

Software Production Engineering

CI/CD Pipeline on Calculator Program

Manav Ketan Desai, MT2019060

May 9, 2020

Contents

1. Introduction.....	3
2. Source Code and Output.....	3
3. Software Development Life Cycle (SDLC)	5
3.1 Source Control Management (SCM).....	6
3.2 Build and Test	6
3.3 Continuous Integration (CI)	7
3.3.1 Jenkins Installation.....	7
3.3.2 Jenkins Pipeline	9
3.3.3 Jenkins Pipeline Job 1: calculator-maven-build-pipeline.....	10
3.4 Continuous Delivery	12
3.4.1 Docker Installation.....	12
3.4.2 Building and Publishing Images	13
3.4.3 Jenkins Pipeline Job 2: calculator-build-publish-image	13
3.5 Continuous Deployment	16
3.5.1 Rundeck Installation	16
3.5.2 Creating Rundeck Project and Job	17
3.5.3 Jenkins Pipeline Job 3: calculator-deploy-image	20
3.6 Monitoring.....	23
3.6.1 Installing the Elastic Stack	23
3.6.2 Configuring Filebeat	24
3.6.3 Visualizing Logs in Kibana.....	27
4. Conclusion	30
5. References	30

1. Introduction

In this report, I aim to automate the development, testing and deployment of a calculator program by creating a CI/CD pipeline on it. The calculator program developed here is a Spring Boot project hosted on a Tomcat web server running in a Docker container. The calculator currently supports the four basic mathematical operations and can be accessed as a service. The service can be called via GET request either through curl or through a web browser.

The CI/CD pipeline is built using the following tools:

1. Development: IntelliJ IDEA, Git, Github
2. Testing: JUnit
3. Integration: Apache Maven, Jenkins
4. Delivery: Docker, Docker hub, Jenkins
5. Deployment: Docker, Rundeck, Jenkins
6. Monitoring: Elastic Stack

The entire source code can be found here: <https://github.com/mkd1997/devops-on-calculator>

2. Source Code and Output

The source code is divided into 2 layers – Controller and Model. The Controller layer has the CalculatorController class that is responsible for accepting the requests from users and calling the necessary operation function in the Model layer depending on the request parameters.

```
package com.mkd.devopsdemo.Controllers;

import com.mkd.devopsdemo.Models.CalculatorModel;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
public class CalculatorController {

    @GetMapping("/calculate/{leftOperand}/{rightOperand}/{operator}")
    public ResponseEntity<String> index(@PathVariable Integer leftOperand, @PathVariable String operator, @PathVariable Integer rightOperand) {

        CalculatorModel calculator = new CalculatorModel();
        double res = 0;
        switch (operator) {
            case "+": res = calculator.add(leftOperand, rightOperand);
                break;
            case "-": res = calculator.subtract(leftOperand, rightOperand);
                break;
            case "*": res = calculator.multiply(leftOperand, rightOperand);
                break;
            case "/": res = calculator.divide(leftOperand, rightOperand);
                break;
        }
        return new ResponseEntity<>( "Your result is " + res, HttpStatus.OK);
    }
}
```

The CalculatorModel class implements the four basic functions of the calculator program.

```
package com.mkd.devopsdemo.Models;

/**
 * Caculator
 */
public class CalculatorModel {
    public int add(int x, int y) {
        return (x + y);
    }

    public int subtract(int x, int y) {
        return (x - y);
    }

    public int multiply(int x, int y) {
        return (x * y);
    }

    public double divide(int x, int y) {
        double quotient;
        try {
            quotient = x / y;
        } catch (Exception e) {
            quotient = Double.MAX_VALUE;
        }
        return quotient;
    }

    public int modulus(int x, int y) {
        return (x % y);
    }
}
```

The Spring Boot application is started by running the Main class.

```
package com.mkd.devopsdemo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CalculatorDriver {
    public static void main(String[] args) {
        SpringApplication.run(CalculatorDriver.class, args);
    }
}
```

I am here using the Embedded Tomcat server that is present in all jar packages created using Spring Boot framework. Here, I configure the Embedded Tomcat server to run on port 8088. Since I will be monitoring Tomcat access logs using Elastic Search later, I here configure Tomcat access logs to be generated in /var/log/tomcat directory.

```
server.port=8088
server.tomcat.accesslog.enabled=true
server.tomcat.accesslog.directory=/var/log/tomcat
server.tomcat.accesslog.suffix=.log
server.tomcat.accesslog.prefix=access_log
server.tomcat.accesslog.file-date-format=.yyyy-MM-dd
```

The output of the program, as seen using curl and Google Chrome web browser is shown below.



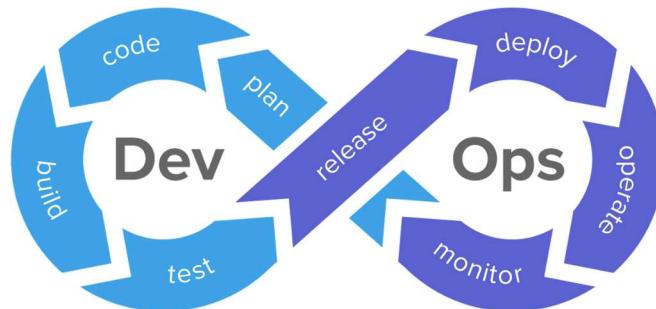
The screenshot shows two windows. The top window is a terminal window titled 'mkd1997@mkd-ubuntu: ~' with a gray header bar and three colored window control buttons (yellow, green, red) in the top right. The terminal has a light gray background and a dark gray border. It displays the following text:

```
File Edit View Search Terminal Help
curl http://localhost:8088/calculate/100/200/*
Your result is 20000.0%
```

The bottom window is a web browser window with a white background and a thin gray border. The address bar at the top says 'localhost:8088/calculate/500/600/'. Below the address bar is a navigation bar with links like 'Apps', 'Machine Learning', 'Python', etc. The main content area displays the text 'Your result is 1100.0'.

3. Software Development Life Cycle (SDLC)

The various stages of Software Development Life Cycle (SDLC) are automated using various DevOps tools. The stages, and how are the automated, is described below:



3.1 Source Control Management (SCM)

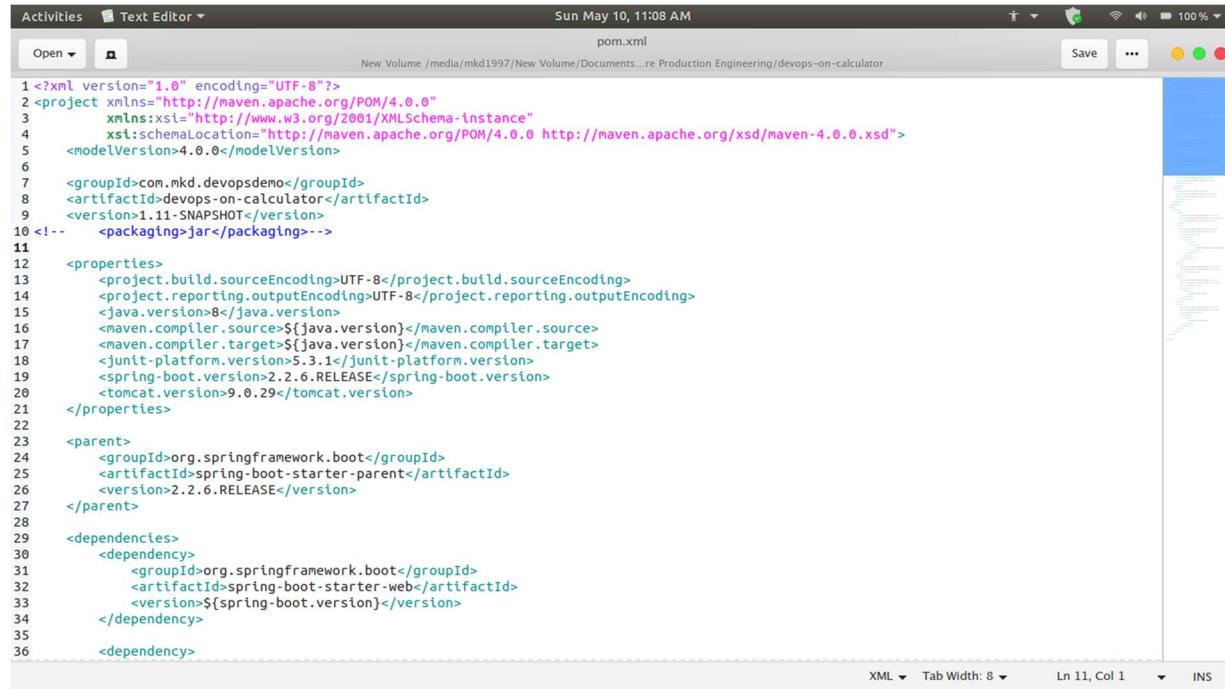
Source control refers to the practice of tracking and managing changes to code. Source control management (SCM) systems provide a running history of code development and help to resolve conflicts when merging contributions from multiple sources. I use Git as the SCM here. Github is an online repository hosting service.

The development of this program has been done incrementally. Some of the important increments done are listed below. Rest can be seen at <https://github.com/mkd1997/devops-on-calculator/commits/master>:

1. Add addition and subtraction operations.
2. Add multiplication and division operations.
3. Convert to Spring Boot application.
4. Enable tomcat access logging.

3.2 Build and Test

In the calculator program, Apache Maven is responsible for managing dependencies and building the project. It is Maven who finally outputs the SNAPSHOT jar of the project that has the compiled classes along with other classes the project depends on. The pom.xml file of the project is shown here.



A screenshot of a Mac OS X Text Editor window titled "pom.xml". The window shows the XML configuration for a Maven project. The code is color-coded, with tags in purple and values in black. The XML includes details about the project's group ID, artifact ID, version, packaging (jar), properties (source and target Java versions, compiler settings, and dependency management), and parent project information. The code spans lines 1 through 36.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>com.mkd.devopsdemo</groupId>
8   <artifactId>devops-on-calculator</artifactId>
9   <version>1.11-SNAPSHOT</version>
10  <!-- <packaging>jar</packaging>-->
11
12  <properties>
13    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
14    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
15    <java.version>8</java.version>
16    <maven.compiler.source>${java.version}</maven.compiler.source>
17    <maven.compiler.target>${java.version}</maven.compiler.target>
18    <junit.platform.version>5.3.1</junit-platform.version>
19    <spring-boot.version>2.2.6.RELEASE</spring-boot.version>
20    <tomcat.version>9.0.29</tomcat.version>
21  </properties>
22
23  <parent>
24    <groupId>org.springframework.boot</groupId>
25    <artifactId>spring-boot-starter-parent</artifactId>
26    <version>2.2.6.RELEASE</version>
27  </parent>
28
29  <dependencies>
30    <dependency>
31      <groupId>org.springframework.boot</groupId>
32      <artifactId>spring-boot-starter-web</artifactId>
33      <version>${spring-boot.version}</version>
34    </dependency>
35
36    <dependency>
```

JUnits is a unit testing framework that is used to write unit test cases. The maven-surefire-plugin runs the test classes at maven build time. The test class for the calculator project is shown here.

```
package com.mkd.devopsdemo;

import com.mkd.devopsdemo.Models.CalculatorModel;

import static org.junit.jupiter.api.Assertions.*;

class CalculatorModelTest {

    @org.junit.jupiter.api.Test
    void addWithInverseShouldReturnZero() {
        CalculatorModel calcTest = new CalculatorModel();
        assertEquals(0, calcTest.add(-5, 5), "-5 + 5 should be 0");
    }

    @org.junit.jupiter.api.Test
    void subtractionOfEqualNumbersShouldBeZero() {
        CalculatorModel calcTest = new CalculatorModel();
        assertEquals(0, calcTest.subtract(5, 5), "5 - 5 should be 0");
    }
}
```

To build the project after running the test cases use:

```
> mvn clean package -U
```

To build the project without running the tests, use:

```
> mvn clean package -U -DskipTests
```

3.3 Continuous Integration (CI)

Continuous Integration (CI) refers to the practice of integrating code changes with the existing code as and when it is written. This includes building the project and running the test cases too automatically as described above. Jenkins is the tool that I have used for Continuous Integration. It keeps watching the SCM system for changes in the code and builds it as and when it detects the changes. Apache Maven is integrated with Jenkins so that Jenkins can trigger maven builds.

3.3.1 Jenkins Installation

To install Jenkins, follow the steps given below:

1. Download and install the necessary GPG key

```
> wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

2. Add the necessary repository

```
> sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

3. Add the universe repository

```
> sudo add-apt-repository universe
```

4. Update apt

```
> sudo apt update
```

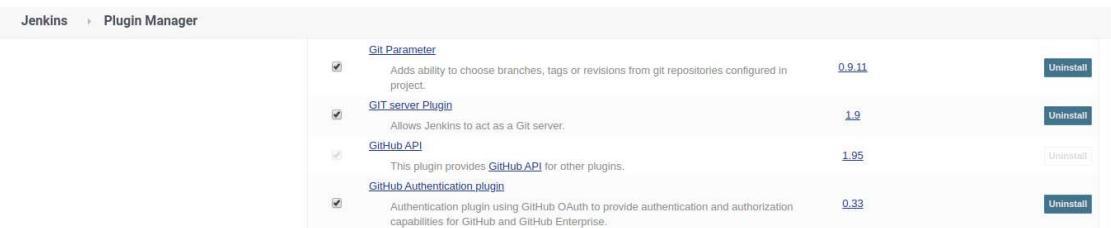
5. Install Jenkins

```
> sudo apt-get install jenkins -y
```

6. Start Jenkins

```
> sudo systemctl start jenkins
```

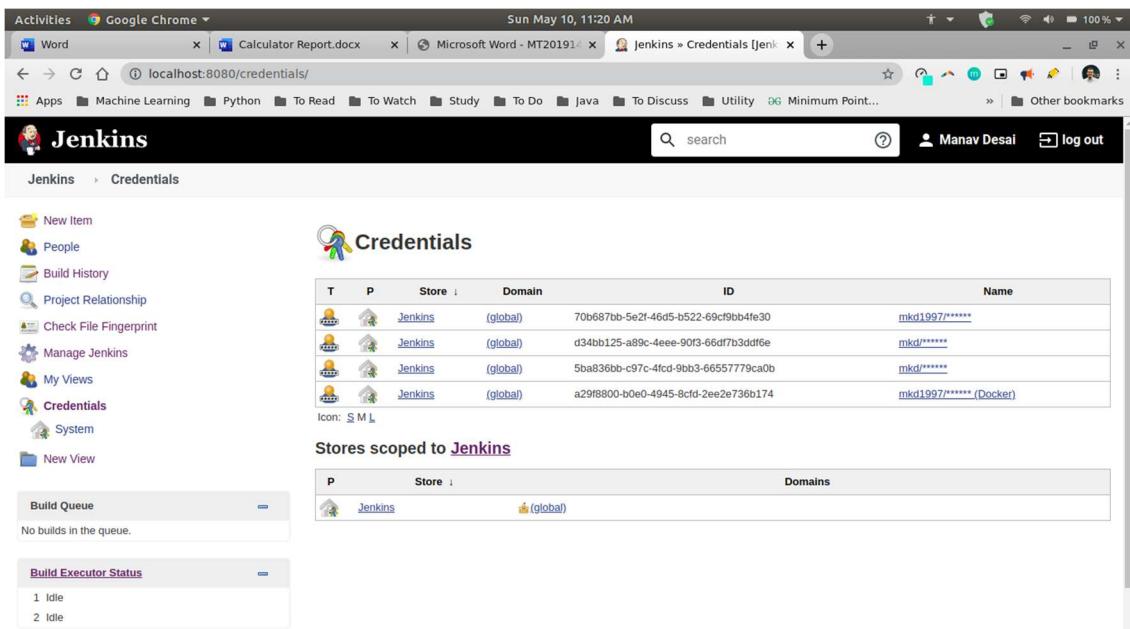
7. Install git, Maven, JUnit plugin by going to Manage Jenkins -> Manage Plugins



The screenshot shows the Jenkins Plugin Manager interface. It lists several GitHub-related plugins that are currently installed:

Plugin	Version	Action
Git Parameter	0.9.11	Uninstall
GIT server Plugin	1.9	Uninstall
GitHub API	1.95	Uninstall
GitHub Authentication plugin	0.33	Uninstall

8. Add git credentials to Jenkins by going to Credentials.



The screenshot shows the Jenkins Credentials management page. On the left, there's a sidebar with navigation links like New Item, People, Build History, etc. The main area is titled "Credentials" and displays a table of stored credentials:

T	P	Store	Domain	ID	Name
Jenkins	(global)	70b687b0-5e2f-46d5-b522-69cf9bb4fe30	mkd1997*****		
Jenkins	(global)	d34bb125-a89c-4eee-90f3-66d7b3dd16e	mkd*****		
Jenkins	(global)	5ba836bb-c97c-4fcf-9bb3-66557779ca0b	mkd*****		
Jenkins	(global)	a29f8800-b0e0-4945-8cf0-2ee2e736b174	mkd1997***** (Docker)		

Below the table, there's a section titled "Stores scoped to Jenkins" which shows a single entry:

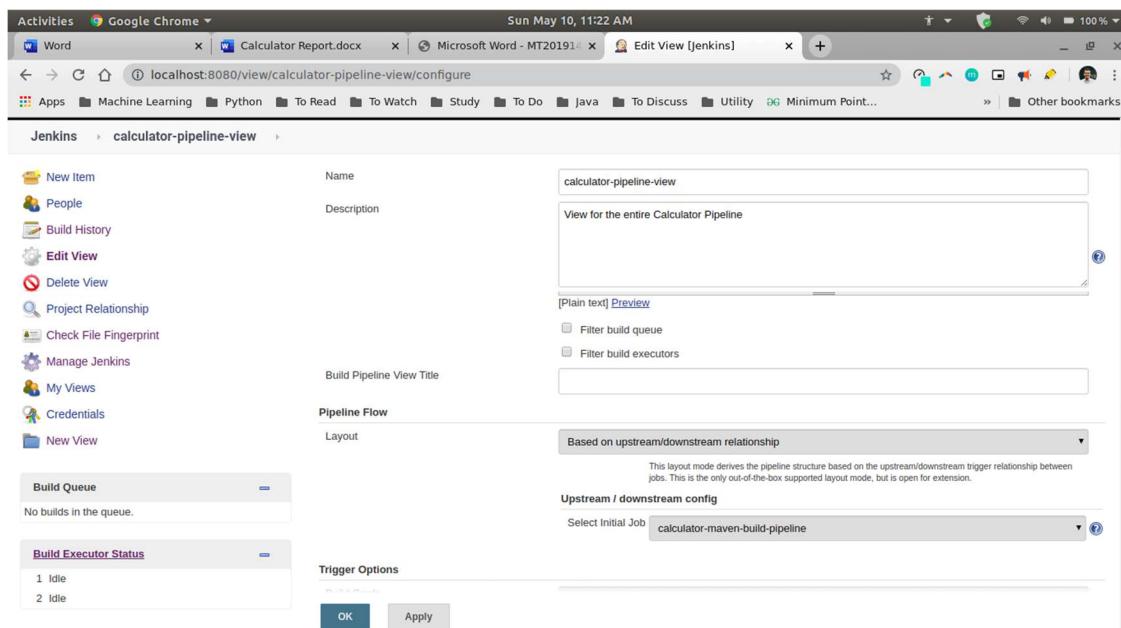
P	Store	Domains
Jenkins	(global)	

3.3.2 Jenkins Pipeline

A Jenkins pipeline gives us a graphical view of the various steps of a CI/CD pipeline. It allows us to link different Jenkins jobs and allows us to check their progress during execution.

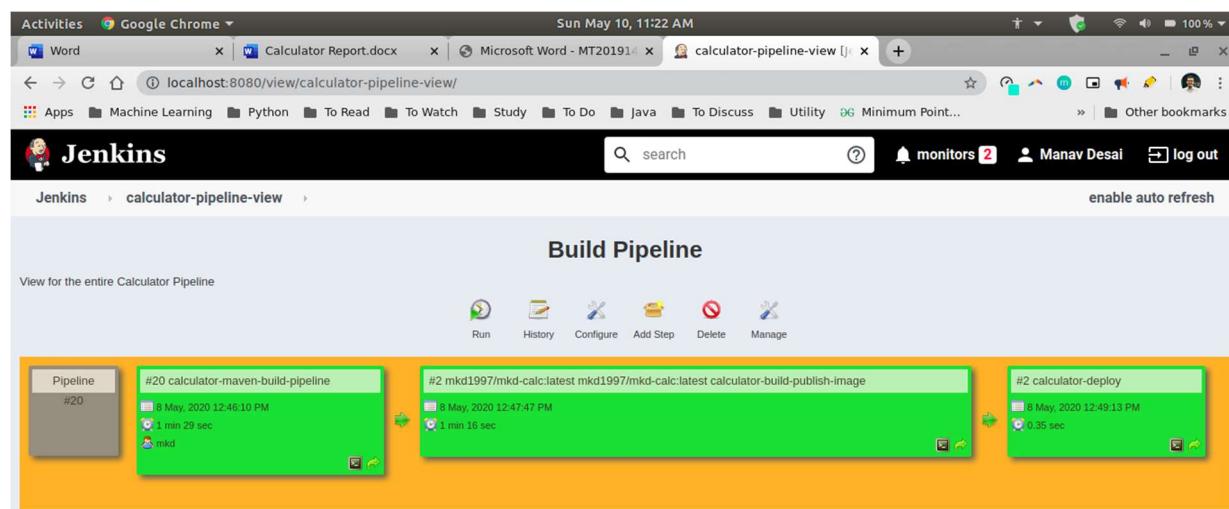
To create Jenkins pipeline, follow the given steps:

1. Click on the ‘+’ sign above the list of Jenkins projects on the main page of Jenkins.
2. Provide a pipeline view name, an optional description and the initial job.



3. Click OK.

The final pipeline is shown below.

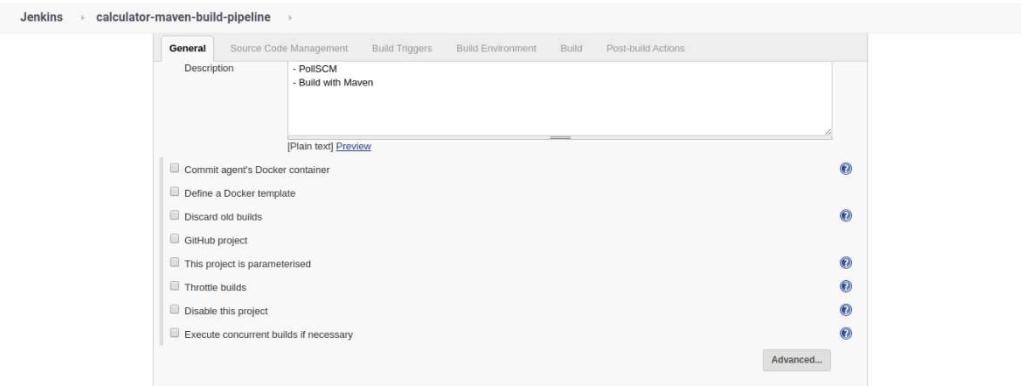


3.3.3 Jenkins Pipeline Job 1: calculator-maven-build-pipeline

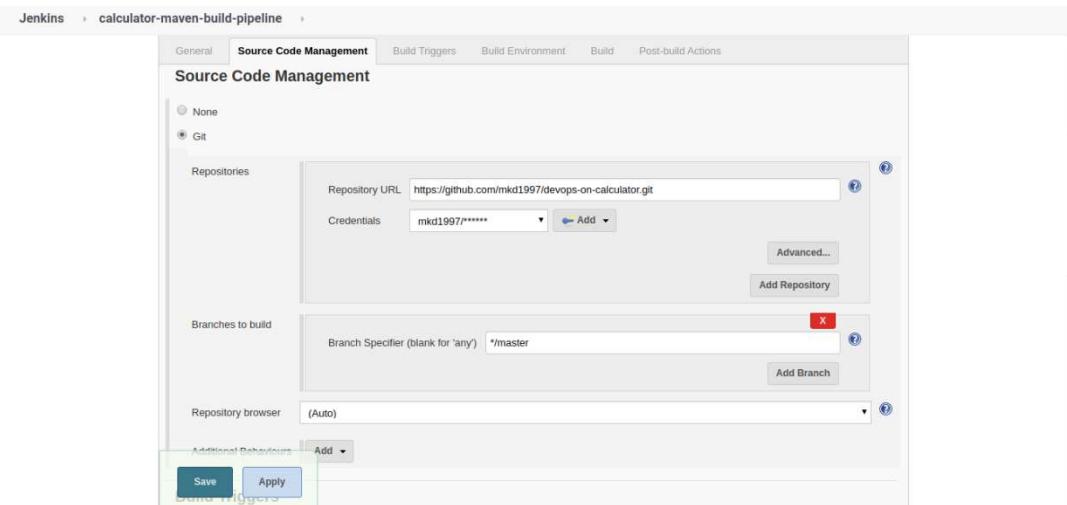
This Jenkins project is responsible for polling the Github repository every 15 minutes and checking for code changes. As soon as it finds that new code has been pushed, it pulls the code into Jenkins workspace and starts a maven build with the specified goal. The goal specified here is clean package –U which also runs the JUnit tests.

To create this project, follow the given steps:

1. Create a new freestyle project and do the following configuration.
 - a. General configuration



- b. Source Code Management



c. Build Triggers

The screenshot shows the 'Build Triggers' configuration page for a Jenkins job named 'calculator-maven-build-pipeline'. The 'Poll SCM' option is selected, and the schedule is set to 'H/15 * * * *'. A note at the bottom indicates the next run will be on Sunday, May 10, 2020, at 11:22:16 AM IST.

d. Build

The screenshot shows the 'Build' configuration page for the same Jenkins job. It includes a single build step: 'Invoke top-level Maven targets' with the goal 'clean package -U'.

2. Click Save.

Console output of the calculator-maven-build-pipeline is shown below

The screenshot shows the Jenkins 'Console Output' page for build #20 of the 'calculator-maven-build-pipeline'. The log output is as follows:

```
Started by user Manav Desai
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/calculator-maven-build-pipeline
using credential 70b687bb-5e2f-46d5-b522-69cf9bb4fe30
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/mkd1997/devops-on-calculator.git # timeout=10
Fetching upstream changes from https://github.com/mkd1997/devops-on-calculator.git
> git --version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --tags --progress -- https://github.com/mkd1997/devops-on-calculator.git +refs/heads/*:refs/remotes/origin/*
# timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 279b69694ef14d672a117de7729eb3059a04308b (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 279b69694ef14d672a117de7729eb3059a04308b # timeout=10
Commit message: "dockerfile changed"
> git rev-list --no-walk 279b69694ef14d672a117de7729eb3059a04308b # timeout=10
[calculator-maven-build-pipeline] $ mvn -f pom.xml clean package -U
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$ReflectUtils$1
(file:/usr/share/maven/lib/guice.jar) to method
ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
Please consider reporting this to the maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
```

```

[INFO] :34mINFO[m]
[INFO] :34mINFO[m] -----
[INFO] :34mINFO[m] T E S T S
[INFO] :34mINFO[m] -----
[INFO] :34mINFO[m] Running com.mkd.devopsdemo.flmCalculatorModelTestflm
[INFO] :34mINFO[m] fl[0;1;32mTests run: fl[0;1;32m2fl[m, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.006 s - in
com.mkd.devopsdemo.flmCalculatorModelTestflm
[INFO] :34mINFO[m]
[INFO] :34mINFO[m] Results:
[INFO] :34mINFO[m]
[INFO] :34mINFO[m] fl[0;1;32mTests run: 2, Failures: 0, Errors: 0, Skipped: 0fl[m
[INFO] :34mINFO[m]
[INFO] :34mINFO[m]
[INFO] :34mINFO[m] fl[0;1;32mmaven-jar-plugin:3.2.0:jarfl[m fl[1m(default-jar)fl[m @ fl[36mdevops-on-calculatorfl[0;1m --
-fl[m
[INFO] :34mINFO[m] Building jar: /var/lib/jenkins/workspace/calculator-maven-build-pipeline/target/devops-on-calculator-
1.11-SNAPSHOT.jar
[INFO] :34mINFO[m]
[INFO] :34mINFO[m] fl[1m-- fl[0;32mspring-boot-maven-plugin:2.2.6.RELEASE:repackagefl[m fl[1m(repackage)fl[m @ fl[36mdevops-on-
calculatorfl[0;1m --fl[m
[INFO] :34mINFO[m] Replacing main artifact with repackaged archive
[INFO] :34mINFO[m] fl[1m-- fl[m
[INFO] :34mINFO[m] fl[1;32mBUILD SUCCESSfl[m
[INFO] :34mINFO[m] fl[1m-- fl[m
[INFO] :34mINFO[m] Total time: 51.558 s
[INFO] :34mINFO[m] Finished at: 2020-05-08T12:47:39+05:30
[INFO] :34mINFO[m] fl[1m-- fl[m
Triggering a new build of calculator-build-publish-image
Finished: SUCCESS

```

Page generated: 10-May-2020 11:28:07 IST REST API Jenkins ver. 2.222.1

3.4 Continuous Delivery

A deliverable in Software Engineering is an artifact that is ready to be delivered to the client. It thus is the end product of a SDLC. Continuous Delivery (CD) is the practice of generating deliverable as soon as code changes happen. CD requires that CI pipeline be in place first. Hence, CI is a prerequisite of CD.

Here, the deliverable is in the form of a docker image. The image consists of everything that our project requires to run including OS, OpenJDK and Tomcat server.

The tools used by me for creating a CD pipeline are Docker and Jenkins.

3.4.1 Docker Installation

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and deploy it as one package.

To install Docker, follow the given steps:

1. Type the following commands on the terminal

```

> sudo apt-get update
> sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

```

```

> sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
> sudo apt-get update
> sudo apt-get install docker-ce docker-ce-cli containerd.io

```

2. Check whether Docker has been installed or not

```
> sudo docker run hello-world
```

3. Configure Docker to run without sudo and also give Jenkins permission to run docker commands without sudo

```

> sudo groupadd docker
> sudo usermod -aG docker mdk1997
> sudo usermod -aG docker jenkins

```

3.4.2 Building and Publishing Images

Maven build outputs a jar file consisting of the compiled classes, their dependencies and also the embedded Tomcat server. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

So, I first created a Dockerfile. The Dockerfile is placed in the root directory of the project. It is shown below.

```

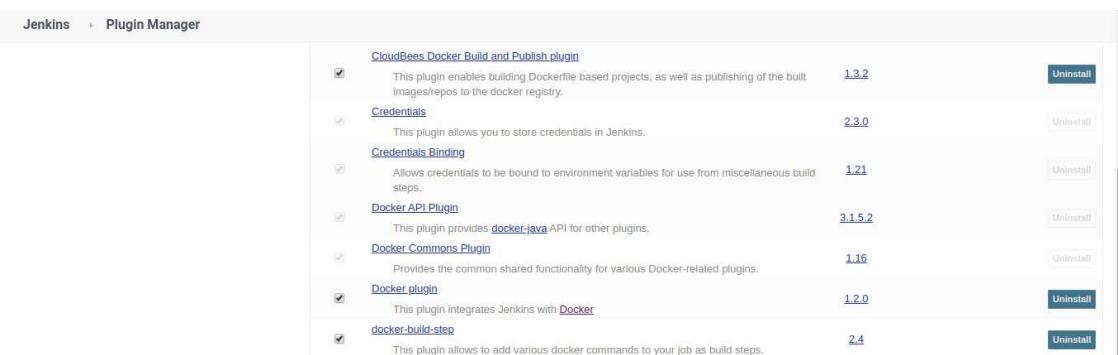
FROM openjdk:8
ADD target/devops-on-calculator-1.11-SNAPSHOT.jar devops-on-calculator-1.11-SNAPSHOT.jar
RUN mkdir -p -m 777 /var/log/tomcat
EXPOSE 8088
ENTRYPOINT ["java", "-jar", "devops-on-calculator-1.11-SNAPSHOT.jar"]

```

3.4.3 Jenkins Pipeline Job 2: calculator-build-publish-image

I now use this Jenkins project to build the Docker image and publish it to Dockerhub. To do this, follow the given steps:

1. On the Jenkins main dashboard, go to Manage Jenkins -> Manage Plugins and Docker plugins.



2. Create a new freestyle project and do the following configuration.

a. General

The screenshot shows the 'General' configuration tab for a Jenkins project. The 'Description' field contains the text: '- Run after calculator-maven-build-pipeline
- Build docker image
- Publish image'. Below this, there is a list of checkboxes for various project settings:

- Commit agent's Docker container
- Define a Docker template
- Discard old builds
- GitHub project
- This project is parameterised
- Throttle builds
- Disable this project
- Execute concurrent builds if necessary

A small 'Advanced...' button is located at the bottom right of the configuration area.

b. Source Code Management

The screenshot shows the 'Source Code Management' configuration tab. It is set to 'Git' as the source code provider. Under 'Repositories', a single repository is configured with the URL <https://github.com/mkd1997/devops-on-calculator.git> and credentials 'mkd1997*****'. The 'Branches to build' section shows a branch specifier of '*/*master'. The 'Repository browser' is set to '(Auto)'. At the bottom, there are 'Save' and 'Apply' buttons.

c. Build Trigger

The screenshot shows the 'Build Triggers' configuration tab. It includes the following triggers:

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built

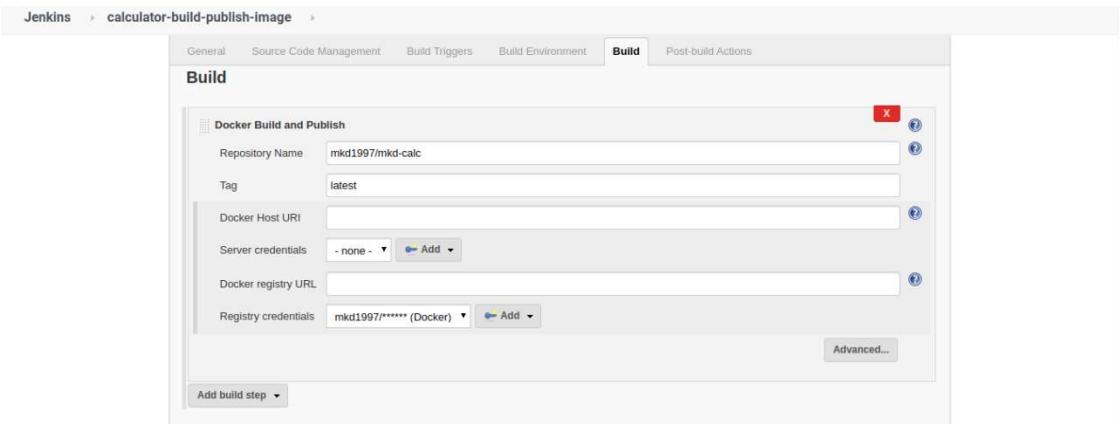
For the trigger 'Build after other projects are built', a 'Projects to watch' dropdown is set to 'calculator-maven-build-pipeline'. There are three options for triggering:

- Trigger only if build is stable
- Trigger even if the build is unstable
- Trigger even if the build fails

Below these, there are several other trigger options that are currently unchecked:

- Build periodically
- Build when we receive a notification from Rundeck
- GitHub hook trigger for GITScm polling
- Poll SCM

d. Build



Console output of the calculator-build-publish-image is shown below

```
Started by upstream project "calculator-maven-build-pipeline" build number 20
originally caused by:
  Started by user Manav Desai
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/calculator-build-publish-image
using credential 70b687bb-5e2f-46d5-b522-69cf9bb4fe30
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/mkd1997/devops-on-calculator.git # timeout=10
Fetching upstream changes from https://github.com/mkd1997/devops-on-calculator.git
> git --version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --tags --progress -- https://github.com/mkd1997/devops-on-calculator.git +refs/heads/*:refs/remotes/origin/*
# timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 279b69694ef14d672a117de7729eb3059a04308b (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 279b69694ef14d672a117de7729eb3059a04308b # timeout=10
Commit message: "dockerfile changed"
> git rev-list --no-walk 279b69694ef14d672a117de7729eb3059a04308b # timeout=10
[calculator-build-publish-image] docker build -t mkd1997/mkd-calc:latest --pull=true
/var/lib/jenkins/workspace/calculator-build-publish-image
Sending build context to Docker daemon 33.68MB
```

```

Activities Google Chrome ▾ Sun May 10, 11:38 AM
Word x | w Calculator Report.docx x | Microsoft Word - MT201914 x calculator-build-publish-im... x + 
localhost:8080/job/calculator-build-publish-image/lastBuild/console
Apps Machine Learning Python To Read To Watch Study To Do Java To Discuss Utility Minimum Point...
Jenkins > calculator-build-publish-image > #2 mkd1997/mkd-calc:latest mkd1997/mkd-calc:latest
$/0u/cb40/bc: Runned
latest: digest: sha256:5ab593258508c8a0b9614a70e8cb72ab05fd919f35cbe8402a86fa99abca5317 size: 2213
[calculator-build-publish-image] $ docker push mkd1997/mkd-calc:latest
The push refers to repository [docker.io/mkd1997/mkd-calc]
37d02c64676c: Preparing
481c76a3f6d8: Preparing
d2f5dc92cc56: Preparing
a109768588b7: Preparing
11533ccb8178f: Preparing
8967306e673e: Preparing
9794a3b3ed45: Preparing
5f77a51ade6a: Preparing
e40d297cf5f8: Preparing
8967306e673e: Waiting
9794a3b3ed45: Waiting
5f77a51ade6a: Waiting
e40d297cf5f8: Waiting
d2f5dc92cc56: Layer already exists
a109768588b7: Layer already exists
11533ccb8178f: Layer already exists
37d02c64676c: Layer already exists
481c76a3f6d8: Layer already exists
8967306e673e: Layer already exists
5f77a51ade6a: Layer already exists
9794a3b3ed45: Layer already exists
e40d297cf5f8: Layer already exists
latest: digest: sha256:5ab593258508c8a0b9614a70e8cb72ab05fd919f35cbe8402a86fa99abca5317 size: 2213
Triggering a new build of calculator-deploy
Finished: SUCCESS

```

Page generated: 10-May-2020 11:37:54 IST REST API Jenkins ver. 2.222.1

3.5 Continuous Deployment

Continuous deployment is a strategy for software releases wherein any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

After the deliverable (image in my case) is created and published to dockerhub, I use Rundeck to fetch the image and deploy it in a container. The Rundeck job is invoked by Jenkins.

3.5.1 Rundeck Installation

Rundeck is an automation tool that executes Rundeck jobs on Rundeck nodes. Rundeck jobs can be thought of a sequence of instructions and Rundeck nodes could be anything like a web server, container, etc.

To install Rundeck, follow the given steps:

1. Rundeck requires that you have Java 8 on your system. So, first check the Java version on your system.

```
> java -version
```

2. If you already have Java 8 installed, skip this step. Else, install Java 8 and configure your system to use Java 8 by default.

```
> sudo apt-get update
> sudo apt-get install openjdk-8-jdk
> sudo update-alternatives --config java      # then select Java 8
```

3. Download deb package from <http://rundeck.org/download/deb/> and run the command

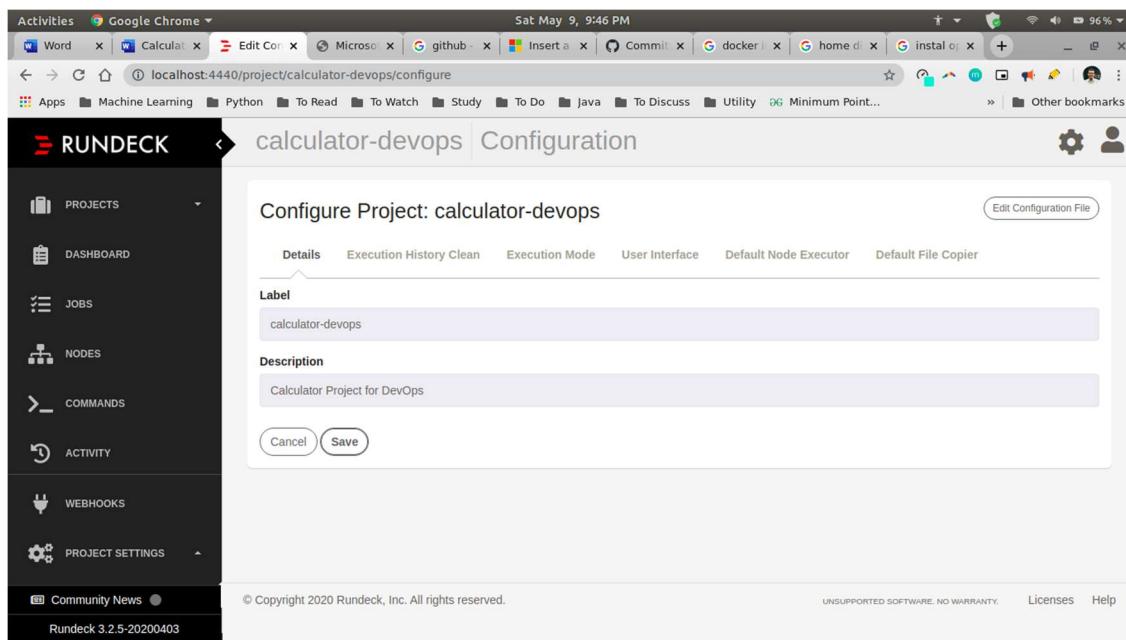
```
> sudo dpkg -i rundeck_3.2.5.20200403-1_all.deb
```
4. Start Rundeck

```
> sudo service rundeckd start
```
5. Rundeck runs at <http://localhost:4440> by default and default username and password are admin and admin respectively.

3.5.2 Creating Rundeck Project and Job

I now create a Rundeck project and Rundeck job that deploys the generated Docker image to a new container. To do so, follow the given steps:

1. Create a new Rundeck project.



The screenshot shows a Google Chrome browser window with the URL localhost:4440/project/calculator-devops/configure. The page title is "calculator-devops Configuration". The left sidebar has a dark theme with icons for PROJECTS, DASHBOARD, JOBS, NODES, COMMANDS, ACTIVITY, WEBHOOKS, and PROJECT SETTINGS. The main content area is titled "Configure Project: calculator-devops". It contains tabs for Details, Execution History Clean, Execution Mode, User Interface, Default Node Executor, and Default File Copier. Under "Label", the value "calculator-devops" is shown. Under "Description", the value "Calculator Project for DevOps" is shown. At the bottom are "Cancel" and "Save" buttons. The footer includes copyright information and links for Community News, Licenses, and Help.

2. Create a new Rundeck job in the project and do the following configurations:

a. Details

calculator-devops

Edit Job: DeployImageToTomcatInContainer 7ba860fc-c308-444f-bce1-cf90883aa8f8

Job Name: DeployImageToTomcatInContainer Category: deployment

Description:

```
1. This job does the following;
2. Remove existing container and image
3. Pull the latest image
4. Run it in tomcat container with logging enabled
```

The first line of the description will be shown in plain text, the rest will be rendered with Markdown.

Cancel Save

b. Workflow

Stop at the failed step. Run remaining steps before failing.

Strategy: Node First

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

Global Log Filters + add

1. echo test

2. docker rm -f calculator-container

3. docker rmi -f mdk1997/mkd-calc:latest

4. docker pull mdk1997/mkd-calc:latest

5. docker run -d -p 8181:8088 -v /var/log/tomcat:/var/log/tomcat --name calculator-container mdk1997/mkd-calc:latest

+ Add a step

c. Nodes

The screenshot shows the Rundeck interface with the 'Nodes' tab selected. The job name is 'DeployImageToTomcatInContainer'. The 'Nodes' tab has two options: 'Dispatch to Nodes' (radio button) and 'Execute locally' (radio button, selected). A note below says 'Choose whether the Job will run on filtered nodes or only on the local node.' Below that, 'Show Excluded Nodes' is set to 'No'. A note says 'If true, the excluded nodes will be indicated when running the Job. Otherwise they will not be shown at all.' At the bottom are 'Cancel' and 'Save' buttons.

3. Note the UUID of the job in the Other tab.

The screenshot shows the Rundeck interface with the 'Other' tab selected. The job UUID is listed as '7ba860fc-c308-444f-bce1-cf90883aa8f8'. The 'Other' tab contains several configuration sections: 'Retry' (maximum number of times to retry execution), 'Retry Delay' (time between failed execution and retry), 'Log Output Limit' (maximum line count per node or file size), 'Log Limit Action' (options include 'Halt with status:' or 'Truncate and continue'), and 'Default Tab' (set to 'Nodes'). At the bottom are 'Cancel' and 'Save' buttons.

3.5.3 Jenkins Pipeline Job 3: calculator-deploy-image

I use this Jenkins job to invoke the Rundeck job that deploys the image to a container.

To do this, follow the given steps:

1. On the Jenkins main dashboard, go to Manage Jenkins -> Manage Plugins and install Rundeck plugin.

The screenshot shows the Jenkins Plugin Manager interface. The 'Installed' tab is selected. A search bar at the top right contains the text 'rundeck'. A single plugin entry is visible: 'Rundeck plugin' by Rundeck. The entry includes a brief description: 'This plugin is a Notifier (Publisher) that will talk to a Rundeck instance (via its HTTP API) to schedule a job execution on Rundeck after a successful build on Jenkins. It is also a Jenkins Trigger, that will schedule a build on Jenkins after a job execution on Rundeck (using Rundeck WebHook Notification). In addition, it turns Jenkins into an Option provider for Rundeck, if you want to use your Jenkins build artifacts as an option to a Rundeck job.' Below the description are buttons for 'Version' (3.6.7), 'Previously installed version', and 'Uninstall'.

2. Create a new freestyle project and do the following configuration.

- a. General

The screenshot shows the 'General' configuration page for a Jenkins project named 'calculator-deploy'. The 'Build Triggers' section is expanded, showing the following configuration:

- Run after calculator-build-publish-image
- Deploy to container using Rundeck

Below this, under 'Post-build Actions', several options are listed:

- Commit agent's Docker container
- Define a Docker template
- Discard old builds
- Github project
- This project is parameterised
- Throttle builds
- Disable this project
- Execute concurrent builds if necessary

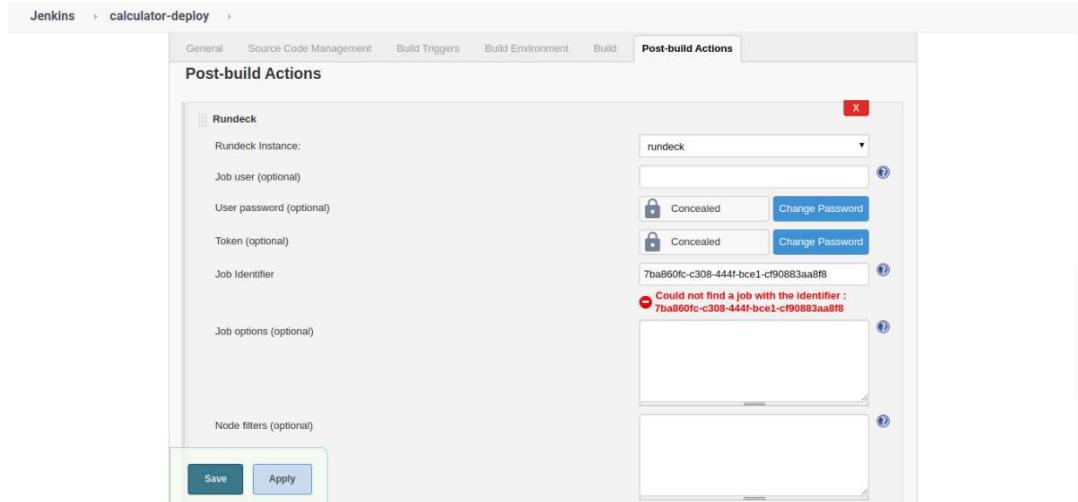
A 'Plain text' link is available to preview the build configuration.

- b. Build Trigger

The screenshot shows the 'Build Triggers' configuration page for the 'calculator-deploy' project. The 'Build Triggers' tab is selected. Under 'Build after other projects are built', the 'calculator-build-publish-image' project is selected. The 'Trigger only if build is stable' radio button is selected. Other trigger options listed include:

- Trigger builds remotely (e.g., from scripts)
- Build periodically
- Build when we receive a notification from Rundeck
- GitHub hook trigger for GITScm polling
- Poll SCM

c. Post-Build Actions



3. Click Save.

The output of the Jenkins job can be seen in Jenkins as well as Rundeck as shown below.

```

Started by upstream project "calculator-build-publish-image" build number 2
originally caused by:
Started by upstream project "calculator-maven-build-pipeline" build number 20
originally caused by:
    Started by user Manav Desai
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/calculator-deploy
Instance 'rundeck' with rundeck user 'admin': Notifying Rundeck...
Looking for jobId : 7ba860fc-c308-444f-bce1-cf90883aa8f8
Notification succeeded ! Execution #56, at http://localhost:4440/api/35/execution/56 (status : RUNNING)
Waiting for Rundeck execution to finish...
BEGIN RUNDECK TAILED LOG OUTPUT
[12:49:17] [NORMAL] test
[12:49:24] [NORMAL] calculator-container
[12:49:26] [NORMAL] Untagged: mkd1997/mkd-calc@sha256:5ab593258508c8a0b9614a70e8cb72ab05fd919f35cbe8402a86fa99abca5317
[12:49:26] [NORMAL] Untagged: sha256:1c81dd6feb75cc33b43cafbaf317e7ec31cb0526bb71b1032733ee97893204ef
[12:49:26] [NORMAL] Deleted: sha256:f847fb258afe59375e6d10a0ae42e4e041ebd9c0380018c5fd2706d665dd2c
[12:49:26] [NORMAL] Deleted: sha256:d4c5b65ae867d42131fdef8230dc03t5264fe71c31fadbf49981d45709aebb4
[12:49:26] [NORMAL] Deleted: sha256:95c219208cce34e6a1cd6eff7213e05fc6a1362acece3a88dbc2520635af736
[12:49:26] [NORMAL] Deleted: sha256:92a43de1ef3187b9308e759bfd760578f3f4633970e8c625e85fed52b482c93c
[12:49:29] [NORMAL] latest: Pulling from mkd1997/mkd-calc
[12:49:29] [NORMAL] 90fe46dd8199: Already exists
[12:49:29] [NORMAL] 35a4f1977689: Already exists
[12:49:29] [NORMAL] 1_hhr27f13ad0d: Already exists

```

Activities Google Chrome ▾ Sun May 10, 11:59 AM

Word Calculator Report.docx Microsoft Word - MT201914 calculator-deploy #2 Cons...

localhost:8080/job/calculator-deploy/lastBuild/console

Apps Machine Learning Python To Read To Watch Study To Do Java To Discuss Utility Minimum Point... Other bookmarks

Jenkins > calculator-deploy > #2

```
BEGIN RUNDECK [ALLED LOG UUIMUI
[12:49:17] [NORMAL] test
[12:49:24] [NORMAL] calculator-container
[12:49:26] [NORMAL] Untagged: mkd1997/mkd-calc:latest
[12:49:26] [NORMAL] Untagged: mkd1997/mkd-calc@sha256:5ab593258508c8a0b9614a70e8cb72ab05fd919f35cbe8402a86fa99abca5317
[12:49:26] [NORMAL] Deleted: sha256:1c81dd6feb75cc33b43cafba317e7ec31cb0526bb71b1032733ee97893204ef
[12:49:26] [NORMAL] Deleted: sha256:f847fb2c58afe59375e6d10a03ae42e4e041eb9dc0380018c5fda2706d056dd2c
[12:49:26] [NORMAL] Deleted: sha256:4c5b65ae867d42131fde0f8230dc03f5264fe71c31fadb49981d45709aebbf4
[12:49:26] [NORMAL] Deleted: sha256:95c219208ccc34e6a1cd6eff7213e05fc6a1362acece3a88dbc2520635af736
[12:49:26] [NORMAL] Deleted: sha256:92a43dlef3187b9308e759bfd760578f3f4633970e8c625e85fed52b482c93c
[12:49:29] [NORMAL] latest: Pulling from mkd1997/mkd-calc
[12:49:29] [NORMAL] 90fe466dd8199: Already exists
[12:49:29] [NORMAL] 35a4f1977689: Already exists
[12:49:30] [NORMAL] b8c37f14aded: Already exists
[12:49:30] [NORMAL] 74e27dc593d4: Already exists
[12:49:30] [NORMAL] 93a01fbfad7f: Already exists
[12:49:30] [NORMAL] 1478df405869: Already exists
[12:49:30] [NORMAL] 64f0d11682b: Already exists
[12:49:31] [NORMAL] b9ca842da066: Already exists
[12:49:31] [NORMAL] d33abdcdeb124: Pulling fs layer
[12:49:32] [NORMAL] d33abdcdeb124: Download complete
[12:49:36] [NORMAL] d33abdcdeb124: Pull complete
[12:49:36] [NORMAL] Digest: sha256:5ab593258508c8a0b9614a70e8cb72ab05fd919f35cbe8402a86fa99abca5317
[12:49:36] [NORMAL] Status: Downloaded newer image for mkd1997/mkd-calc:latest
[12:49:36] [NORMAL] docker.io/mkd1997/mkd-calc:latest
[12:49:38] [NORMAL] 9fb6274541928ccc8668ea36235e22d3a44f1b791ae52a0bc7a36c2003acf0bd
END RUNDECK TAILED LOG OUTPUT
Rundeck execution #56 finished in 28 seconds, with status : SUCCEEDED
Finished: SUCCESS
```

Page generated: 10-May-2020 11:58:58 IST REST API Jenkins ver. 2.222.1

Activities Google Chrome ▾ Sun May 10, 12:02 PM

Word Calculator Report.docx Microsoft Word - MT201914 [OK] Rundeck - DeployIn...

localhost:4440/project/calculator-devops/execution/show/56

Apps Machine Learning Python To Read To Watch Study To Do Java To Discuss Utility Minimum Point... Other bookmarks

RUNDECK

PROJECTS DASHBOARD JOBS NODES COMMANDS ACTIVITY WEBHOOKS PROJECT SETTINGS

PULL THE LATEST IMAGE
RUN IT IN TOMCAT IN CONTAINER WITH LOGGING ENABLED

Log Output »

100% 1/1 COMPLETE		0 FAILED	0 INCOMPLETE	0 NOT STARTED
Node	localhost	All Steps OK		
	> Command	OK	12:49:17 pm	0:00:00
	> Command	OK	12:49:17 pm	0:00:07
	> Command	OK	12:49:25 pm	0:00:01
	> Command	OK	12:49:26 pm	0:00:10
	> Command	OK	12:49:37 pm	0:00:04

Stats Activity

13 EXECUTIONS 69% SUCCESS RATE 1m20s AVG DURATION

© Copyright 2020 Rundeck, Inc. All rights reserved. UNSUPPORTED SOFTWARE. NO WARRANTY. Licenses Help

Rundeck 3.2.5-20200403

3.6 Monitoring

Monitoring refers to monitoring the deployed artifacts in real-time to keep a check on faults and measure performance. This is done by analyzing logs generated by the system. The Elastic Stack, also known as BELK Stack, comprises of Elasticsearch, Logstash, Kibana and Beats.

Elasticsearch is a modern search and analytics engine which is based on Apache Lucene. It is used as to store and index logs and can be then queried to extract meaningful insights. It can be used for numerous types of data including textual, numerical, geospatial, structured, and unstructured.

Logstash is a tool that is used for parsing logs. It is very useful in parsing unstructured logs and giving them structure so that logs can be efficiently searched and analyzed. Log aggregated and processed by Logstash go through 3 stages – collection, processing and dispatching.

Kibana adds a visualization layer to the Elastic Stack. It is a browser-based user interface that can be used to search, analyze and visualize the data stored in Elasticsearch indices.

Beats are a collection of open source log shippers that act as agents installed on the different servers in your environment for collecting logs or metrics. These shippers are designed to be lightweight in nature — they leave a small installation footprint, are resource efficient, and function with no dependencies. Common Beats that are used today included Filebeat, Metricbeat, Winlogbeat, Packetbeat, etc.

I use the Elastic Stack to perform log analysis in the Calculator project. I use Filebeat to collect logs generated by Tomcat which are then given to Elasticsearch for analysis. Kibana is used to create a pie chart that shows the ratio of requests that were successful.

3.6.1 Installing the Elastic Stack

To install the Elastic Stack, follow the given steps:

1. First install Elasticsearch using the following commands.

```
> wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
> sudo apt-get update> sudo apt-get install apt-transport-https
> echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a
/etc/apt/sources.list.d/elastic-7.x.list
> sudo apt-get install elasticsearch
```

2. Elasticsearch runs at <http://localhost:9200>. To start Elasticsearch

```
> sudo service elasticsearch start
```

3. Now, install Kibana using the following commands.

```
> sudo apt-get install kibana
```

4. Make sure you have the following configuration in /etc/kibana/kibana.yml.

```
server.port: 5601
elasticsearch.url: http://localhost:9200
```

5. Kibana runs at <http://localhost:5601>. To start Elasticsearch

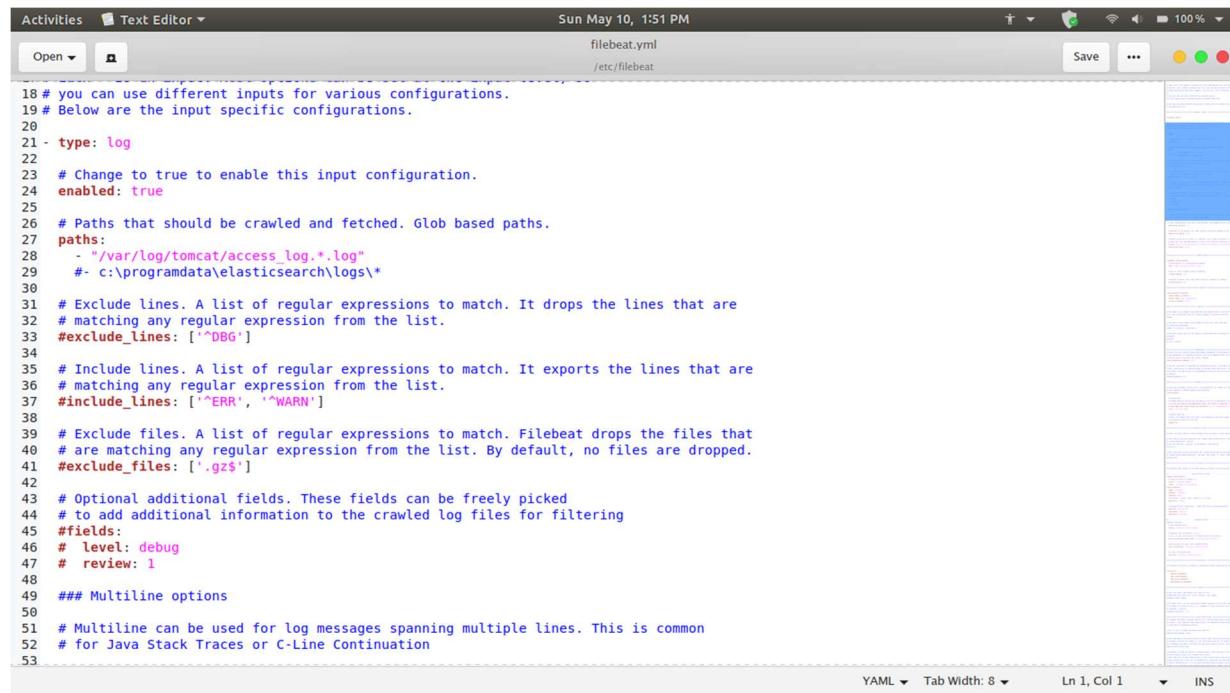
```
> sudo service elasticsearch start
```

6. Now, install filebeat using the following commands.

```
> sudo apt-get install filebeat
```

3.6.2 Configuring Filebeat

I have configured Filebeat to directly send the Tomcat access logs to Elasticsearch for processing. The importance of deploying the image in a container with /var/log/tomcat folder in container mapped to the same folder on host system can now be seen. The Elastic Stack runs outside the container. Since, Filebeat cannot access the folders in the container (where the logs are generated), it accesses the folder on the host system. The path of configuration file for any Beats in Linux is /etc/filebeat/filebeat.yml. To configue Filebeat, edit the yml file as shown in the screenshots below.



The screenshot shows a Mac OS X Text Editor window titled "filebeat.yml" located in the "/etc/filebeat" directory. The file contains a YAML configuration for Filebeat. The code is as follows:

```
18 # you can use different inputs for various configurations.
19 # Below are the input specific configurations.
20
21 - type: log
22
23 # Change to true to enable this input configuration.
24 enabled: true
25
26 # Paths that should be crawled and fetched. Glob based paths.
27 paths:
28   - "/var/log/tomcat/access.log.*.log"
29   #- c:\programdata\elasticsearch\logs\*
30
31 # Exclude lines. A list of regular expressions to match. It drops the lines that are
32 # matching any regular expression from the list.
33 #exclude_lines: [ '^DBG' ]
34
35 # Include lines. A list of regular expressions to match. It exports the lines that are
36 # matching any regular expression from the list.
37 #include_lines: [ '^ERR', '^WARN' ]
38
39 # Exclude files. A list of regular expressions to match. Filebeat drops the files that
40 # are matching any regular expression from the list. By default, no files are dropped.
41 #exclude_files: [ '.gz$' ]
42
43 # Optional additional fields. These fields can be freely picked
44 # to add additional information to the crawled log files for filtering
45 #fields:
46 #  level: debug
47 #  review: 1
48
49 ### Multiline options
50
51 # Multiline can be used for log messages spanning multiple lines. This is common
52 # for Java Stack Traces or C-Line Continuation
53
```

Activities Text Editor ▾ Sun May 10, 1:52 PM filebeat.yml /etc/filebeat

```
51 # Multiline can be used for log messages spanning multiple lines. This is common
52 # for Java Stack Traces or C-Line Continuation
53
54 # The regexp Pattern that has to be matched. The example pattern matches all lines starting with [
55 #multiline.pattern: ^\[
56
57 # Defines if the pattern set under pattern should be negated or not. Default is false.
58 #multiline.negate: false
59
60 # Match can be set to "after" or "before". It is used to define if lines should be append to a pattern
61 # that was (not) matched before or after or as long as a pattern is not matched based on negate.
62 # Note: After is the equivalent to previous and before is the equivalent to to next in Logstash
63 #multiline.match: after
64
65
66 ===== Filebeat modules =====
67
68 filebeat.config.modules:
69 # Glob pattern for configuration loading
70 path: ${path.config}/modules.d/*.yml
71
72 # Set to true to enable config reloading
73 reload.enabled: true
74
75 # Period on which files under path should be checked for changes
76 #reload.period: 10s
77
78 ===== Elasticsearch template setting =====
79
80 setup.template.settings:
81 index.number_of_shards: 1
82 #index.codec: best_compression
83 #_source.enabled: false
84
85 ===== General =====
86
```

Save ...

YAML ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

Activities Text Editor ▾ Sun May 10, 1:52 PM filebeat.yml /etc/filebeat

```
100 ===== Dashboards =====
101 # These settings control loading the sample dashboards to the Kibana index. Loading
102 # the dashboards is disabled by default and can be enabled either by setting the
103 # options here or by using the `setup` command.
104 #setup.dashboards.enabled: true
105
106 # The URL from where to download the dashboards archive. By default this URL
107 # has a value which is computed based on the Beat name and version. For released
108 # versions, this URL points to the dashboard archive on the artifacts.elastic.co
109 # website.
110 #setup.dashboards.url:
111
112 ===== Kibana =====
113 # Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
114 # This requires a Kibana endpoint configuration.
115 #setup.kibana:
116
117 # Kibana Host
118 # Scheme and port can be left out and will be set to the default (http and 5601)
119 # In case you specify an additional path, the scheme is required: http://localhost:5601/path
120 # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
121 host: "localhost:5601"
122
123 # Kibana Space ID
124 # ID of the Kibana Space into which the dashboards should be loaded. By default,
125 # the Default Space will be used.
126 #space.id:
127
128 ===== Elastic Cloud =====
129
130 # These settings simplify using Filebeat with the Elastic Cloud (https://cloud.elastic.co/).
131
132 # The cloud.id setting overwrites the `output.elasticsearch.hosts` and
133 # `setup.kibana.host` options.
134 # You can find the `cloud.id` in the Elastic Cloud web UI.
135
136 #
```

Save ...

YAML ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

```
Activities Text Editor ▾ Sun May 10, 1:52 PM
Open ⌂ Save ...
filebeat.yml
/etc/filebeat
143 #===== Outputs =====
144
145 # Configure what output to use when sending the data collected by the beat.
146
147 #----- Elasticsearch output -----
148 output.elasticsearch:
149   # Array of hosts to connect to.
150   hosts: ["localhost:9200"]
151   index: "filebeat-%{+yyyy.MM.dd}"
152 setup.template:
153   name: 'filebeat'
154   pattern: 'filebeat-*'
155   enabled: false
156   # Protocol - either `http` (default) or `https`.
157   #protocol: "https"
158
159   # Authentication credentials - either API key or username/password.
160   #api_key: "id:api key"
161   #username: "elastic"
162   #password: "changeme"
163
164 #----- Logstash output -----
165 #output.logstash:
166   # The Logstash hosts
167   #hosts: ["http://localhost:5044"]
168
169   # Optional SSL. By default is off.
170   # List of root certificates for HTTPS server verifications
171   #ssl.certificateAuthorities: ["/etc/pki/root/ca.pem"]
172
173   # Certificate for SSL client authentication
174   #ssl.certificate: "/etc/pki/client/cert.pem"
175
176   # Client Certificate Key
177   #ssl.key: "/etc/pki/client/cert.key"
178
```

```
Activities Text Editor ▾ Sun May 10, 1:52 PM
Open ⌂ Save ...
filebeat.yml
/etc/filebeat
179 #===== Processors =====
180
181 # Configure processors to enhance or manipulate events generated by the beat.
182
183 processors:
184   - add_host_metadata: ~
185   - add_cloud_metadata: ~
186   - add_docker_metadata: ~
187   - add_kubernetes_metadata: ~
188
```

Restart Filebeat after editing the configuration file.

```
> sudo service filebeat restart
```

3.6.3 Visualizing Logs in Kibana

I now create visualizations to show the number of requests that our service failed to serve. I first create an index pattern which allows Kibana to fetch the logs coming from Filebeat. I then use the HTTP Response code to show the proportion of requests that were served successfully.

To do this, follow the given steps:

1. On the Kibana dashboard, go to Management -> Index Pattern. Create the index pattern filebeat-*

The screenshot shows the Kibana Management interface in Google Chrome. The URL is `localhost:5601/app/kibana#/management/kibana/index_pattern?_g=()`. The left sidebar has sections for Elasticsearch (Index Management, Index Lifecycle Policies, Rollup Jobs, Transformations, Remote Clusters, Snapshot and Restore, License Management, 8.0 Upgrade Assistant) and Kibana (Index Patterns, Saved Objects, Spaces, Reporting, Advanced Settings). The main area is titled "Create index pattern" with the sub-step "Step 1 of 2: Define index pattern". The "Index pattern" field contains "filebeat-*". A note says: "You can use a * as a wildcard in your index pattern. You can't use spaces or the characters \, /, ?, *, <, >,]." Below it, a success message says: "Success! Your index pattern matches 1 index." The index found is "filebeat-7.6.2-2020.05.02-000001". The "Rows per page" dropdown is set to 10. A "Next step" button is visible on the right.

2. On the Kibana dashboard, go to Discover and change the index pattern to filebeat-* from the dropdown list.

The screenshot shows the Kibana Discover interface in Google Chrome. The URL is `localhost:5601/app/kibana#/discover?_g=(&_a=(columns:_source),index:filebeat-*&interval:auto,query:(language:ku...))`. The left sidebar has "Discover" options (New, Save, Open, Share, Inspect) and a "CHANGE INDEX PATTERN" section with a dropdown set to "filebeat-*". The main area shows search results for "filebeat-*" with 3 hits from May 10, 2020, between 12:06:41.408 and 12:21:41.408. The results are grouped by time. A "Refresh" button is at the top right.

3. Add a filter as shown below.

Discover: HTTP Status

New Save Open Share Inspect

KQL Last 15 minutes Show dates Refresh

EDIT FILTER

Field Operator
message is one of

Values
200, 404

You've selected all available options

Cancel Save

3 hits Reset search

May 10, 2020 @ 12:08:38.083 - May 10, 2020 @ 12:23:38.083 — Auto

2:11:00 12:12:00 12:13:00 12:14:00 12:15:00 12:16:00 12:17:00 12:18:00 12:19:00 12:20:00 12:21:00 12:22:00 12:23:00

message

> May 10, 2020 @ 12:18:04.694 0:0:0:0:0:0:1 ~ [10/May/2020:11:06:38 +0530] "GET /calculate/500/600/+ HTTP/1.1" 200 21
> May 10, 2020 @ 12:18:04.694 0:0:0:0:0:0:1 ~ [10/May/2020:11:06:41 +0530] "GET /favicon.ico HTTP/1.1" 404 143
> May 10, 2020 @ 12:18:04.418 127.0.0.1 ~ [10/May/2020:11:04:34 +0530] "GET /calculate/100/200/+ HTTP/1.1" 200 22

- Save the search by first clicking the Save button and then clicking the Save link on top. This search can now be used as a data source for creating charts.
- On the Kibana dashboard, go to Visualise and click on Create Visualisation. Choose Pie Chart there.
- Choose the saved search in the Choose Source dialog box.

Visualize

New Pie / Choose a source

HTT

HTTP Status

HTTP Status

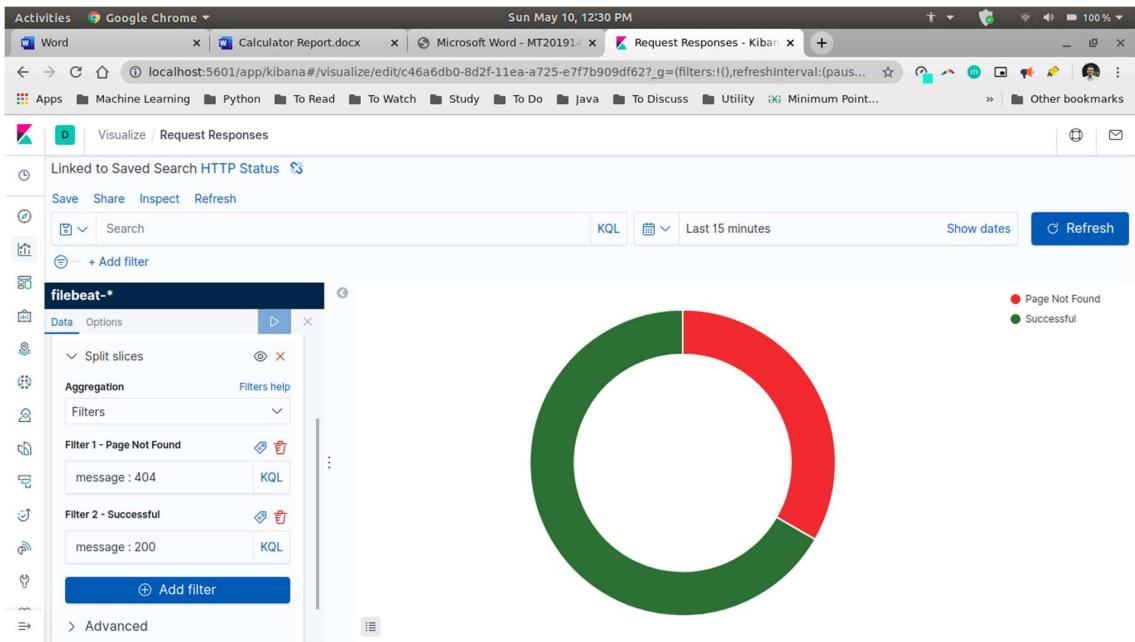
Actions

Visualization

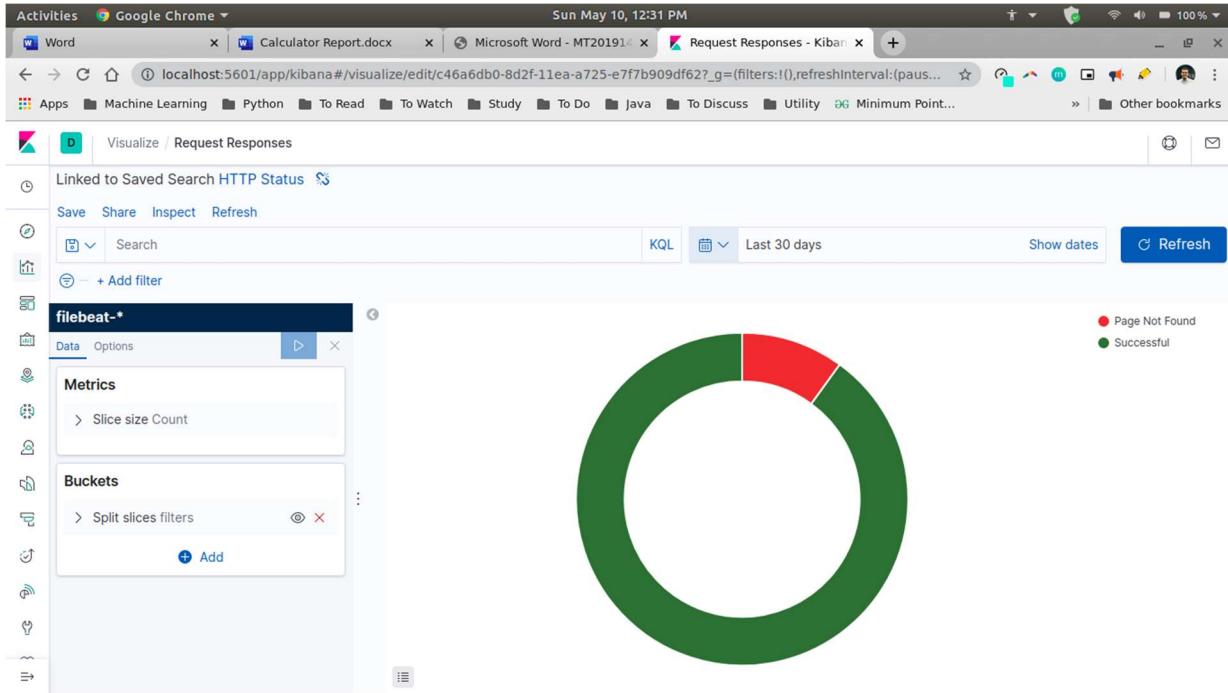
7. Add buckets on the left pane.

The screenshot shows the Kibana visualization editor interface. At the top, it says "filebeat-*". Below that is a navigation bar with "Data" and "Options" tabs, and a "Split slices" section. Under "Aggregation", there is a "Filters" dropdown. Two filters are listed: "Filter 1 - Page Not Found" (KQL: message: 404) and "Filter 2 - Successful" (KQL: message: 200). At the bottom of the editor is a blue button labeled "+ Add filter".

8. The visualization can now be seen. You can also change the colour scheme used and the labels from the legend on the top right corner of the screen.



The final output is shown below. The visualisation will get updated in real-time as the system gets more and more requests.



4. Conclusion

In this project, I automated the entire SDLC using DevOps toolchain. This makes the life of development team and operations team easy as the DevOps pipeline gives the comfort of making code changes easily and also reduces the chances of encountering errors in production. The toolchain allows software companies to quickly build, test and deploy new versions of their products. This can be very useful to companies like Amazon which release new versions many times a day.

5. References

I have referred to the following resources while developing this project:

1. <https://git-scm.com/doc>
2. <https://maven.apache.org/guides/index.html>
3. <https://junit.org/junit5/docs/current/user-guide/>
4. <https://www.jenkins.io/doc/>
5. <https://docs.docker.com/>
6. <https://logz.io/learn/complete-guide-elk-stack/>
7. <https://logz.io/blog/filebeat-tutorial/>
8. <https://logz.io/blog/apache-tomcat-monitoring/>