
        }

        stage('Dockerize') {
            steps {
                dir('ansible_project') {
                    sh 'ansible-playbook -i inventory site.yml --tags dockerize --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER"'
                }
            }
        }

        stage('Deploy') {
            steps {
                dir('ansible_project') {
                    sh 'ansible-playbook -i inventory site.yml --tags deploy --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER"'
                }
            }
        }

        stage('Monitor') {
            steps {
                dir('ansible_project') {
                    sh 'ansible-playbook -i inventory site.yml --tags monitor --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER"'
                }
            }
        }

        stage('Rollback (if needed)') {
            steps {
                dir('ansible_project') {
                    sh 'ansible-playbook -i inventory site.yml --tags rollback --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER"'
                }
            }
        }
    }
}

```

✿ Each stage here corresponds to a role or task in the **Ansible automation**. The playbook (site.yml) is executed with the respective --tags like build, test, etc., and passes required variables (build\_path, build\_number) to make the process dynamic and traceable.

## ✿ 4. Build the Pipeline

To trigger the pipeline No need to manually click "**Build Now**" every time.

- Every 2 minutes, Jenkins checks the Git repository (your local repo path in the config).
- If it detects **any new commit**, Jenkins will **automatically trigger the build** using the latest version of the Jenkinsfile.

So the workflow becomes:

1. You make changes in code or configuration.
2. You commit and push the changes to Git repository.
3. Jenkins polls the repo, sees the new commit, and triggers the entire pipeline automatically.

Jenkins

Dashboard > pipeline\_for\_project > #4 > Pipeline Overview

Run

Graph

Start → Checkout SCM → Build → Test → Dockerize → Deploy → Monitor → Rollback (if...)

Checkout SCM 0.3 sec

Build 8.6 sec

Test 3.8 sec

Dockerize 9.8 sec

Deploy 10 sec

Monitor 5.9 sec

Rollback (if needed) 3.0 sec

Logs

```
ansible-playbook -i inventory.site.yml --tags rollback --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER" 
0 + ansible-playbook -i inventory.site.yml --tags rollback --extra-vars build_path=/var/lib/jenkins/workspace/pipeline_for_project build_number=4
1
2 PLAY [Full DevOps Pipeline] ****
3
4 TASK [Gathering Facts] ****
5 [WARNING]: Platform linux on host localhost is using the discovered Python
6 interpreter at /usr/bin/python3.12, but future installation of another Python
7 interpreter could change the meaning of that path. See
8 https://docs.ansible.com/ansible-
9 core/2.18/reference_appendices/interpreter_discovery.html for more information.
10 ok: [localhost]
11
12 TASK [rollback : Check if rollback is needed] ****
13 ok: [localhost]
14
15 TASK [rollback : Skip rollback because build was successful] ****
16 ok: [localhost] => {
17     "msg": "Build succeeded - skipping rollback."
18 }
19
20 TASK [rollback : End rollback role early if build succeeded] ****
21
22 PLAY RECAP ****
localhost : ok=3    changed=0   unreachable=0   failed=0    skipped=0   rescued=0   ignored=0
```

Manually run by Dhyanidhi S   Started 1 min 15 sec ago   Queued 10 ms   Took 8.6 sec   Changes

Graph

```
graph LR; Start((Start)) --> SCM[Checkout SCM]; SCM --> Build[Build]; Build --> Test[Test]; Test --> Dockerize[Dockerize]; Dockerize --> Deploy[Deploy]; Deploy --> Monitor[Monitor]; Monitor --> Rollback[Rollback (if...)]; Rollback --> End((End))
```

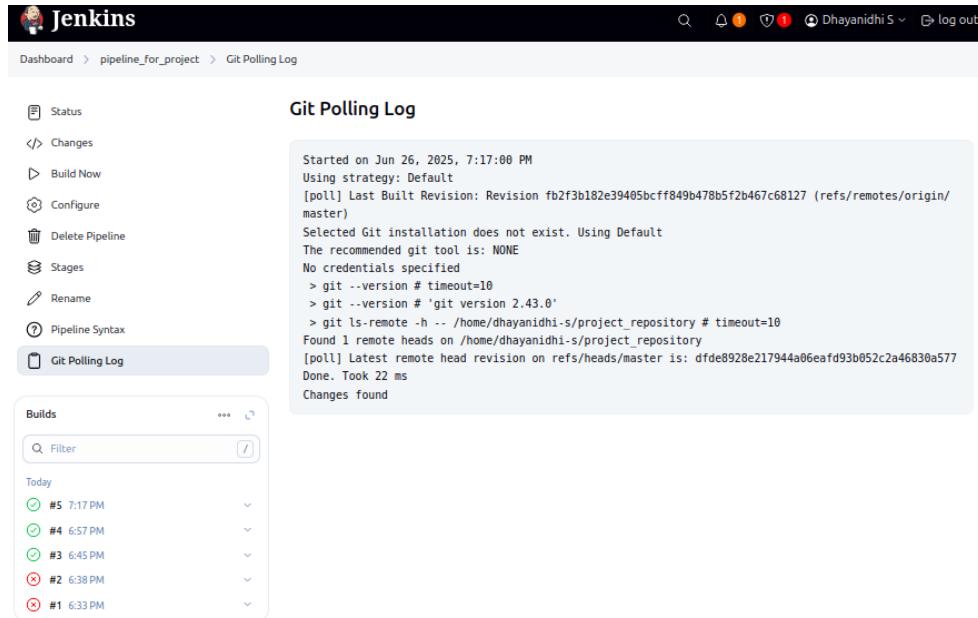
 Jenkins Search 🔍 ⚡ 🛡️ 🚫 Dhayanchi S ✎ Log Out  
Dashboard > pipeline\_for\_project > Stages Build Configure

## Stages

Stage	Last Run	Status
#4	June 26, 2025	18:57 - 58 sec - 1 change
#3	June 26, 2025	18:45 - 8.6 sec - 1 change
#2	June 26, 2025	18:38 - 9.2 sec
#1	June 26, 2025	18:38 - 9.2 sec

## Git Polling and Build Trigger Demonstration

Jenkins is configured to poll the Git repository every 2 minutes for changes. If any new commits are detected, it automatically triggers the pipeline build process. This helps ensure that the latest updates are always tested and deployed without manual intervention.

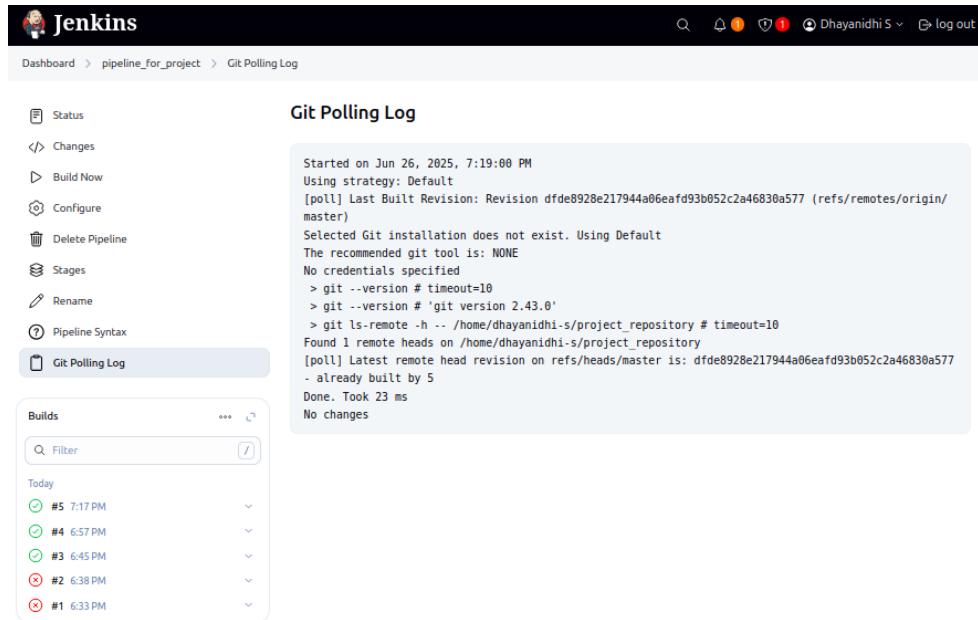


The screenshot shows the Jenkins interface with the 'Git Polling Log' page selected. The log output indicates a successful poll on June 26, 2025, at 7:17:00 PM. It shows the last built revision and the command used to poll the repository. The 'Builds' section lists five builds from today, with the most recent one being successful (#5 at 7:17 PM).

```
Started on Jun 26, 2025, 7:17:00 PM
Using strategy: Default
[poll] Last Built Revision: Revision fb2f3b182e39405bcff849b478b5f2b467c68127 (refs/remotes/origin/master)
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git ls-remote -h -- /home/dhayanidhi-s/project_repository # timeout=10
Found 1 remote heads on /home/dhayanidhi-s/project_repository
[poll] Latest remote head revision on refs/heads/master is: dfde8928e217944a06eaf93b052c2a46830a577
Done. Took 22 ms
Changes found

Builds
...
#5 7:17 PM
#4 6:57 PM
#3 6:45 PM
#2 6:38 PM
#1 6:33 PM
```

If there are no changes, the pipeline remains idle, avoiding unnecessary builds.



The screenshot shows the Jenkins interface with the 'Git Polling Log' page selected. The log output indicates a successful poll on June 26, 2025, at 7:19:00 PM. It shows the last built revision and the command used to poll the repository. The 'Builds' section lists five builds from today, all of which are successful. The log notes that the build was already built by 5 previous runs.

```
Started on Jun 26, 2025, 7:19:00 PM
Using strategy: Default
[poll] Last Built Revision: Revision dfde8928e217944a06eaf93b052c2a46830a577 (refs/remotes/origin/master)
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git ls-remote -h -- /home/dhayanidhi-s/project_repository # timeout=10
Found 1 remote heads on /home/dhayanidhi-s/project_repository
[poll] Latest remote head revision on refs/heads/master is: dfde8928e217944a06eaf93b052c2a46830a577
- already built by 5
Done. Took 23 ms
No changes

Builds
...
#5 7:17 PM
#4 6:57 PM
#3 6:45 PM
#2 6:38 PM
#1 6:33 PM
```

## Updated Playbook and Roles for Smooth Pipeline Execution and Achieving Proper Rollback

```

Open v main.yml ~project_repository/ansible_project/roles/build/tasks
- name: Ensure target directory is removed to avoid permission issues
  file:
    path: "{{ build_path }}/{{ target }}"
    state: absent

- name: Build the Java project using Maven
  command: mvn clean package
  args:
    chdir: "{{ build_path }}"
  register: build_result
  ignore_errors: yes

- name: Create rollback trigger file if build failed
  file:
    path: /tmp/build_failed
    state: touch
  when: build_result.rc != 0

- name: Remove rollback trigger file if build succeeded
  file:
    path: /tmp/build_failed
    state: absent
  when: build_result.rc == 0

Open v main.yml ~project_repository/ansible_project/roles/deploy/tasks
...
- name: Remove old financial-calc container if it exists
  docker_container:
    name: financial-calc
    state: absent
    force_kill: true
    purge_networks: true
  ignore_errors: yes # In case container doesn't exist

- name: Run Docker container from built image
  docker_container:
    name: financial-calc
    image: financial-calc:build-{{ build_number }} # We use build-specific tag
    state: started
    restart_policy: always
    ports:
      - "8182:8080"
    detach: true

- name: Pause to allow container startup
  pause:
    seconds: 5

Open v main.yml ~project_repository/ansible_project/roles/dockerize/tasks
- name: Copy Dockerfile to workspace
  copy:
    src: /home/dhayanidhi-s/Documents/dockerProjects/Dockerfile
    dest: "{{ build_path }}/{{ Dockerfile }}"

- name: Build Docker image with unique version tag and 'latest'
  command: >
    docker build
    -t financial-calc:build-{{ build_number }}
    -t financial-calc:latest
  ...
  args:
    chdir: "{{ build_path }}"

- name: Save successful build number to file
  copy:
    content: "{{ build_number }}"
    dest: "{{ build_path }}/{{ last_success_build }}"

- name: Tag current successful image as 'previous'
  shell: docker tag financial-calc:build-{{ build_number }} financial-calc:previous

Open v main.yml ~project_repository/ansible_project/roles/monitor/tasks
- name: Create monitoring directory inside workspace
  file:
    path: "{{ build_path }}/{{ monitoring }}"
    state: directory
    mode: '0755'

- name: Copy Docker Compose file to monitoring folder
  copy:
    src: docker-compose.yml # This looks inside roles/monitor/files/
    dest: "{{ build_path }}/{{ monitoring }}/{{ docker-compose }}"

- name: Start monitoring stack with Docker Compose
  command: docker-compose up -d
  args:
    chdir: "{{ build_path }}/{{ monitoring }}"

Open v main.yml ~project_repository/ansible_project/roles/rollback/tasks
...
- name: Check if rollback is needed
  stat:
    path: /tmp/build_failed
  register: rollback_flag

- name: Skip rollback because build was successful
  debug:
    msg: "Build succeeded - skipping rollback."
  when: not rollback_flag.stat.exists

- name: End rollback role early if build succeeded
  meta: end_play
  when: not rollback_flag.stat.exists

- name: Check if previous Docker image exists
  shell: docker images -q financial-calc:previous
  register: previous_image
  changed_when: false

- name: Remove broken container (if any)
  docker_container:
    name: financial-calc
    state: absent
    force_kill: true
    ignore_errors: yes

- name: Rollback to previous Docker image (if it exists)
  docker_container:
    name: financial-calc
    image: financial-calc:previous
    state: started
    restart_policy: always
    ports:
      - "8182:8080"
  when: previous_image.stdout != ""

Open v main.yml ~project_repository/ansible_project/roles/test/tasks
- name: Run JUnit tests using Maven
  command: mvn test
  args:
    chdir: "{{ build_path }}"
  register: test_result
  ignore_errors: yes # Allow Ansible to continue even if tests fail

- name: Create rollback trigger file if tests failed
  file:
    path: /tmp/build_failed
    state: touch
  when: test_result.rc != 0

```



## Rollback Implementation – Auto-Recovery

---

### 1. Install Version Number Plugin

Go to **Manage Jenkins → Plugin Manager → Available**

→ **Install Version Number Plugin**

Helps manage versions like 0.2.\$BUILD\_NUMBER for rollback.

---

### 2. Intentionally Break a Test Case

→ In Git repo, modify ProjectMainTest.java to fail a test.

Correct Test Case (with testValidCompoundInterest result as 1025)

```
@Test
public void testValidCompoundInterest() {
    double result = ProjectMain.compoundInterest(10000, 5, 2);
    assertEquals(1025.0, result, 1.0); // compound = 10000*(1+0.05)^2 - 10000
= 1025
}
```

Intentionally making the Test case to Fail by changing the value as 125

```
@Test
public void testValidCompoundInterest() {
    double result = ProjectMain.compoundInterest(10000, 5, 2);
    assertEquals(125.0, result, 1.0); // compound = 10000*(1+0.05)^2 - 10000
= 1025
}
```

---

### 3. Update Build Commands or Jenkinsfile for Versioning

---

### 4. Trigger Jenkins Build and Observe

→ Commit the change to trigger a build failure:

→ Watch for:

Build Failure

Rollback Script Execution

Jenkins has now auto-recovered from the failed build using rollback.

Dashboard > pipeline\_for\_project > #6 > Pipeline Overview

Search

- Checkout SCM 0.34 sec
- Build 12 sec
- Test 8.5 sec**
- Dockerize 8.7 sec
- Deploy 10 sec
- Monitor 5.4 sec
- Rollback (if needed) 6.8 sec

```
ansible-playbook -i inventory site.yml -tags test --extra-vars "build_path=$BUILD_PATH build_number=$BUILD_NUMBER" -vvv
[Pipeline] {
  [Pipeline] sh
+ ansible-playbook -i inventory site.yml --tags rollback --extra-vars build_path=/var/lib/jenkins/workspace/pipeline_for_project build_number=7
PLAY [Full DevOps Pipeline] *****
TASK [Gathering Facts] *****
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [rollback : Check if rollback is needed] *****
ok: [localhost]

TASK [rollback : Skip rollback because build was successful] *****
skipping: [localhost]

TASK [rollback : End rollback role early if build succeeded] *****
skipping: [localhost]

TASK [rollback : Check if previous Docker image exists] *****
ok: [localhost]

TASK [rollback : Remove broken container (if any)] *****
changed: [localhost]

TASK [rollback : Rollback to previous Docker image (if it exists)] *****
changed: [localhost]

PLAY RECAP *****
localhost          : ok=5    changed=2    unreachable=0    failed=0    skipped=1
rescued=0   ignored=0

[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

8.3 sec

## Failure of Test case

Dashboard > pipeline\_for\_project > #7

```
Running in /var/lib/jenkins/workspace/pipeline_for_project/ansible_project
[Pipeline] {
  [Pipeline] sh
+ ansible-playbook -i inventory site.yml --tags rollback --extra-vars build_path=/var/lib/jenkins/workspace/pipeline_for_project build_number=7
PLAY [Full DevOps Pipeline] *****
TASK [Gathering Facts] *****
[WARNING]: Platform linux on host localhost is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.18/reference_appendices/interpreter_discovery.html for more information.
ok: [localhost]

TASK [rollback : Check if rollback is needed] *****
ok: [localhost]

TASK [rollback : Skip rollback because build was successful] *****
skipping: [localhost]

TASK [rollback : End rollback role early if build succeeded] *****
skipping: [localhost]

TASK [rollback : Check if previous Docker image exists] *****
ok: [localhost]

TASK [rollback : Remove broken container (if any)] *****
changed: [localhost]

TASK [rollback : Rollback to previous Docker image (if it exists)] *****
changed: [localhost]

PLAY RECAP *****
localhost          : ok=5    changed=2    unreachable=0    failed=0    skipped=1
rescued=0   ignored=0

[Pipeline] }
[Pipeline] // dir
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

## Rolling back to Previous Build

## Migration from Local Git Repository to GitHub

In the initial stages of the project setup, we created a **local Git server and bare repository** (`git@localhost:project_repository.git`) to manage version control and integrate with Jenkins. This worked effectively in a closed environment, allowing us to test Git-based automation like **SCM polling** and **CI/CD triggers**.

However, to make the project more **scalable, collaborative, and accessible remotely**, **migrate the Git repository to GitHub**.

---

### What We Changed

#### 1. Jenkins Configuration:

- Replace or use Github repository URL in place of Local repository path . Update under
  - Freestyle Project: Git → Repository URL
  - Pipeline Project: Script from SCM → Repository URLJenkins is now connected to the **GitHub repository** instead of the local Git server.

#### 2. SCM Trigger:

- No change needed in trigger logic (Poll SCM still works), but it now checks GitHub instead of local Git.
- Optionally, **GitHub Webhooks** can be configured for real-time triggering instead of polling.

#### 3. Push Workflow:

- Code is now committed and pushed to **GitHub** .
- 

### How Jenkins Works After Migration

- It continues to:
  - **Poll GitHub** every 2 minutes for changes (based on H/2 \* \* \* \* schedule).
  - **Trigger builds** automatically if new commits are detected.
  - **Run Ansible playbooks** for build, test, dockerization, deployment, monitoring, and rollback.
- No disruption to your existing Ansible roles or pipeline logic — only the **Git source URL changed**.

## Why We Switched to GitHub

Reason	Benefit
 Remote Access	Access code from anywhere, not just the local machine
 Collaboration	Easily share and collaborate with team members
 Better Security	GitHub offers role-based access, SSH key integration
 CI/CD Integration	Jenkins works seamlessly with GitHub for SCM polling or webhook triggers
 Cloud Backup	Prevents data loss by storing your code in the cloud

---

## Conclusion

This project presents a comprehensive end-to-end DevOps pipeline designed to automate the entire lifecycle of a Java-based CLI application. It starts with setting up a Maven project for building and managing dependencies, followed by the integration of **JUnit** for automated testing to ensure code quality. The application is then **containerized using Docker**, making it portable and consistent across environments. This Dockerized application is deployed to a **Kubernetes (Minikube)** cluster, offering scalability and efficient resource management.

To enable seamless automation, **Ansible** is used for orchestrating various stages like build, test, dockerization, deployment, monitoring, and rollback. A centralized site.yml playbook and well-structured role-based automation scripts ensure that each task is repeatable, maintainable, and modular. These Ansible playbooks are executed through **Jenkins**, which acts as the heart of the CI/CD pipeline. The Jenkins pipeline is configured with **SCM polling**, so every new code push triggers the entire process automatically. This minimizes manual intervention and ensures rapid and reliable delivery of updates.

Furthermore, robust **monitoring using Collectd, Graphite, and Grafana** is integrated to visualize system metrics in real time. This allows for proactive identification of issues in deployed environments. In case of a failure during build or deployment, **automatic rollback** mechanisms ensure stability by reverting to the last successful version of the application. Altogether, this project demonstrates not only the technical integration of industry-standard DevOps tools but also emphasizes the importance of automation, resilience, and observability in modern software delivery workflows.

 **DevOps as a practice** bridges the gap between development and operations teams by promoting collaboration, continuous integration, and delivery. It helps organizations accelerate product releases, improve deployment frequency, and ensure system reliability. By embracing DevOps principles, this project reflects how automation and toolchain integration can drastically reduce deployment risks, enhance feedback loops, and maintain high-quality software standards at scale

---

## GITHUB LINK

<https://github.com/Dhaya09/DevOpsProject.git>