

Scientific Calculator

With DevOps

Version: V0.0.1

By: Akshath Kaushal

akshath.kaushal@iiitb.ac.in

16th April, 2022



Table of contents

Introduction

Problem statement

DevOps tool chain

Requirements

Key benefits of DevOps

OpenJDK (Java)

Git and GitHub

Testing and Building

Junit

Maven

Continuous Integration

Jenkins

Containerization

Docker

Docker image

Docker file

Docker hub

Deployment

Ansible

Ansible playbooks

Monitoring using ELK Stack

ElasticSearch

Kibana

Logstash

Complete Pipeline of Jenkins

Links and References

Revision History

| Name | Date | Reason | Version |
|-------------|-------------|---------------|----------------|
| Calculator | 14-04-2022 | First Version | Version 0.0.1 |

1. Introduction

Problem statement

Create a scientific calculator program with user menu driven operations

- Square root function - \sqrt{x}
- Factorial function - $x!$
- Natural logarithm (base e) - $\ln(x)$
- Power function - x^b

DevOps tool chain

You can use any set of DevOps tool chains you want, but the pipeline would be the same.

The pipeline includes

1. Using a source control management tool - like GitHub, GitLab, BitBucket etc
2. Testing - test your code using either JUnit, Selenium, PyUnit and many more
3. Build - build your code using tool like Maven, Gradle, Ant and many more
4. Continuous Integration - Continuous integrate your code using tool like Jenkins, GitLab CLI, Travis CLI, and many more
5. Containerize - Containerize your code using Docker.
6. Push your created Docker image to Docker hub.
7. Deployment - Do configuration management and deployment using either Chef, Puppet, Ansible, Rundeck. Using these do configuration management and pull your docker image and run it on the managed hosts.
8. For Deployment you can either do it on your local machine or on Kubernetes cluster or OpenStack cloud. You can also use Amazon AWS or Google Cloud or some other 3rd party cloud.
9. Monitoring - for monitoring use the ELK stack. Use a log file to do the monitoring. Generate the log file for your mini project and pass in your ELK stack.

Requirements

- OpenJDK (Java): building the code
- Git/GitHub: Source control management
- Junit: logging results and errors
- Maven: testing and building
- Jenkins: continuous integration
- Docker: containerization
- Ansible: deployment
- ELK stack: monitoring

2. Key benefits of DevOps

There are numerous benefits of devops when it comes to software production and distribution

- 1) Ensure faster deployment
- 2) Stabilize work environment
- 3) Significant improvement in product quality
- 4) Automation in repetitive tasks leaves more room for innovation
- 5) Promotes agility in your business
- 6) Continuous delivery of software
- 7) Fast and reliable problem-solving techniques
- 8) Transparency leads to high productivity
- 9) Minimal cost of production

3. OpenJDK (Java)

Java is a general purpose programming language that was first released by Sun microsystems in 1995. It follows an object oriented approach and has least implementation dependencies that make it ideal for industrial applications.

Installation:

```
$ sudo apt-get update
```

```
$ sudo apt-get install openjdk-11-jdk
```

Configuring \$JAVA_HOME env variable:

```
$ sudo nano /etc/environment
```


Add the following line to the end of the file


```
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
```

Restart the system to check the installation and verify it using the following command:

```
$ echo $JAVA_HOME
```

4. Git and GitHub

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows 

GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features 

Installation:

```
$ sudo apt update
```

```
$ sudo apt install git
```

Configuration for github

```
$ git config --global user.name "Your name"
```

```
$ git config --global user.email "Your email"
```

Check git if installed correctly

```
$ git --version
```

Git commands

1) Git init

Initializes a git repository

```
$ git init
```

2) Git remote add origin

Binds the remote repository to origin in VCS.

```
$ git remote add origin <remote repository link>
```

3) Git status

Checks the status of the local repository

```
$ git status
```

4) Git add

It adds the file's contents to the index to be tracked.

```
$ git add <file name(s)>
```

5) Git commit

It records the changes that are now monitored by the VCS to the repository.

```
$ git commit -m "Commit message"
```

6) Git push

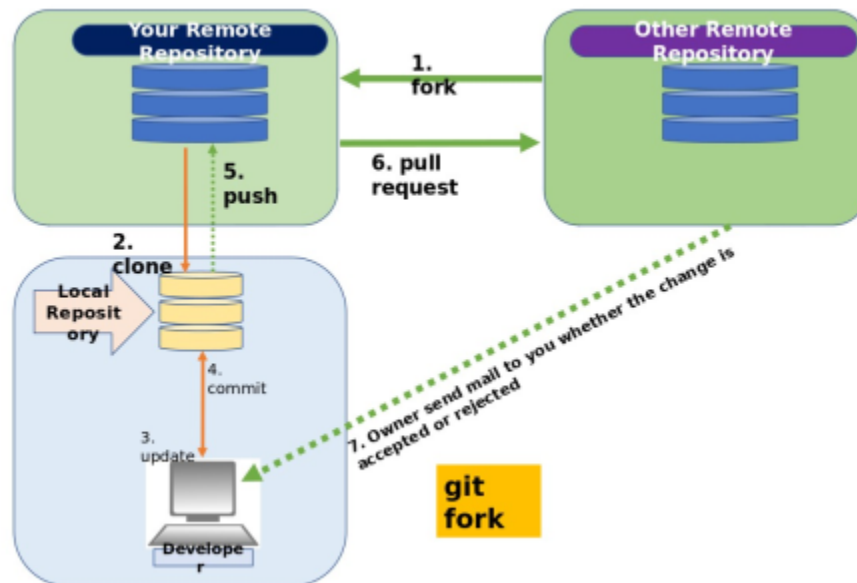
It pushes the changes to the remote repository.

```
$ git push <origin> <branch name>
```

7) Git pull

It pulls the changes to the remote repository and merges it with the local repository if there are no merge conflicts.

```
$ git pull <origin> <branch name>
```



```

akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:
2302 o git init
Initialized empty Git repository in /home/akshathkaushal/Desktop/Work/DevOps/test-repo
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(~|git@master)
2303 ± git remote add origin https://github.com/akshathkaushal/testrepo.git
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(~|git@master)
2304 ± git pull origin master
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 101 (delta 19), reused 95 (delta 13), pack-reused 0
Receiving objects: 100% (101/101), 2.12 MiB | 837.00 KiB/s, done.
Resolving deltas: 100% (19/19), done.
From https://github.com/akshathkaushal/testrepo
 * branch          master      -> FETCH_HEAD
 * [new branch]     master      -> origin/master
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(1d17h53m|git@master)
2305 ± touch testfile.txt
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(1d17h53m|git@master!)
2306 ± git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      testfile.txt

nothing added to commit but untracked files present (use "git add" to track)
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(1d17h54m|git@master!)
2308 ± git add testfile.txt
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(1d17h54m|git@master!)
2309 ± git commit -m "Testing by adding file"
[master eb691a2] Testing by adding file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 testfile.txt
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(0m|git@master)
2310 ± git pull origin master
From https://github.com/akshathkaushal/testrepo
 * branch          master      -> FETCH_HEAD
Already up to date.
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(0m|git@master)
2311 ± git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/akshathkaushal/testrepo.git
  70ee537..eb691a2  master -> master
akshathkaushal-Latitude-5420 3% ~/Desktop/Work/DevOps/test-repo:(0m|git@master)
2312 ± 

```

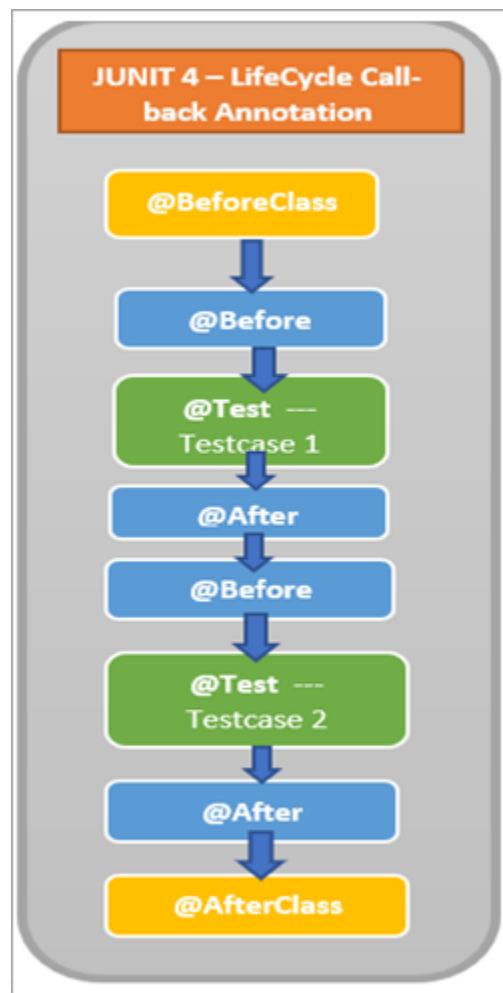

5. Testing and Building

a. JUnit

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

JUnit features include:

- i. Assertions for testing expected results
- ii. Test fixtures for sharing common test data
- iii. Test runners for running tests JUnit.



```
@Test
public void squareRootTest() {
    assertEquals("Correct answer for Square Root Function", 2.0, calculator.squareRoot(4.0), 0.0f);
    assertEquals("Correct answer for Square Root Function", 17.0, calculator.squareRoot(289.0), 0.0f);
}

@Test
public void factorialTest() {
    assertEquals("Correct answer for Factorial Function", 120, calculator.factorial(5));
    assertEquals("Correct answer for Factorial Function", 5040, calculator.factorial(7));
}

@Test
public void naturalLogTest() {
    assertEquals("Correct answer for Natural Log Function", 1.3862943611198906, calculator.naturalLog(4.0), 0.0f);
    assertEquals("Correct answer for Natural Log Function", 1.9459101490553132, calculator.naturalLog(7.0), 0.0f);
}

@Test
public void powerTest() {
    assertEquals("Correct answer for Power Function", 8.0, calculator.power(2.0, 3.0), 0.0f);
    assertEquals("Correct answer for Power Function", 81.0, calculator.power(3.0, 4.0), 0.0f);
}
```

b. Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Installation:

\$ sudo apt install maven

\$ maven --version

Maven directory structure

```
2356 ○ tree spe.miniproject
spe.miniproject
├── ansible.cfg
├── Calculator.log
├── Calculator_logger_logstash.conf
├── deploy-docker
│   ├── deploy-image.yml
│   └── inventory
├── docker-compose.yml
├── Dockerfile
├── Jenkinsfile
├── LICENSE
├── pom.xml
├── README.md
└── src
    ├── main
    │   ├── java
    │   │   └── Calculator.java
    │   └── resources
    │       └── log4j2.xml
    └── test
        ├── java
        │   └── CalculatorTest.java
        └── resources

8 directories, 14 files
```

The main code rests in the src/main/java folder and the test script lies in the src/test/java directory

Configuring pom.xml to include all the dependencies

```

<dependencies>
  <!-- https://mvnrepository.com/artifact/junit/junit -->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.testng/testng -->
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.5</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core -->
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.17.2</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api -->
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.17.2</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>2.0.0-alpha7</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.0.0-alpha7</version>
    <scope>test</scope>
  </dependency>
</dependencies>

```

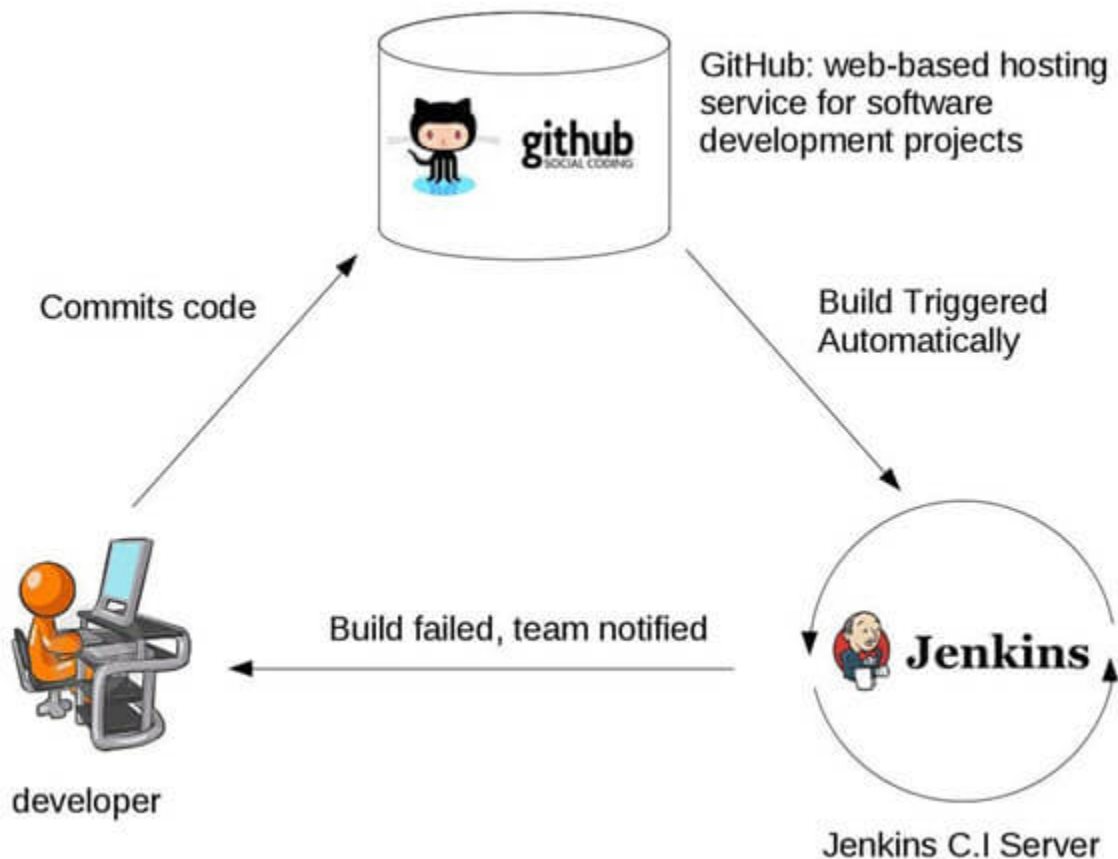
6. Continuous Integration

Continuous Integration (CI) is a DevOps software development practice that enables the developers to merge their code changes in the central repository to run automated builds and tests. It refers to the process of automating the integration of code changes coming from several sources. The process comprises several automation tools that emphasize on the code's correctness before Integration. Continuous Integration is the best practice for software development that has a set of critical principles. Some of the principles of CI are revision control, automated testing, and build automation. The process is not known to get rid of bugs but makes it easy to find and remove bugs. Continuous Integration integrates code into a shared repository frequently. This is done by developers several times a day each time they update the codebase. Each of these integrations can then be tested automatically. One of the main benefits of

integrating regularly and testing each integration is that you can detect errors more quickly and locate them easily. Since each integration or update to the codebase is usually small, pinpointing the exact change that causes the error can be done quickly.

a. Jenkins

Jenkins is an open-source server that is written entirely in Java. It lets you execute a series of actions to achieve the continuous integration process, that too in an automated fashion. This CI server runs in servlet containers such as Apache Tomcat. Jenkins facilitates continuous integration and continuous delivery in software projects by automating parts related to build, test, and deployment. This makes it easy for developers to continuously work on the betterment of the product by integrating changes to the project. Jenkins automates the software builds in a continuous manner and lets the developers know about the errors at an early stage. A strong Jenkins community is one of the prime reasons for its popularity. Jenkins is not only extensible but also has a thriving plugin ecosystem.



Installation:

Add the key

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

Add the repository

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

Start jenkins

```
$ sudo service start jenkins  
$ sudo service status jenkins
```

Jenkins by default runs on port 8080. Navigate to localhost:8080 to use jenkins.

Jenkins's password is stored in the file

```
/var/lib/jenkins/secrets/initialAdminPassword
```

7. Containerization

Containerization is the packaging of software code with just the operating system (OS) libraries and dependencies required to run the code to create a single lightweight executable—called a container—that runs consistently on any infrastructure. More portable and resource-efficient than virtual machines (VMs), containers have become the de facto compute units of modern cloud-native applications.

Containerization allows developers to create and deploy applications faster and more securely. With traditional methods, code is developed in a specific computing environment which, when transferred to a new location, often results in bugs and errors. For example, when a developer transfers code from a desktop computer to a virtual machine (VM) or from a Linux to a Windows operating system. Containerization eliminates this problem by bundling the application code together with the related configuration files, libraries, and dependencies required for it to run. This single package of software or “container” is abstracted away from the host operating system, and hence, it

stands alone and becomes portable—able to run across any platform or cloud, free of issues.

a. Docker

Docker is a set of the platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker, Inc.

Installation:

```
$ sudo apt update
```

Add the docker repository to APT sources

```
$ sudo apt install apt-transport-https ca-certificates curl  
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 bionic stable"
```

Install Docker

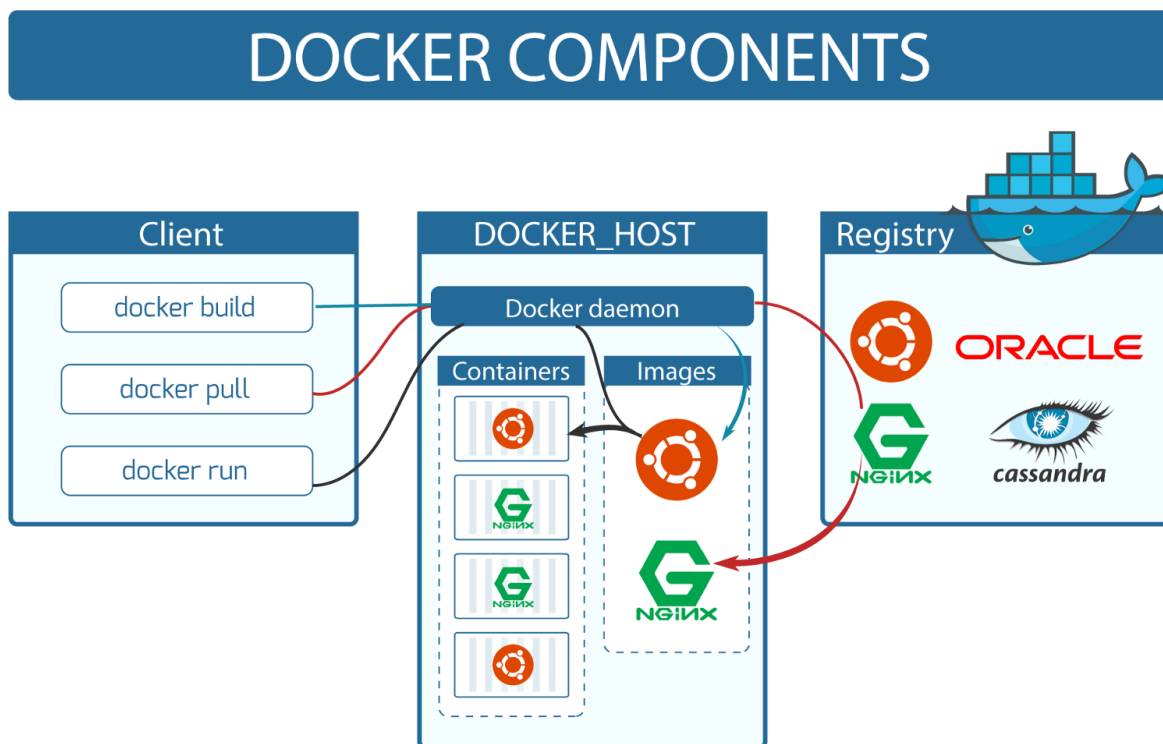
```
$ sudo apt install docker-ce
```

Check docker status

```
$ sudo service docker status
```

Add docker to sudo group

```
$ sudo usermod -aG docker $USER
```



1. Docker image

A Docker image is a file used to execute code in a Docker container. Docker images act as a set of instructions to build a Docker container, like a template. Docker images also act as the starting point when using Docker. An image is comparable to a snapshot in virtual machine (VM) environments.

To be able to run the JAR application in the docker image we need java to be installed.

2. Docker file

A Dockerfile is a text document that contains commands that are used to assemble an image. We can use any command that calls on the command line. Docker builds images automatically by reading the instructions from the Dockerfile. To run the JAR file, we build the container from a JAVA file that is downloaded from Dockerhub. Here is an example of a dockerfile used to run the project:


```
# set the base image
FROM openjdk:11
# file author
MAINTAINER Akshath Kaushal akshath.kaushal@iiitb.ac.in
# Copy the JAR file from local directory
COPY ./target/spe.miniproject-0.0.1-SNAPSHOT.jar
# Set the working directory
WORKDIR ./
# Command to run the JAR file
CMD
["java","-cp","spe.miniproject-0.0.1-SNAPSHOT-jar-with-dependenci
es.jar","Calculator"]
```

```
akshathkaushal-Latitude-5420 ~ /eclipse-workspace/spe.miniproject: (1d2h39m | git@main!)
2364 ± docker build -t akshathkaushal/dimage .
Sending build context to Docker daemon 2.937MB
Step 1/4 : FROM openjdk:11
--> 8c5fc4518cc2
Step 2/4 : COPY ./target/spe.miniproject-0.0.1-SNAPSHOT-jar-with-dependencies.jar ./
--> fa7e5f07d7aa
Step 3/4 : WORKDIR ./
--> Running in 8ff2cd5aebf3
Removing intermediate container 8ff2cd5aebf3
--> 518470f7e346
Step 4/4 : CMD ["java","-cp","spe.miniproject-0.0.1-SNAPSHOT-jar-with-dependencies.jar","Calculator"]
--> Running in 4350d856fcec
Removing intermediate container 4350d856fcec
--> 3fd41a469f76
Successfully built 3fd41a469f76
Successfully tagged akshathkaushal/dimage:latest
```

```
akshathkaushal-Latitude-5420 ~ /eclipse-workspace/spe.miniproject: (1d2h42m | git@main!)
2367 ± docker run --name dimage_container -it akshathkaushal/dimage:latest
Options:
1. Square root
2. Factorial
3. Natual logarithm
4. Raise to power
Press any other key to exit
Enter choice:
2
Option chosen: Factorial
Enter the number:
10
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
3628800
Enter choice:
c
Exiting...
bye bye!
```

3. Docker hub

Docker Hub is a service provided by Docker for finding and sharing container images with your team. It is the world's largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors (ISV) building and distributing their code in containers.

Push the image to Dockerhub

```
akshath@akshath-VirtualBox:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: akshathkaushal7
Password:
WARNING! Your password will be stored unencrypted in /home/akshath/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
akshath@akshath-VirtualBox:~$
```

```
akshathkaushal@Latitude-5420 3% ~/eclipse-workspace/spe.miniproject:(1d2h53m|git@main!)
2377 ± docker push akshathkaushal7/dimage:latest
The push refers to repository [docker.io/akshathkaushal7/dimage]
93b55a22ee0e: Pushed
0816d1f73744: Mounted from akshathkaushal7/spe-miniproject
84f2cb0fc541: Mounted from akshathkaushal7/spe-miniproject
b0dc1a441986: Mounted from akshathkaushal7/spe-miniproject
7a7698da17f2: Mounted from akshathkaushal7/spe-miniproject
d59769727d80: Mounted from akshathkaushal7/spe-miniproject
348622fdcc61: Mounted from akshathkaushal7/spe-miniproject
4ac8bc2cd0be: Mounted from akshathkaushal7/spe-miniproject
latest: digest: sha256:9c86e20b22281b4c07a22c1e132d7a714ce26178cd8011c446c1b7bee7924da9 size: 2006
```

The screenshot displays the Docker Hub interface for the user 'akshathkaushal7'. The top navigation bar includes the Docker Hub logo, a search bar, and links for 'Explore', 'Repositories', 'Organizations', and 'Help'. A yellow 'Upgrade' button and the user's profile icon are also present. The main content area shows a list of repositories under the namespace 'akshathkaushal7'. The repositories listed are 'dimage', 'spe-miniproject', and 'buildhosts'. Each repository entry shows its last push time, a 'Not Scanned' status, star and download counts, and a 'Public' status. A 'Create Repository' button is located at the top right of the repository list. On the right side of the page, there is a prompt to 'Create an Organization' and a banner for 'dockercon 2022' on May 9-10.

8. Deployment

Software deployment is all of the activities that make a software system available for use. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the producer side or at the consumer side or both.

a. Ansible

Ansible is a software tool that provides simple but powerful automation for cross-platform computer support. It is primarily intended for IT professionals, who use it for application deployment, updates on workstations and servers, cloud provisioning, configuration management, intra-service orchestration, and nearly anything a systems administrator does on a weekly or daily basis. Ansible doesn't depend on agent software and has no additional security infrastructure, so it's easy to deploy.

Installation:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository ppa:deadsnakes/ppa
```

Install python3

```
$ sudo apt update
$ sudo apt install python3.8
```

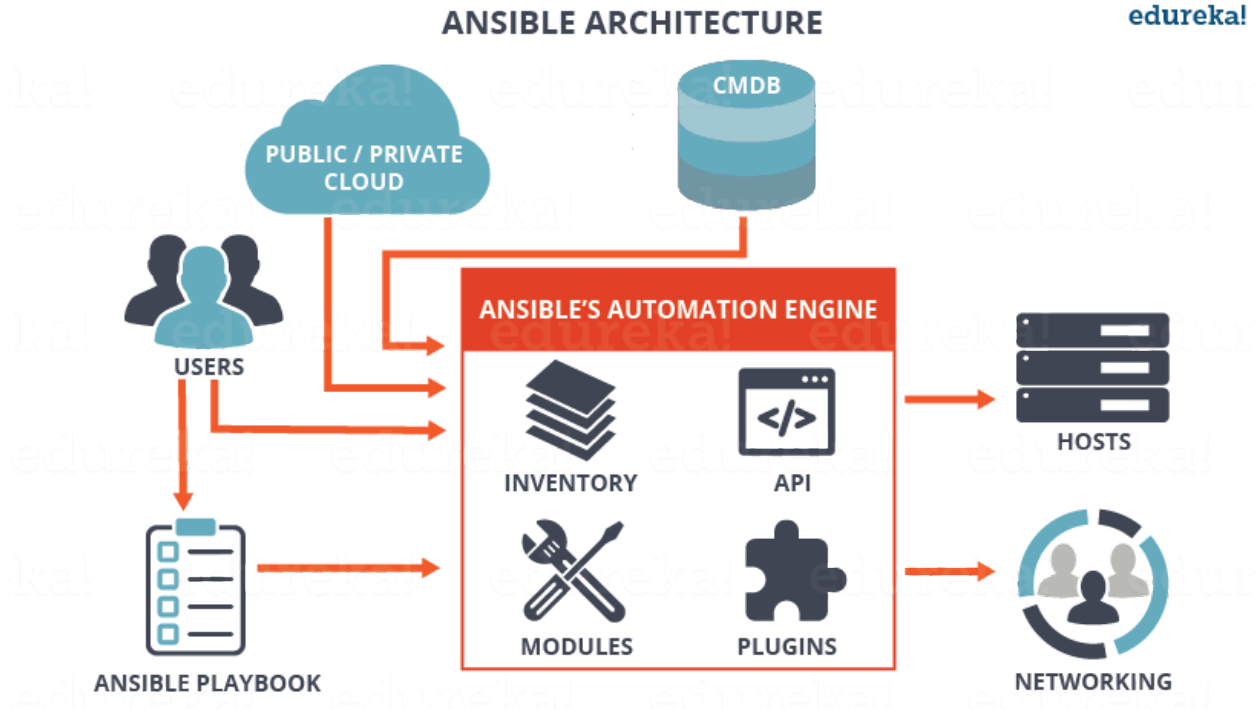
Install Openssh server

```
$ sudo apt update
$ sudo apt install openssh-server
```

```
$ sudo apt update
$ sudo apt install ansible
```

Configure ansible

```
$ sudo apt install sshpass
# edit the sshd_config file to allow root login
$ sudo gedit /etc/ssh/sshd_config
# set PermitRootLogin to yes
$ sudo systemctl restart sshd
```



b. Ansible playbooks

Playbooks are the files where Ansible code is written. Playbooks are written in YAML format. YAML stands for Yet Another Markup Language. Playbooks are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks. Playbooks contain the steps which the user wants to execute on a particular machine. Playbooks are run sequentially. Playbooks are the building blocks for all the use cases of Ansible.

Ansible playbook

```

1 ---
2 - name: Pull docker image of Calculator
3   hosts: all
4   tasks:
5     - name: Pull calculator image
6       docker_image:
7         name: akshathkaushal7/spe-miniproject:latest
8         source: pull

```

Check the syntax of the play book first:

`ansible-playbook --syntax-check <path-to-playbook-yaml-file>`

Dry run the playbook using the following command:
`ansible-playbook -C <path-to-playbook-yaml-file>`

9. Monitoring using ELK Stack

Continuous monitoring enables management to review business processes for adherence to and deviations from their intended performance and effectiveness levels. Thanks to CM, DevOps professionals can observe and detect compliance issues and security threats. CM also helps teams study relevant metrics and aid in solving issues in real-time when they arise. Continuous monitoring enables management to review business processes for adherence to and deviations from their intended performance and effectiveness levels. Thanks to CM, DevOps professionals can observe and detect compliance issues and security threats. CM also helps teams study relevant metrics and aid in solving issues in real-time when they arise.

a. Elasticsearch

Elasticsearch is a free and open-source search and analytics engine based on the Apache Lucene library that was first released in 2010. It's equipped with a rich and powerful HTTP RESTful API that enables you to perform fast searches in near real-time. Elasticsearch is developed in Java, supporting clients in many different languages, such as PHP, Python, C#, and Ruby.

In the context of using ELK as a tool for log management and analytics, Elasticsearch is in charge of indexing and storing data. You can read more about Elasticsearch in our Elasticsearch tutorial, from basic concepts to how it works, the benefits of using Elasticsearch, and use cases.

Download the Elasticsearch archive for your OS
<https://www.elastic.co/downloads/elasticsearch>

b. Kibana

Kibana is a free and open-source analysis and visualization layer that works on top of Elasticsearch and Logstash. It's actually the preferred choice for visualizing logs stored in Elasticsearch. Kibana makes it really easy to search, analyze, and visualize large volumes of data, as well as to detect trends and patterns. The dashboard features various interactive

charts and allows for customization, depending on what team in the company uses it.

Download the Kibana archive for your OS,
<https://www.elastic.co/downloads/kibana>

For visualization, you need to create index patterns so that the elasticsearch database can be queried. For that, go to Management → Stack Management

Go to Kibana and find Index Patterns → Create Index Pattern

There, specify the index pattern that you created in your logstash config file followed by an asterisk (*). After that, select @timestamp as the Time field and click on create. Now go to Explore → Dashboard to visualize the data.

c. Logstash

Logstash is a free and open-source log aggregator and processor that works by reading data from many sources and sending it to one or more destinations for storage or stashing – in this case, when using ELK for data analytics, to Elasticsearch. However, along the way, data is processed by filtering, massaging and shaping it to reach a uniform and structured view. Logstash is equipped with ready-made inputs, filters, codecs, and outputs, to help you extract relevant, high-value data from your logs.

Similar to Elasticsearch, Logstash too has a rich library of plugins, allowing it to collect, convert, and enrich various log types, from system logs to web server logs, error logs, and app logs.

Download the Logstash archive for your OS,
<https://www.elastic.co/downloads/logstash>

Or use cloud service for the above. But the configuration file we have to be changed accordingly

Syntax for configuration file for logstash:

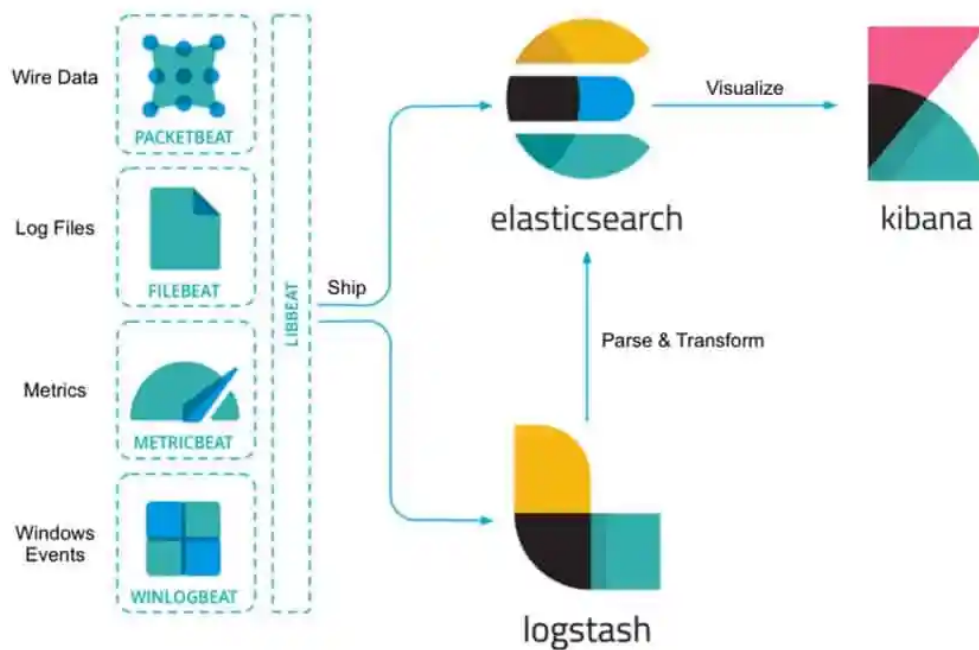
```
input {  
  plugin {  
    settings  
  }  
}
```

```
filter {  
  plugin {  
    settings  
  }  
}  
  
output {  
  plugin {  
    settings  
  }  
}
```

To run logstash, go to the logstash directory and run the following command:

```
./bin/logstash -f /path/to/your/logstash.conf/file
```

Elastic (ELK) Stack Architecture



10. Complete Pipeline of Jenkins

Install Plugins

Go to Manage Jenkins → Manage Plugins → Available → <Choose Plugins> → Install Without restart

Install the following plugins

- a) Git
- b) GitHub plugin
- c) Maven Integration plugin
- d) Docker plugin
- e) Pipeline
- f) Ansible Plugin

The screenshot shows the Jenkins Plugin Manager interface. On the left, there's a sidebar with 'Dashboard' and 'Plugin Manager' (selected). Below 'Plugin Manager' are links for 'Back to Dashboard' and 'Manage Jenkins'. The main area is titled 'Plugin Manager' and has tabs for 'Updates', 'Available' (selected), 'Installed', and 'Advanced'. A search bar is on the right. Below the tabs, there's a table of available plugins. The table has columns for 'Install', 'Name', and 'Released'. Three plugins are listed: 'Javadoc' (217.v905b_86277a_2a_), 'Maven Integration' (3.18), and 'WMI Windows Agents' (1.8). Each plugin has a checkbox in the 'Install' column and a description. At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. A status bar at the bottom right says 'Update information obtained: -664 ms ago'.

| Install | Name | Released |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <input type="checkbox"/> | Javadoc 217.v905b_86277a_2a_ This plugin adds Javadoc support to Jenkins. | 2 mo 8 days ago |
| <input type="checkbox"/> | Maven Integration 3.18 Build Tools This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (JUnit, ...). | 1 mo 18 days ago |
| <input type="checkbox"/> | WMI Windows Agents 1.8 Agent Management windows Allows you to setup agents on Windows machines over Windows Management Instrumentation (WMI) | 11 mo ago |

Configure Ansible plugin

Manage Jenkins → Global tool configuration → Add Ansible

Provide the name and the path of ansible on your local system which can be found by the command:

\$ whereis ansible

Ansible installations

Add Ansible

Ansible

Name

ansibleenv

Path to ansible executables directory

/usr/bin/

☐ Install automatically ?

Delete Ansible

Add Ansible

List of Ansible installations on this system

Configure Docker Hub

Go to Credentials → System → Provide username and password of Docker Hub
Provide your Docker Hub credentials here.

 **Credentials**

| T | P | Store ↓ | Domain | ID | Name |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|---------|----------|--------------------------------------|------------------------------|
|  |  | Jenkins | (global) | a33b090a-d58b-4ebf-a74d-662f5ee612d7 | Secret text |
|  |  | Jenkins | (global) | akshathkaushal7 | My dockerhub id and password |

Icon: S M L

Stores scoped to Jenkins

| P | Store ↓ | Domains |
|-------------------------------------------------------------------------------------|---------|----------------------------------------------------------------------------------------------|
|  | Jenkins |  (global) |

Creating a Jenkins Pipeline

The aim of the pipeline is to orchestrate different tasks like pulling code, building, testing and deploying sequentially. This helps in automating the process of code building and deployment.

To create a pipeline, do

Jenkins → New Item → Enter the name of the pipeline → Pipeline → OK

This will create a bare pipeline with no tasks.

Triggering the pipeline whenever changes are made in the code repository on github:

For this, first select the option “GitHub project” and provide the URL, then select the option “GitHub hook for SCM polling” option under Build Triggers.

The screenshot shows the Jenkins configuration interface for a new Pipeline item. The 'General' tab is selected. Under the 'Project type' section, 'GitHub project' is checked. The 'Project url' field is populated with 'https://github.com/akshathkaushal/testrepo/'. Below this, there are several unchecked checkboxes: 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterised', and 'Throttle builds'. An 'Advanced...' button is located to the right of these options. The 'Build Triggers' section contains several unchecked checkboxes: 'Build after other projects are built', 'Build periodically', 'Poll SCM', 'Disable this project', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'. The 'GitHub hook trigger for GITScm polling' checkbox is checked. At the bottom of the configuration area, there are 'Save' and 'Apply' buttons.

After this, enter the pipeline script either manually or through a Jenkinsfile. For the latter, create a file with the name “Jenkinsfile” and add the pipeline script into it and select the option “Pipeline script from SCM” under Pipeline. And select Git as the SCM and provide the path to your repository (with the extension .git) Also, specify the branch which has to be built and finally the path to your Jenkinsfile.

General

Build Triggers

Advanced Project Options

Pipeline

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/akshathkaushal/SPE-MiniProject.git

Credentials ?

- none -

Add

Advanced...

Save

Apply

General

Build Triggers

Advanced Project Options

Pipeline

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Script Path ?

Jenkinsfile

Save

Apply

Jenkinsfile

```

pipeline {
    agent any
    tools {
        maven 'MAVEN'
    }
    stages {
        stage('Download the git repo') {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/akshathkaushal/SPE-MiniProject.git']]])
            }
        }
        stage('Build and Test maven') {
            steps {
                script {
                    sh 'mvn -Dmaven.test.failure.ignore=true clean package'
                }
            }
            post {
                success {
                    junit '**/target/surefire-reports/TEST-*.xml'
                    archiveArtifacts 'target/*.jar'
                }
            }
        }
        stage('Build docker image') {
            steps {
                script {
                    sh 'docker build -t akshathkaushal7/spe-miniproject .'
                }
            }
        }
        stage('Push docker image to Dockerhub') {
            steps {
                script {
                    withCredentials([string(credentialsId: 'a33b090a-d58b-4ebf-a74d-662f5ee612d7', variable: 'dockerhubpwd')]) {
                        sh 'docker login -u akshathkaushal7 -p ${dockerhubpwd}'
                        sh 'docker push akshathkaushal7/spe-miniproject'
                    }
                }
            }
        }
        stage('Deploy using Ansible') {
            steps {
                ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking:true, installation: 'ansibleenv', inventory: 'deploy-docker/inventory', playbook: 'deploy-docker/deploy-image.yml', sudoUser:null
            }
        }
    }
}

```

Now, in order to run the script automatically, you need to create a webhook to run the build. For this, first create an ssh key in your local system using the following command:
\$ ssh-keygen -t rsa

This will create an ssh key in your filesystem and the path can be specified while creating the key. Copy this key

```

akshathkaushal-Latitude-5420 % ~/eclipse-workspace/spe.miniproject:(1d21h5n|git@main|)
2397 ± ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/akshathkaushal/.ssh/id_rsa): /home/akshathkaushal/.ssh/id_temp
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/akshathkaushal/.ssh/id_temp
Your public key has been saved in /home/akshathkaushal/.ssh/id_temp.pub
The key fingerprint is:
SHA256:WREFIcW9IIPKzoIe2Z0BwhxQcVwJZ9GLjrkQrZdAhEo akshathkaushal@akshathkaushal-Latitude-5420
The key's randomart image is:
+-----[RSA 3072]-----+
|.B+o+==+
|E.+.o=+.
|o o + + o.o
|.o . oo.
|o o * *So
|+ * * +
|. o .
|. .
+-----[SHA256]-----+
akshathkaushal-Latitude-5420 % ~/eclipse-workspace/spe.miniproject:(1d22h36m|git@main|)
2398 ± cat /home/akshathkaushal/.ssh/id_temp.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgODh5vaGLK6Yf1c2rjU67jUXQg8j8t5c5PwxBkHkWEEdg0fgthU77PFLS0XCP7lpMD9nMTgSWtay6qnhyv0dca02ySnPs0mAuAgip0RBCKNjVwCBiPcHu+jg9VfnFAj0foj
mBYGqzGoaomh7xUpGJe17d1/YmdMiUJu0Kw7jTpy1/JXfw0LQ5T2kiuInHJCa/cFc12yvalqG5CQ2Kvq4kXyVewR4uTAJmFuLrYXud57TsJ8HJrflRY2J0EZry20ppjv0t06zhMv9U2UD0RW3sK8lWtaPduur+3gxZI
ovHVaUGiWcmq7GmFjQuU/QZ9eBMsGrGcW0fheQZpVW17d6McJLxjIA1GETvk115hw0WTawPxbjMncpDuvBac0KLnJXnBsxzTZjDBZMSLXccvWTW2Z0SA1W0vrDgE6zna3xmwuZwoSq6c0qeZ93q7+mG3KE2xps2X4YJJC
x2SKVQ5WxjIx8HwdldXNxyYGZ0h3Z0BYaC4B8hxS8dak= akshathkaushal@akshathkaushal-Latitude-5420
akshathkaushal-Latitude-5420 % ~/eclipse-workspace/spe.miniproject:(1d22h36m|git@main|)
2399 ±

```

Go to settings → Deploy keys and add the copied key

Deploy keys / Add new

Title

SPEMiniproject

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGCzTSILieUKvMDGk06ttP+HOpP9dX4DWNvoRHAKe0/fh2plufYOyhVnj9NY
8OGCH//jtdDtmQESzXfEgL8zyrZxp9FfU2Y1rH9wZe0a4dlveOkW8y6UZCJh0fTEeSJwf0vlcvauF0wXh6GKiCo4DmWb
NYAP11j9hzH0nqUvryTfmAW+OcpXZmBMBEKfRRqou4FYAGtNhS0zdUBAl1KbLEBWFZDJuOykiJRGiH8/IPaa9gROO
8PsT8GYXTQ7blyMeeE0l47Uwdz04rYmgILaFdUwNqlzJm1yj0Hkq34a+SZvldNjZfELMkF4nckXu76G+RxfrWqvBreUzT
cO8j5/A1P+5HVAu5qv3SL/Lj5JHfxi+aun8M686QCurMtN7koD7CAuCwuJUvH+HX1xBK4lwbeB7k0jLw+/l2edZpimve+6d
UJYhgVtGJsK9xAJbyFUNqj8khOL56GyMo+uJ7b8F8X7X3a49LmEWwSMSQ0197YgXbBIU9Z2EnuNSyRhsuHtORs=
akshathkaushal@akshathkaushal-Latitude-5420
```



☐ Allow write access

Can this key be used to **push** to this repository? Deploy keys always have pull access.

Add key

Now go to Settings → webhooks

If you Jenkins server is running on a public ip address, create a webhook by providing the ip address

Webhooks / Add webhook

We'll send a post request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).


Payload URL *

<your-ip-here>/github-webhook

Content type

application/x-www-form-urlencoded ⇅

Secret

 **This doesn't look like a valid URL.**

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

If you are running Jenkins server on localhost, you need ngrok to expose the port to the internet.

Install ngrok

```
$ curl -s https://ngrok-agent.s3.amazonaws.com/ngrok.asc | sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null
```

```
$ echo "deb https://ngrok-agent.s3.amazonaws.com buster main" | sudo tee /etc/apt/sources.list.d/ngrok.list
```

```
$ sudo apt update
```

```
$ sudo apt install ngrok
```

Once installed, expose the port 8080 or whichever port is your Jenkins server running by the following command

```
$ ngrok http 8080
```

```
ngrok by @inconshreveable

Session Status      online
Account             Akshath (Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://a3c7-119-161-98-68.ngrok.io -> http://localhost:8080
Forwarding           https://a3c7-119-161-98-68.ngrok.io -> http://localhost:8080

Connections          ttl    opn    rt1    rt5    p50    p90
                   0      0      0.00   0.00   0.00   0.00
```

Now, copy the http url created by ngrok and paste it as the ip address of the webhook followed by “/github-webhook/”

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://a3c7-119-161-98-68.ngrok.io/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

- ☒ Just the push event.
- ☐ Send me **everything**.
- ☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

Webhooks

[Add webhook](#)

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://d0b3-203-192-253-115.ngrok...> (push)

[Edit](#)
[Delete](#)

Now, whenever you push new changes to your GitHub, the pipeline will be triggered.

Adding Docker file for containerizing the project

For this, you need to create a Dockerfile in order to build a docker image which can then be pushed to Dockerhub and pulled by ansible into remote hosts

Dockerfile

```
1 FROM openjdk:11
2 COPY ./target/spe.miniproject-0.0.1-SNAPSHOT-jar-with-dependencies.jar ./
3 WORKDIR ./
4 CMD ["java", "-cp", "spe.miniproject-0.0.1-SNAPSHOT-jar-with-dependencies.jar", "Calculator"]
```

In this, we run the JAR file that is created by the maven when the image is executed in a container.

Adding Ansible configuration files for automatic pulling of docker image into a remote host

For this, Ansible needs to be configured which we already did above. Now, we need three files

- a) Ansible playbook
- b) Inventory file for hosts
- c) Ansible.cfg file

Create a separate folder in your project directory and create a) and b) files in it. Create c) in your project repository.

Playbook (deploy-image.yml)

```
---
- name: Pull docker image of Calculator
  hosts: all
  tasks:
    - name: Pull calculator image
      docker_image:
        name: akshathkaushal7/spe-miniproject:latest
        source: pull
```

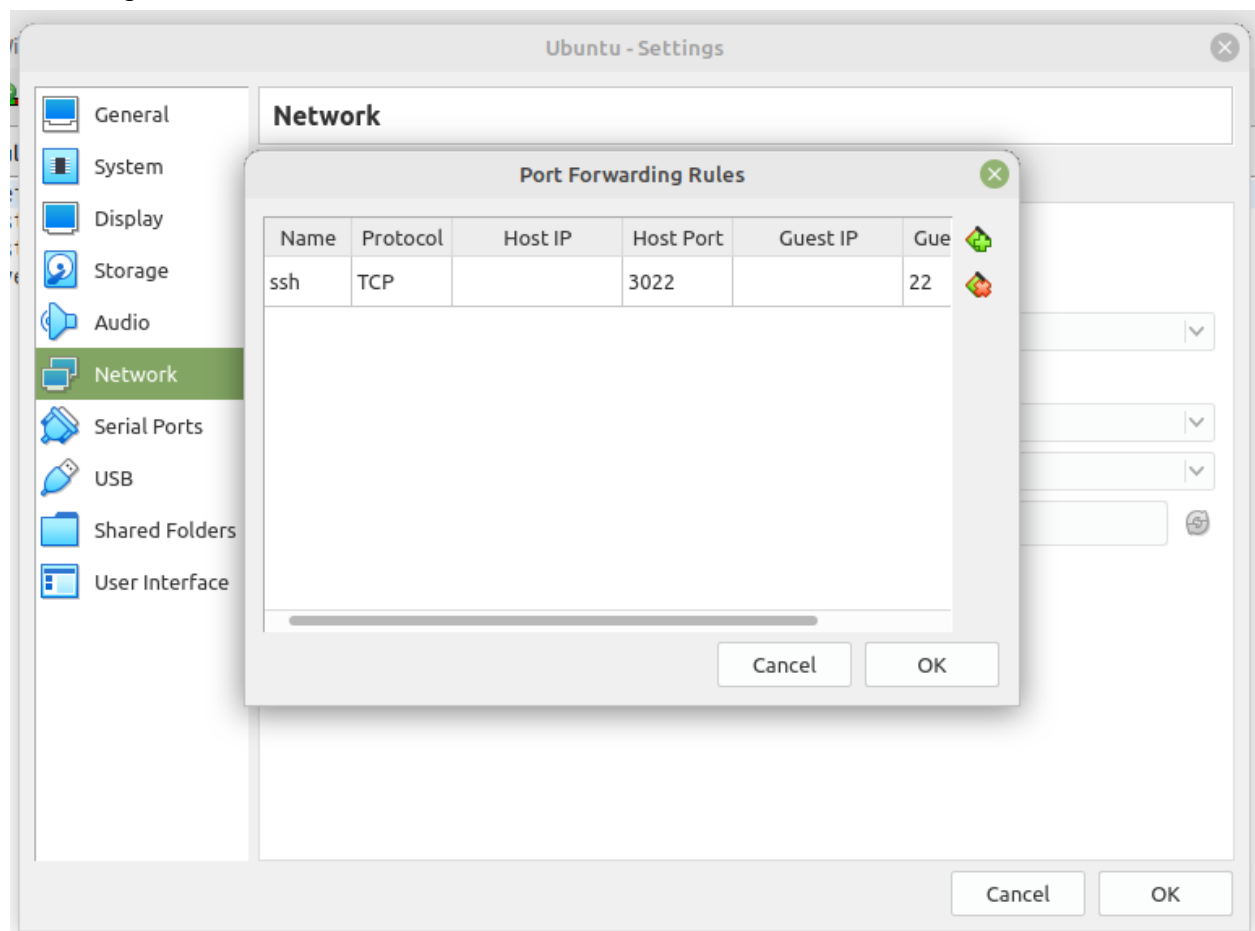

Inventory

```
[remotehosts]
localhost ansible_port=3022 ansible_user=akshath
```

ansible.cfg

```
[defaults]
host_key_checking = False
host_key_check = False
inventory = ./deploy-docker/inventory
```

In this, I have created a separate VM using Oracle's virtualbox and have installed openssh-server in it. Also, I have mapped the port 3022 of my host to the port 22 of the VM using the Virtualbox's console.



Add the key from your host to the VM using the following command

```
$ ssh-copy-id <host name>@localhost -p <port no.>
```

```

akshathkaushal-Latitude-5420 ~ /eclipse-workspace/spe.miniproject:(1d23h1m|git@main)
2417 ± ssh-copy-id akshath@localhost -p 3022
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

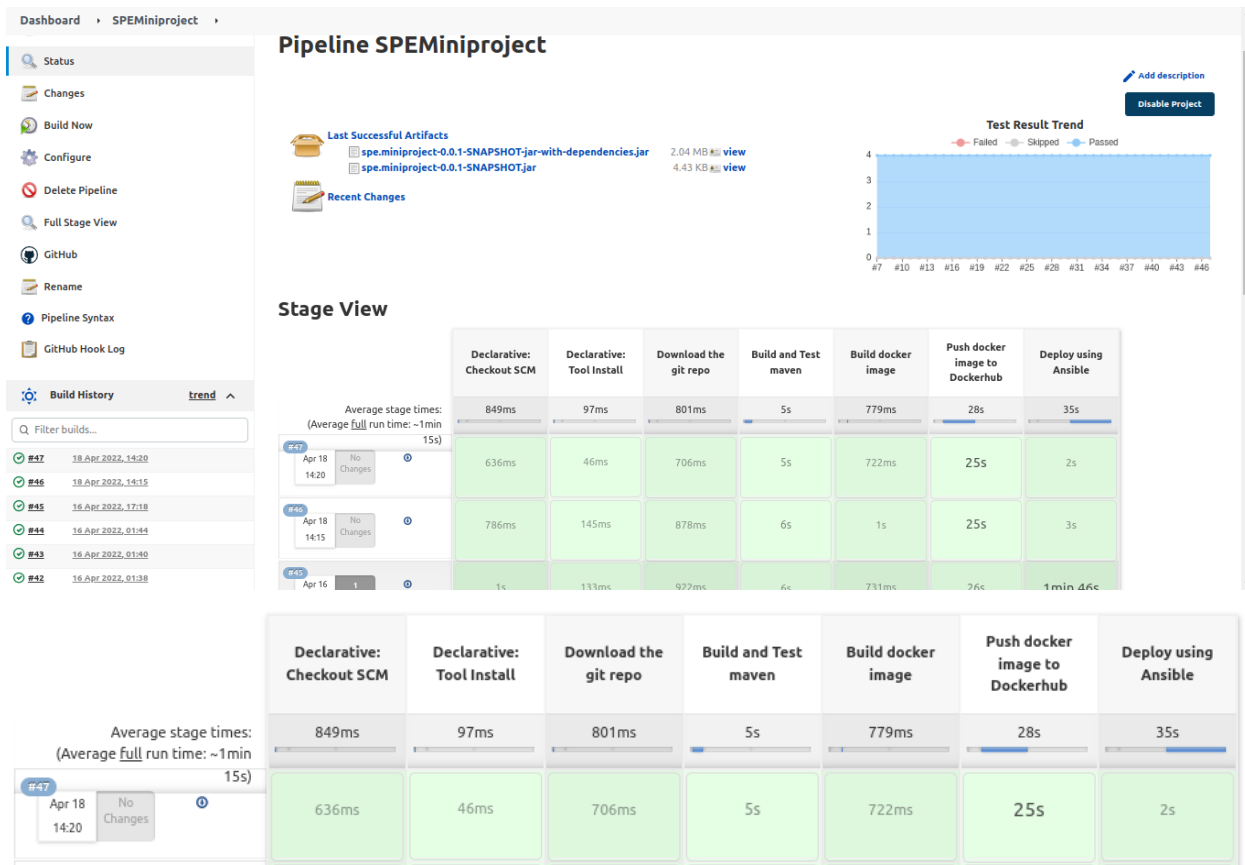
Number of key(s) added: 1

Now try logging into the machine, with: "ssh -p '3022' 'akshath@localhost'"
and check to make sure that only the key(s) you wanted were added.

```

Also, do the same with the jenkins user since Jenkins will run on jenkins user.

Now, Ansible will be able to deploy the docker image onto the virtual machine. This is not limited to just the local VM, it can be done for any VM that is running in the cloud.



Now that the docker image is created in the virtual machine, you can run the code and monitor the output using ELK stack

```
akshath@akshath-VirtualBox:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
akshath@akshath-VirtualBox:~$ docker images
REPOSITORY      TAG    IMAGE ID      CREATED        SIZE
akshathkaushal7/spe-miniproject   latest  4d26fabdb1fa   About a minute ago  662MB
akshath@akshath-VirtualBox:~$ docker run --name spe-container -it akshathkaushal7/spe-miniproject:latest
Options:
1. Square root
2. Factorial
3. Natual logarithm
4. Raise to power
Press any other key to exit
Enter choice:
1
Option chosen: Square root
Enter the number:
2
WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
1.4142135623730951
Enter choice:
2
Option chosen: Factorial
Enter the number:
3
6
Enter choice:
c
Exiting...
bye bye!
akshath@akshath-VirtualBox:~$
```

Note: I am running elastic search and kibana in docker containers due to memory constraints and logstash locally.

For this, run the **docker-compose.yml** file in the project directory by

\$ docker-compose up -d --remove-orphans

```
akshathkaushal-Latitude-5420 35 ~/eclipse-workspace/spe.miniproject: (1d23h11m|git@main!)
2419 ± docker-compose up -d --remove-orphans
Starting speminiproject_elasticsearch_1 ... done
Starting speminiproject_kibana_1      ... done
akshathkaushal-Latitude-5420 35 ~/eclipse-workspace/spe.miniproject: (1d23h15m|git@main!)
2420 ±
```

And logstash as

\$./bin/logstash -f /path/to/your/conf/file

Calculator_logger_logstash.conf

```
input {
  file {
    path => "/home/akshathkaushal/eclipse-workspace/spe.miniproject/Calculator.log"
    start_position => "beginning"
  }
}

filter {
  grok {
    match => [
      'message', '%{HTTPDATE:timestamp_string} \[%{GREEDYDATA:thread}\] \[%{LOGLEVEL:level}\] %{GREEDYDATA:logger} \[%{GREEDYDATA:action}\] \- %{GREEDYDATA:line}'
    ]
  }

  date {
    match => ["timestamp_string", "dd/MMM/YYYY:HH:mm:ss SSS"]
  }

  mutate {
    remove_field => [timestamp_string]
  }
}

output {
  elasticsearch {
    hosts => ["http://localhost:9200"]
    index => "calculator_index"
  }

  stdout {
    codec => rubydebug
  }
}
```

In this file:

1. Input - An input plugin enables a specific source of events to be read by Logstash

<https://www.elastic.co/guide/en/logstash/current/input-plugins.html>

2. Filter - A filter plugin performs intermediary processing on an event. Filters are often applied conditionally depending on the characteristics of the event

<https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>

3. Output - An output plugin sends event data to a particular destination. Outputs are the final stage in the event pipeline

<https://www.elastic.co/guide/en/logstash/current/output-plugins.html>

```

akshathkaushal-Latitude-5420 % ~:
2422 o cd Documents/Software/logstash-7.9.2
akshathkaushal-Latitude-5420 % ~/Documents/Software/logstash-7.9.2:
2423 o ./bin/logstash -f /home/akshathkaushal/eclipse-workspace/spe.miniproject/Calculator_logger logstash.conf

[2022-04-18T16:34:16.484][INFO ][logstash.agent ] Pipelines running (:count=>1, :running_pipelines=>[:main], :non_running_pipelines=>[])
[2022-04-18T16:34:16.676][INFO ][logstash.agent ] Successfully started Logstash API endpoint {:port=>9600}
{
  "host" => "akshathkaushal-Latitude-5420",
  "@version" => "1",
  "message" => "18/Apr/2022:14:13:07 855 [Calculator.java] [INFO] Calculator [Factorial function result for input 7]: 5040",
  "@timestamp" => 2022-04-18T11:04:16.800Z,
  "path" => "/home/akshathkaushal/eclipse-workspace/spe.miniproject/Calculator.log",
  "tags" => [
    [0] "_grokparsefailure"
  ]
}
{
  "host" => "akshathkaushal-Latitude-5420",
  "@version" => "1",
  "message" => "18/Apr/2022:14:13:07 857 [Calculator.java] [INFO] Calculator [Power function result for inputs 3.0, 4.0]: 81.0",
  "@timestamp" => 2022-04-18T11:04:16.801Z,
  "path" => "/home/akshathkaushal/eclipse-workspace/spe.miniproject/Calculator.log",
  "tags" => [
    [0] "_grokparsefailure"
  ]
}
{
  "host" => "akshathkaushal-Latitude-5420",
  "@version" => "1",
  "message" => "18/Apr/2022:14:13:07 856 [Calculator.java] [INFO] Calculator [Power function result for inputs 2.0, 3.0]: 8.0",
  "@timestamp" => 2022-04-18T11:04:16.801Z,
  "path" => "/home/akshathkaushal/eclipse-workspace/spe.miniproject/Calculator.log",
  "tags" => [
    [0] "_grokparsefailure"
  ]
}

```

Now, follow the steps above to create an index in kibana and the visualization.



11. Links and References

- [Link to GitHub repository](#)
- [Link to DockerHub repository](#)

- <https://ngrok.com/docs>
- <https://www.elastic.co/elasticsearch/>
- <https://www.elastic.co/kibana/>
- <https://www.elastic.co/logstash/>
- <https://docs.docker.com/>
- <https://maven.apache.org/guides/>
- <https://docs.ansible.com/ansible-core/devel/index.html>
- <https://about.gitlab.com/images/press/git-cheat-sheet.pdf>