

SPE MINI PROJECT

BY

PANKAJ KHEMANI

IMT2018054

Problem Statement:

Create a scientific calculator program with user menu driven operations

- Addition - $x+y$
- Subtraction - $x-y$
- Multiplication - $x*y$
- Division - x/y
- Square root function - \sqrt{x}
- Factorial function - $x!$
- Natural logarithm (base e) - $\ln(x)$
- Power function - x^b

We have to use any set of DevOps tool chains and build a pipeline that includes the following-

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.

What and Why of DevOps?

DevOps can be best explained as people working together to conceive, build and deliver secure software at top speed. DevOps practices enable software developers (devs) and operations (ops) teams to accelerate delivery through automation, collaboration, fast feedback, and iterative improvement.

Stemming from an Agile approach to software development, a DevOps delivery process expands on the cross-functional approach of building and shipping applications in a faster and more iterative manner. In adopting a DevOps development process, we are making a decision to improve the flow and value delivery of our application by encouraging a more collaborative environment at all stages of the development cycle.

DevOps represents a change in mindset for IT culture. In building on top of Agile, lean practices, and systems theory, DevOps focuses on

incremental development and rapid delivery of software. Success relies on the ability to create a culture of accountability, improved collaboration, empathy, and joint responsibility for business outcomes.

By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster. Hence, we use DevOps.

Tools Used:

1. Java (OpenJDK8)
2. Maven
3. Git
4. GitHub
5. IntelliJ IDEA
6. Docker
7. Docker Hub
8. Jenkins
9. Ansible
10. VM Azure Cloud
11. Ngrok
12. Log4j
13. Junit
14. Docker-ELK
15. Logstash
16. Kibana

Part 1 - Setup for Environments and Tools:

1) Java(OpenJDK 8):

Java is an object-oriented programming language that produces software for multiple platforms(platform independent). When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS.

Steps to install:

- \$ sudo apt update
- \$ sudo apt install openjdk-8-jdk
- To configure \$ JAVA_HOME path
 - \$ sudo vim /etc/environment
 - Add this path at the end of the file:
 - JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
- Restart the system to apply changes
- \$ source /etc/environment
- Verify JAVA_HOME path
 - \$ echo \$JAVA_HOME

```
pankaj@Coder:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64
pankaj@Coder:~$ █
```

2) Maven:

Maven is a project management and comprehension tool that provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle. Maven just needs a pom.xml(config file for dependencies) in a project and then it will automatically start building the dependencies for the project.

Steps to install:

- \$ sudo apt install maven
- \$ mvn --version

```
pankaj@Coder:~$ mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 1.8.0_312, vendor: Private Build, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.13.0-39-generic", arch: "amd64", family: "unix"
```

3) Git:

Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to track changes in the source code, enabling multiple developers to work together on non-linear development.

Commands in Git

- Create Repositories
git init
- Make Changes
add
commit
status
- Parallel Development
branch
merge
rebase
- Sync Repositories
push
pull
add origin

Steps to install:

- \$ sudo apt update
- \$ sudo apt install git
- \$ git config --global user.name "Your name"
- \$ git config --global user.email "Your email"
- \$ git --version

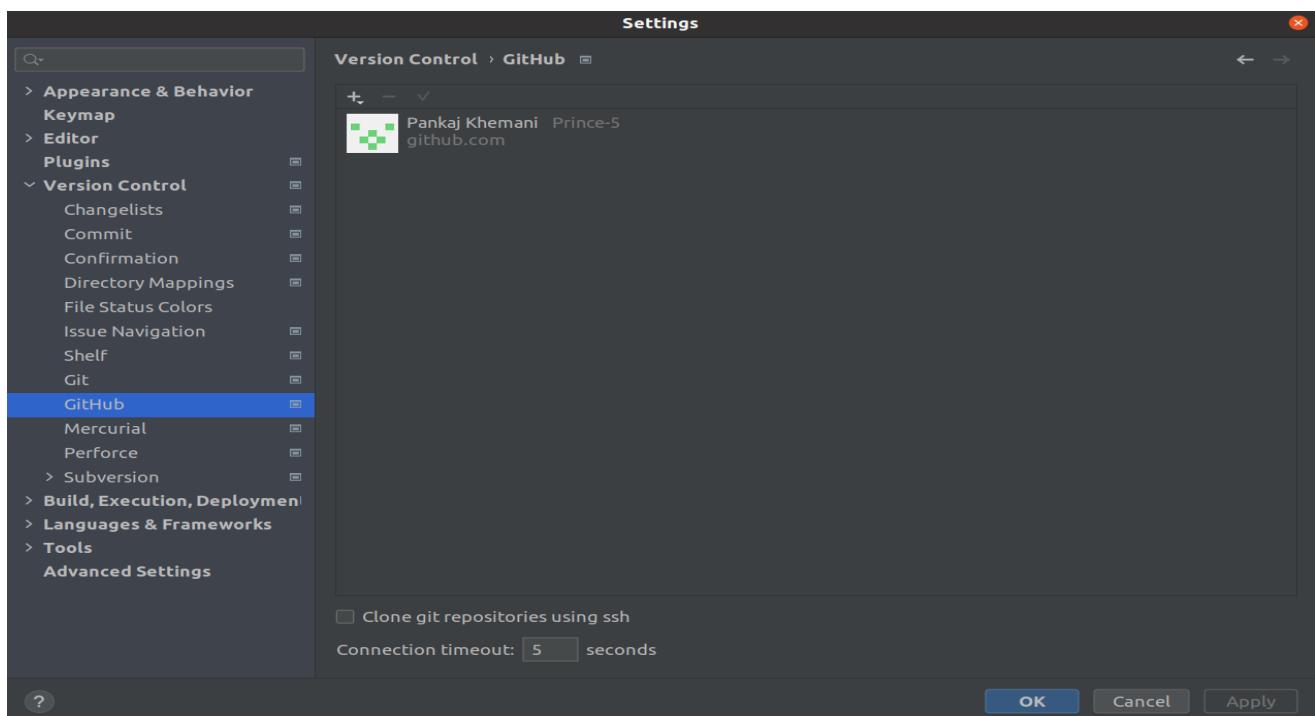
```
pankaj@Coder:~$ git --version
git version 2.25.1
```

4) Github:

GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration. GitHub's interface is user-friendly enough so even novice coders can take advantage of Git. Without GitHub, using Git generally requires a bit more technical savvy and use of the command line. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

Steps to use and link account with IntelliJ:

- Signup for github at <https://github.com/join>
- Create a new repository
- Enter repo name and click create
- In IntelliJ go to settings->Version Control->GitHub and click +
- Log in to your account on Github via password or token



5) IntelliJ IDEA:

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development. IntelliJ allows plugins, which may be used to extend the IDE's capability. Plugins may be downloaded and installed via IntelliJ's plugin repository website or the IDE's built-in plugin search and install tool.

Steps to install and configure build tools:

- Download IntelliJ IDEA from
<https://www.jetbrains.com/idea/download> or directly download it from the ubuntu software store.
- After downloading from the ubuntu store it automatically gets installed.
- Open IntelliJ and go to settings->build, executions, deployment->build tools->maven
- Select use plugin registry then click on save
- Your IntelliJ IDEA is now ready to use.



IDEA Community

Capable & Ergonomic Java IDE

★★★★★ (242)

Remove

```
private LanguageFolding() { super("com.intellij.lang.FoldingBuilder"); }

@NotNull
public static FoldingDescriptor[] buildFoldingDescriptors(@NotNull FoldingBuilder builder, @NotNull PsiElement root, @NotNull Document document, boolean quick) {
    if (!DumbService.isDumbAware(builder) && DumbService.getInstance(root.getProject())
        .isDumb()) {
        return FoldingDescriptor.EMPTY;
    }
    if (builder instanceof FoldingBuilderEx) {
        return ((FoldingBuilderEx) builder).buildFoldingRegions(root, document, quick);
    }
    final ASTNode astNode = root.getNode();
    if (astNode == null || builder == null) {
        return FoldingDescriptor.EMPTY;
    }
    return builder.buildFoldingRegions(astNode, document);
}

public FoldingBuilder forLanguage(@NotNull Language l) {
    FoldingBuilder cached = l.getUserData(getLanguageCache());
    if (cached != null) return cached;
    List<FoldingBuilder> extensions = forKey(l);
    FoldingBuilder result;
    if (extensions.isEmpty()) {
        Language base = l.getBaseLanguage();
        if (base != null) {
            result = forLanguage(base);
        } else {
            result = getDefaultValue();
        }
    } else {
        result = extensions.get(0);
    }
    return result;
}
```

IntelliJ IDEA Community Edition is a free and open-source edition of IntelliJ IDEA, the commercial Java IDE by JetBrains. IntelliJ IDEA Community Edition provides all the tools you need for Java, Groovy, Kotlin, Scala, and Android. It offers instant and clever code completion, on-the-fly code analysis, and reliable refactoring tools. Mission-critical tools such as integrated version controls systems and a wide variety of supported languages and frameworks are at hand — no plugin hustle included.

Settings

Build, Execution, Deployment > Build Tools > Maven

Appearance & Behavior

Editor

Version Control

Build, Execution, Deployment

Build Tools

Maven

Gradle

Gant

Compiler

Debugger

Remote Jar Repositories

Android

Coverage

Gradle-Android Compiler

Package Search

Required Plugins

Testing

Trusted Locations

Languages & Frameworks

Tools

Advanced Settings

Work offline

Use plugin registry

Execute goals recursively

Print exception stack traces

Always update snapshots

Output level: Info

Checksum policy: No Global Policy

Multiproject build fail policy: Default

Thread count: -T option

Maven home path: Bundled (Maven 3) (Version: 3.8.1)

User settings file: /home/pankaj/.m2/settings.xml

Local repository: /home/pankaj/.m2/repository

Use settings from .mvn/maven.config

OK Cancel Apply

6) Jenkins:

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat. It supports version control tools, including Git, Mercurial, AccuRev, CVS, Subversion, etc.

Steps to install and setup jenkins:

- Add Key
 - \$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | apt-key add -
- Add the repository, update local package index and install
 - \$ 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 jenkins start
 - \$ sudo service jenkins status

```
pankaj@Coder:~$ sudo service jenkins status
[sudo] password for pankaj:
● Jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2022-04-18 11:45:43 IST; 3h 20min ago
    Main PID: 985 (java)
       Tasks: 48 (limit: 8660)
      Memory: 758.0M
         CGroup: /system.slice/jenkins.service
             └─985 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Apr 18 14:57:54 Coder jenkins[985]: 2022-04-18 09:27:54.092+0000 [id=148]      INFO      c.n.j.p.d.DockerContainerWatchdog$Statistics#writeStatisticsToLog: Watchdog Sta...
Apr 18 14:57:54 Coder jenkins[985]: 2022-04-18 09:27:54.092+0000 [id=148]      INFO      c.n.j.p.d.DockerContainerWatchdog#loadNodeMap: We currently have 0 nodes assign...
Apr 18 14:57:54 Coder jenkins[985]: 2022-04-18 09:27:54.092+0000 [id=148]      INFO      c.n.j.p.d.DockerContainerWatchdog#execute: Docker Container Watchdog check has ...
Apr 18 14:57:54 Coder jenkins[985]: 2022-04-18 09:27:54.092+0000 [id=148]      INFO      hudson.model.AsyncPeriodicWork$lambda$doRun$1: Finished DockerContainerWatchdog...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.091+0000 [id=149]      INFO      hudson.model.AsyncPeriodicWork$lambda$doRun$1: Started DockerContainerWatchdog...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.091+0000 [id=149]      INFO      c.n.j.p.d.DockerContainerWatchdog#loadNodeMap: We currently have 0 nodes assign...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.092+0000 [id=149]      INFO      c.n.j.p.d.DockerContainerWatchdog$Statistics#writeStatisticsToLog: Watchdog Sta...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.092+0000 [id=149]      INFO      c.n.j.p.d.DockerContainerWatchdog#execute: Docker Container Watchdog has been t...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.092+0000 [id=149]      INFO      c.n.j.p.d.DockerContainerWatchdog#loadNodeMap: We currently have 0 nodes assign...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.092+0000 [id=149]      INFO      c.n.j.p.d.DockerContainerWatchdog#execute: Docker Container Watchdog check has ...
Apr 18 15:02:54 Coder jenkins[985]: 2022-04-18 09:32:54.092+0000 [id=149]      INFO      hudson.model.AsyncPeriodicWork$lambda$doRun$1: Finished DockerContainerWatchdog...
Lines 1-19/19 (END)
```

- Jenkins runs on the localhost at port 8080 by default.
- Open localhost:8080 in a browser.
- Print default password to login into jenkins
 - \$ cat /var/lib/jenkins/secrets/initialAdminPassword

- Choose to install suggested plugins, configure the user and login using Username and password.

localhost:8080/login?from=%2F

Welcome to Jenkins!

JenkinsAdmin

Sign in

Keep me signed in

[OBJ]

The screenshot shows the Jenkins dashboard at localhost:8080. The left sidebar contains links for New Item, People, Build History, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, and New View. The main area displays a table with one item: SPEMiniPipeline. The table columns are S (Status), W (Work), Name (SPEMiniPipeline), Last Success (21 hr #30), Last Failure (21 hr #29), Last Duration (46 sec), and Fav (Favorite). Below the table are links for Icon legend, Atom feed for all, Atom feed for failures, and Atom feed for just latest builds. At the bottom right, there are links for REST API and Jenkins 2 222 2.

7) Docker:

Docker is a Linux-based, open-source containerization platform that developers use to build, run, and package applications for deployment using containers. Unlike virtual machines, Docker containers offer OS-level abstraction with optimum resource utilization, Interoperability, Efficient build and test.

Steps to install:

- sudo apt update
- \$ 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"

- \$ sudo apt install docker-ce
- \$ sudo service docker status

```
pankaj@Coder:~$ sudo service docker status
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2022-04-18 11:45:40 IST; 3h 30min ago
    TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 1830 (dockerd)
     Tasks: 25
    Memory: 66.0M
      CGroup: /system.slice/docker.service
              └─1830 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.439259175+05:30" level=warning msg="Your kernel does not support CPU realtime scheduler"
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.439308347+05:30" level=warning msg="Your kernel does not support cgroup blkio weight"
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.439313487+05:30" level=warning msg="Your kernel does not support cgroup blkio weight_device"
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.439890049+05:30" level=info msg="Loading containers: start."
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.644946709+05:30" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. Daemon"
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.676558028+05:30" level=info msg="Loading containers: done."
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.712735715+05:30" level=info msg="Docker daemon" commit=87a90dc graphdriver(s)=overlay2 version=20.10.14
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.713221036+05:30" level=info msg="Daemon has completed initialization"
Apr 18 11:45:40 Coder systemd[1]: Started Docker Application Container Engine.
Apr 18 11:45:40 Coder dockerd[1830]: time="2022-04-18T11:45:40.736630747+05:30" level=info msg="API listen on /run/docker.sock"
lines 1-21/21 (END)
```

- Executing docker without sudo
 - \$ sudo groupadd docker
 - \$ sudo usermod -aG docker \${USER}
 - \$ newgrp docker
 - Restart the OS
 - Check if docker runs without sudo
 - \$ docker run hello-world

```
pankaj@Coder:~$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

If the above steps for executing docker without sudo don't work then you can also try the below command if docker is giving any permission related error.

- \$ sudo chmod 666 /var/run/docker.sock

```
pankaj@Coder:~$ sudo chmod 666 /var/run/docker.sock
[sudo] password for pankaj:
pankaj@Coder:~$ █
```

8) Virtual Machine on Azure cloud:

Azure Virtual Machines (VM) is one of several types of on-demand, scalable computing resources that Azure offers. Typically, we choose a Azure VM because we need more control over the computing environment than the other choices offer.

Essentials

Resource group (move)	SPE	Operating system	: Linux (ubuntu 18.04)
Status	: Running	Size	: Standard DS12 v2 (4 vcpus, 28 GB memory)
Location	: Central India (Zone 1)	Public IP address	: 20.40.48.32
Subscription (move)	: Azure for Students	Virtual network/subnet	: spe-vnet/default
Subscription ID	: 2746d444-4820-42b8-ab12-1e6a5f64e6db	DNS name	: Not configured
Availability zone	: 1		

Tags (edit) : Click here to add tags

Properties **Monitoring** **Capabilities (7)** **Recommendations (1)** **Tutorials**

Virtual machine

Computer name	spe-ansible
Health state	-
Operating system	Linux (ubuntu 18.04)
Publisher	Canonical
Offer	UbuntuServer
Plan	18.04-lts-gen2
VM generation	V2
Agent status	Ready
Agent version	2.7.1.0
Host group	None
Host	-
Proximity placement group	-
Colocation status	N/A
Capacity reservation group	-

Networking

Public IP address	20.40.48.32
Private IP address (IPv6)	-
Private IP address (IPv6)	-
Virtual network/subnet	spe-vnet/default
DNS name	Configure

Size

Size	Standard DS12 v2
vCPUs	4
RAM	28 GB

Disk

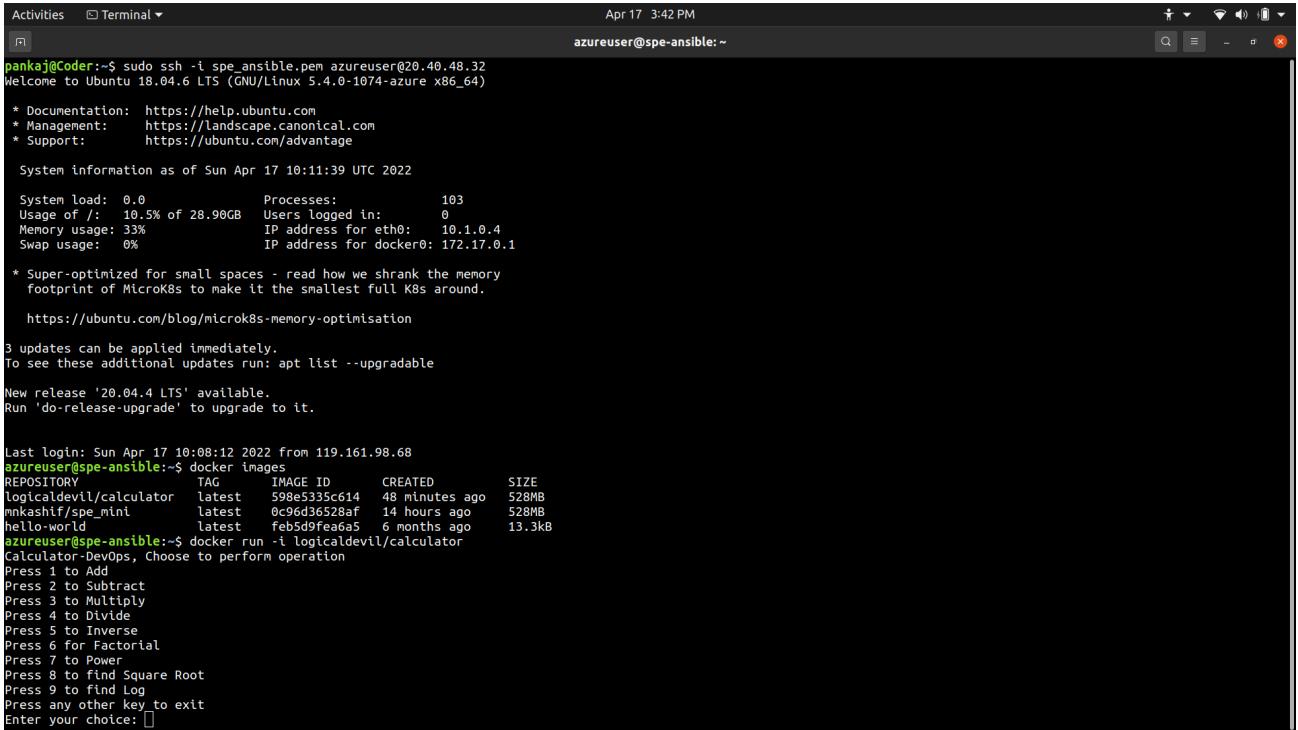
OS disk	spe-ansible_OsDisk_1_d4e0ba140314420896dee1523da6eac
Encryption at host	Disabled

We are creating this VM for checking the deployment of our calculator application. The Jenkins server runs on our local machine and in one of the stages of the jenkins pipeline, we use ansible to pull the docker image of our calculator application on the azure VM.

We connect to the azure VM by using the command,
\$ sudo ssh -i <path to .pem key file> <VMusername>@<ip>

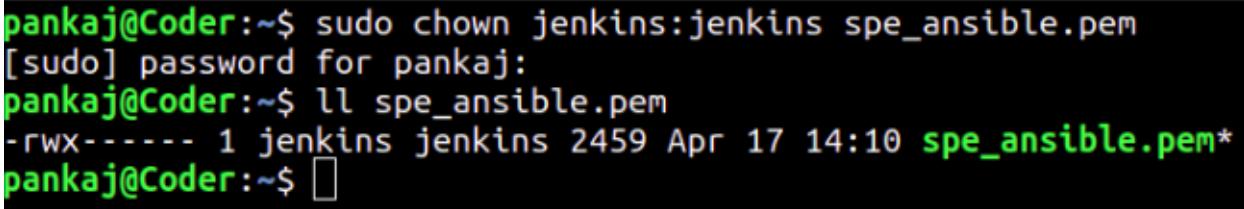
Then we install docker on the VM using pip.
\$ pip install docker

After our jenkins pipeline runs successfully, we simply check if our deployed calculator application works properly by running this command on the VM,
\$ docker run -i <dockerRepo Name>



```
Activities Terminal ▾ Apr 17 3:42 PM azureuser@spe-ansible:~  
pankaj@Coder:~$ sudo ssh -i spe_ansible.pem azureuser@20.40.48.32  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1074-azure x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
System information as of Sun Apr 17 10:11:39 UTC 2022  
  
System load: 0.0 Processes: 103  
Usage of /: 10.5% of 28.90GB Users logged in: 0  
Memory usage: 33% IP address for eth0: 10.1.0.4  
Swap usage: 0% IP address for docker0: 172.17.0.1  
  
* Super-optimized for small spaces - read how we shrank the memory  
footprint of MicroK8s to make it the smallest full K8s around.  
  
https://ubuntu.com/blog/microk8s-memory-optimisation  
  
3 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
New release '20.04.4 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Sun Apr 17 10:08:12 2022 from 119.161.98.68  
azureuser@spe-ansible:~$ docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
logicaldev1/calculator latest 598e535cc614 48 minutes ago 528MB  
mnkashif/spe_mini latest 0c96d36528af 14 hours ago 528MB  
hello-world latest feb5d9fea6a5 6 months ago 13.3kB  
azureuser@spe-ansible:~$ docker run -i logicaldev1/calculator  
Calculator-DevOps, Choose to perform operation  
Press 1 to Add  
Press 2 to Subtract  
Press 3 to Multiply  
Press 4 to Divide  
Press 5 to Inverse  
Press 6 for Factorial  
Press 7 to Power  
Press 8 to Find Square Root  
Press 9 to Find Log  
Press any other key to exit  
Enter your choice: □
```

If some error related to permission comes up, that may be because jenkins is not the owner of the file .pem key file. So, we can change the ownership of that file by using the chown command.



```
pankaj@Coder:~$ sudo chown jenkins:jenkins spe_ansible.pem  
[sudo] password for pankaj:  
pankaj@Coder:~$ ll spe_ansible.pem  
-rwx----- 1 jenkins jenkins 2459 Apr 17 14:10 spe_ansible.pem*  
pankaj@Coder:~$ □
```

9) Ansible:

Ansible is an open source DevOps tool which can help the business in configuration management, deployment, provisioning, etc. It is straightforward to deploy; it leverages SSH to communicate between

servers. It uses the playbook to describe automation jobs, and the playbook uses a very simple language YAML.

Steps to install:

- Install Ansible on Local machine
 - \$ sudo apt update
 - \$ sudo apt install ansible
- Check Ansible Version
 - \$ ansible --version

```
pankaj@Coder:~$ ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['~/home/pankaj/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.8.10 (default, Mar 15 2022, 12:22:08) [GCC 9.4.0]
```

We also need to configure the ansible.cfg file created by ansible to deploy our application on a remote server(our azure VM). This should be done as shown below:

The screenshot shows a Visual Studio Code interface. On the left, there's a terminal window titled 'pankaj@Coder: ~' displaying configuration settings for Ansible. On the right, there's a code editor window titled 'ansible.cfg - Visual Studio Code' showing the actual Ansible configuration file.

```
[colors]
#highlight = white
#verbose = blue
#warn = bright purple
#error = red
#debug = dark gray
#deprecate = purple
#skip = cyan
#unreachable = red
#ok = green
#changed = yellow
#diff_add = green
#diff_remove = red
#diff_lines = cyan

[diff]
# Always print diff when running ( same as always running with -D/--diff )
# always = no

# Set how many context lines to show in diff
# context = 3
pankaj@Coder:~$ code /etc/ansible/ansible.cfg
pankaj@Coder:~$ []

etc > ansible > ansible.cfg
1  # config file for ansible -- https://ansible.com/
2  #
3  #
4  # nearly all parameters can be overridden in ansible-
5  # or with command line flags. ansible will read ANSIBLE
6  # ansible.cfg in the current working directory, .ansibl
7  # the home directory or /etc/ansible/ansible.cfg, which
8  # finds first
9
10 [defaults]
11
12 # some basic default values...
13
14 #inventory      = /etc/ansible/hosts
15 #library        = /usr/share/my_modules/
16 #module_utils   = /usr/share/my_module_utils/
17 #remote_tmp     = ~/.ansible/tmp
18 #local_tmp      = ~/.ansible/tmp
19 #plugin_filters_cfg = /etc/ansible/plugin_filters.yml
20 #forks          = 5
21 #poll_interval  = 15
22 #sudo_user      = root
23 #ask_sudo_pass  = True
24 #ask_pass       = True
25 #transport      = smart
26 #remote_port    = 22
27 #module_lang    = C
28 #module_set_locale = False
29
```

Then in the ansible.cfg file uncomment the `#remote_user` line and assign remote user to azureuser. Also, uncomment the `#private_key` line and assign `private_key` to path of the .pem key file.

Activities ➔ Visual Studio Code ▾ April 17 4:23 PM ansible.cfg - Visual Studio Code

File Edit Selection View Go Run Terminal Help

✖ Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

ansible.cfg

```
etc > ansible > ansible.cfg
107 remote_user = azureuser
108
109 # logging is off by default unless this path is defined
110 # if so defined, consider logrotate
111 #log_path = /var/log/ansible.log
112
113 # default module name for /usr/bin/ansible
114 #module_name = command
115
116 # use this shell for commands executed under sudo
117 # you may need to change this to bin/bash in rare instances
118 # if sudo is constrained
119 #executable = /bin/sh
120
121 # if inventory variables overlap, does the higher precedence one win
122 # or are hash values merged together? The default is 'replace' but
123 # this can also be set to 'merge'.
124 #hash_behaviour = replace
125
126 # by default, variables from roles will be visible in the global variable
127 # scope. To prevent this, the following option can be enabled, and only
128 # tasks and handlers within the role will see the variables there
129 #private_role_vars = yes
130
131 # list any Jinja2 extensions to enable here:
132 #jinja2_extensions = jinja2.ext.do,jinja2.ext.i18n
133
134 # if set, always use this private key file for authentication, same as
135 # if passing --private-key to ansible or ansible-playbook
136 #private_key_file = /home/pankaj/spe_ansible.pem
137
138 # If set, configures the path to the Vault password file as an alternative to
139 # specifying --vault-password-file on the command line.
140 #vault_password_file = /path/to/vault_password_file
141
142 # format of string {{ ansible_managed }} available within Jinja2
143 # templates indicates to users editing templates files will be replaced.
144 # replacing {file}, {host} and {uid} and strftime codes with proper values.
145 #ansible_managed = Ansible managed: {file} modified on %Y-%m-%d %H:%M:%S by {uid} on {host}
146 # {file}, {host}, {uid}, and the timestamp can all interfere with idempotence
147 # in some situations so the default is a static string:
148 #ansible_managed = Ansible managed
149
150 # by default, ansible-playbook will display "Skipping [host]" if it determines a task
151 # should not be run on a host. Set this to "False" if you don't want to see these "Skipping"
152 # messages. NOTE: the task header will still be shown regardless of whether or not the
153 # task is skipped.
154 #display_skipped_hosts = True
155
156 # by default, if a task in a playbook does not include a name: field then
157 # ansible-playbook will construct a header that includes the task's action but
158 # not the task's args. This is a security feature because ansible cannot know
159 # if the *module* considers an argument to be no_log at the time that the
160 # header is printed. If your environment doesn't have a problem securing
161 # stdout from ansible-playbook (or you have manually specified no_log in your
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF Properties ⚙️

Activities ➔ Visual Studio Code ▾ April 17 4:23 PM ansible.cfg - Visual Studio Code

File Edit Selection View Go Run Terminal Help

✖ Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

ansible.cfg

```
etc > ansible > ansible.cfg
133
134 # if set, always use this private key file for authentication, same as
135 # if passing --private-key to ansible or ansible-playbook
136 #private_key_file = /home/pankaj/spe_ansible.pem
137
138 # If set, configures the path to the Vault password file as an alternative to
139 # specifying --vault-password-file on the command line.
140 #vault_password_file = /path/to/vault_password_file
141
142 # format of string {{ ansible_managed }} available within Jinja2
143 # templates indicates to users editing templates files will be replaced.
144 # replacing {file}, {host} and {uid} and strftime codes with proper values.
145 #ansible_managed = Ansible managed: {file} modified on %Y-%m-%d %H:%M:%S by {uid} on {host}
146 # {file}, {host}, {uid}, and the timestamp can all interfere with idempotence
147 # in some situations so the default is a static string:
148 #ansible_managed = Ansible managed
149
150 # by default, ansible-playbook will display "Skipping [host]" if it determines a task
151 # should not be run on a host. Set this to "False" if you don't want to see these "Skipping"
152 # messages. NOTE: the task header will still be shown regardless of whether or not the
153 # task is skipped.
154 #display_skipped_hosts = True
155
156 # by default, if a task in a playbook does not include a name: field then
157 # ansible-playbook will construct a header that includes the task's action but
158 # not the task's args. This is a security feature because ansible cannot know
159 # if the *module* considers an argument to be no_log at the time that the
160 # header is printed. If your environment doesn't have a problem securing
161 # stdout from ansible-playbook (or you have manually specified no_log in your
```

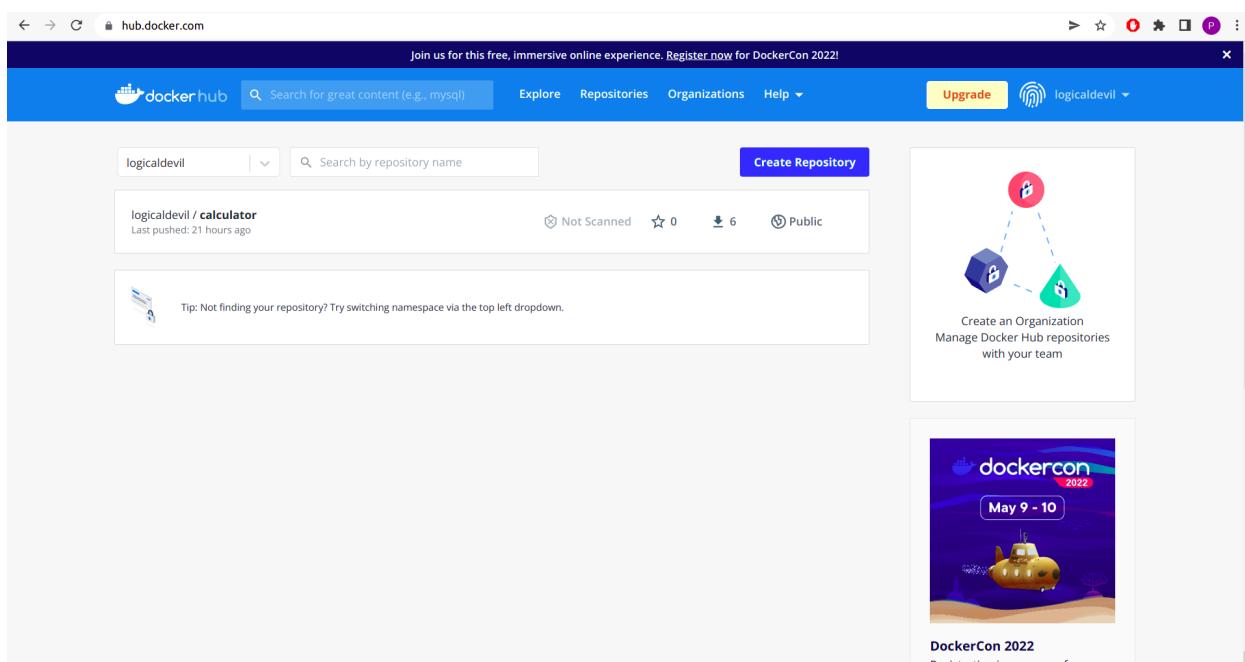
Ln 1, Col 1 Spaces: 4 UTF-8 LF Properties ⚙️

10) Docker Hub:

Docker Hub is a service provided by Docker for finding and sharing container images with your team.

Steps to use:

- Signup at <https://hub.docker.com> and create a repository

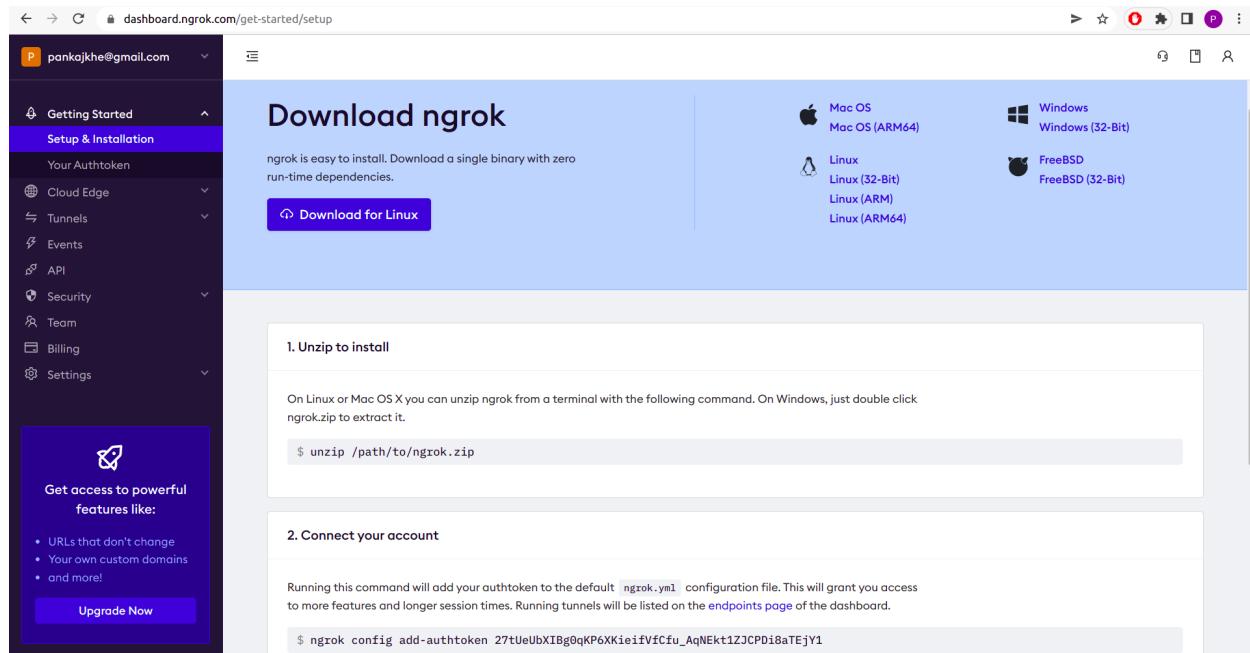


11) Ngrok and webhook:

Ngrok is a cross-platform application that enables developers to expose a local development server to the Internet with minimal effort. The software makes your locally-hosted web server appear to be hosted on a subdomain of ngrok.com, meaning that no public IP or domain name on the local machine is needed.

Steps to install and setup ngrok and webhook:

- Download Ngrok from this link :
<https://dashboard.ngrok.com/get-started/setup>
- Unzip the downloaded file
 - \$ tar -xvf ngrok-v3-stable-linux-amd64.tgz
- Connect your account
 - \$./ngrok config add-authtoken your_auth_token
- Start a HTTP tunnel forwarding to your local port 8080
 - \$ ngrok http 8080
- Now go to your Github repo, settings -> webhooks. Add the payload URL that is displayed on the ngrok terminal. Add /github-webhook/ at the end of the url.
- Also set the gitscm polling option as true on jenkins.



github.com/Prince-5/SPEMini/settings/hooks/353641452

Prince-5 / SPEMini Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Webhooks / Manage webhook

General Access Collaborators Moderation options

Code and automation Tags Branches Actions

Webhooks Environments Pages

Security Code security and analysis Deploy keys Secrets

Integrations GitHub apps Email notifications

Payload URL * https://5899-103-156-19-229.in.ngrok.io/github-webhook/

Content type application/x-www-form-urlencoded

Secret (empty)

SSL verification By default, we verify SSL certificates when delivering payloads.

Enable SSL verification Disable (not recommended)

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.

Update webhook **Delete webhook**

Activities Terminal ▾ Apr 17 3:39 PM

localhost:8080/job/SPEMiniPipeline/

Jenkins

Dashboard > SPEMiniPipeline >

Pipeline SPEMiniPipeline

Status

Recent Changes

Stage View

Average stage times: (Average full run time: ~47s)

#28	Apr 17 15:37	1 commit	829ms
#27	Apr 17 15:32	5 commits	961ms
#26	Apr 17 14:53	No Changes	1s
			798ms

Add description **Disable Project**

ngrok

```
pankaj@Coder:~/ngrok-v3-stable-linux-amd64$ (Ctrl+C to quit)
Session Status          online
Account                  pankajkhe@gmail.com (Plan: Free)
Version                 3.0.2
Region                  India (in)
Latency                42.993457ms
Web Interface           http://127.0.0.1:4040
Forwarding              https://5899-103-156-19-229.in.ngrok.io -> http://
Connections
  ttl     opn      rti      rt5      p50      p90
    1       0      0.00     0.00    5.33    5.33
HTTP Requests
POST /github-webhook/    200 OK
```

12) Docker-ELK(Logstash, kibana):

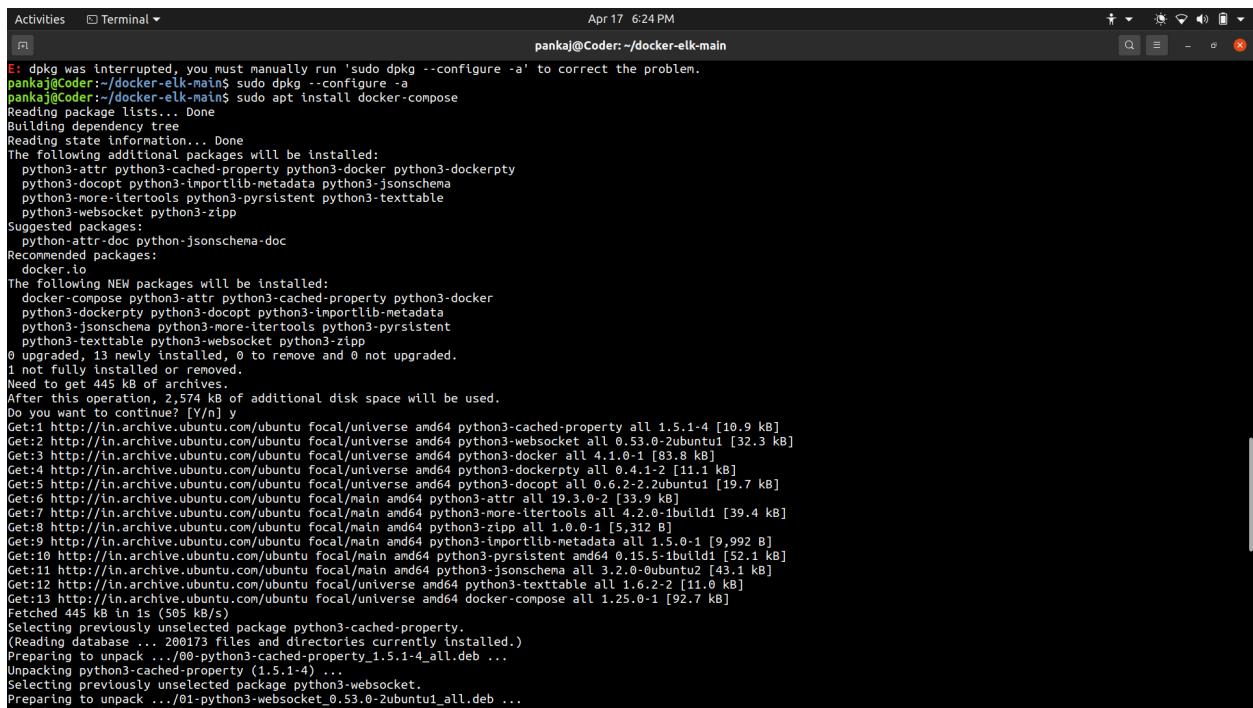
We use these tools for monitoring the logs generated by the calculator application.

We need to clone the repo

<https://github.com/deviantony/docker-elk.git>

Then install docker-compose,

\$ sudo apt install docker-compose



The screenshot shows a terminal window titled "Terminal" with the command history and output of the apt-get installation process. The terminal shows the user running "sudo apt install docker-compose" and the subsequent package download and installation steps. The terminal window has a dark theme with white text and a black background.

```
Activities Terminal ▾
E: dpkg was interrupted, you must manually run 'sudo dpkg --configure -a' to correct the problem.
pankaj@Coder:~/docker-elk-main$ sudo dpkg --configure -a
pankaj@Coder:~/docker-elk-main$ sudo apt install docker-compose
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
python3-attr python3-cached-property python3-docker python3-dockerpty
python3-docopt python3-importlib-metadata python3-jsonschema
python3-more-itertools python3-persistent python3-texttable
python3-websocket python3-zipp
Suggested packages:
python-attr-doc python-jsonschema-doc
Recommended packages:
docker.io
The following NEW packages will be installed:
docker-compose python3-attr python3-cached-property python3-docker
python3-dockerpty python3-docopt python3-importlib-metadata
python3-jsonschema python3-more-itertools python3-persistent
python3-texttable python3-websocket python3-zipp
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
1 not fully installed or removed.
Need to get 445 kB of archives.
After this operation, 2,574 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-cached-property all 1.5.1-4 [10.9 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-websocket all 0.53.0-2ubuntu1 [32.3 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-docker all 4.1.0-1 [83.8 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-dockerpty all 0.4.1-2 [11.1 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-docopt all 0.6.2-2.2ubuntu1 [19.7 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal/main amd64 python3-attr all 19.3.0-2 [33.9 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal/main amd64 python3-more-itertools all 4.2.0-1ubuntu1 [39.4 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu focal/main amd64 python3-zipp all 1.0.0-1 [5,312 B]
Get:9 http://in.archive.ubuntu.com/ubuntu focal/main amd64 python3-importlib-metadata all 1.5.0-1 [9,992 B]
Get:10 http://in.archive.ubuntu.com/ubuntu focal/main amd64 python3-persistent amd64 0.15.5-1ubuntu1 [52.1 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-jsonschema all 3.2.0-0ubuntu2 [43.1 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 python3-texttable all 1.6.2-2 [11.0 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu focal/universe amd64 docker-compose all 1.25.0-1 [92.7 kB]
Fetched 445 kB in 1s (505 kB/s)
Selecting previously unselected package python3-cached-property.
(Reading database ... 200173 files and directories currently installed.)
Preparing to unpack .../00-python3-cached-property_1.5.1-4_all.deb ...
Unpacking python3-cached-property (1.5.1-4) ...
Selecting previously unselected package python3-websocket.
Preparing to unpack .../01-python3-websocket_0.53.0-2ubuntu1_all.deb ...
```

Then change the working directory to docker-elk-main and use the command \$docker-compose up -d to set all the logstash, elasticsearch and kibana servers. Now, we can use all of them for monitoring.

Part 2 - Developing the Project:

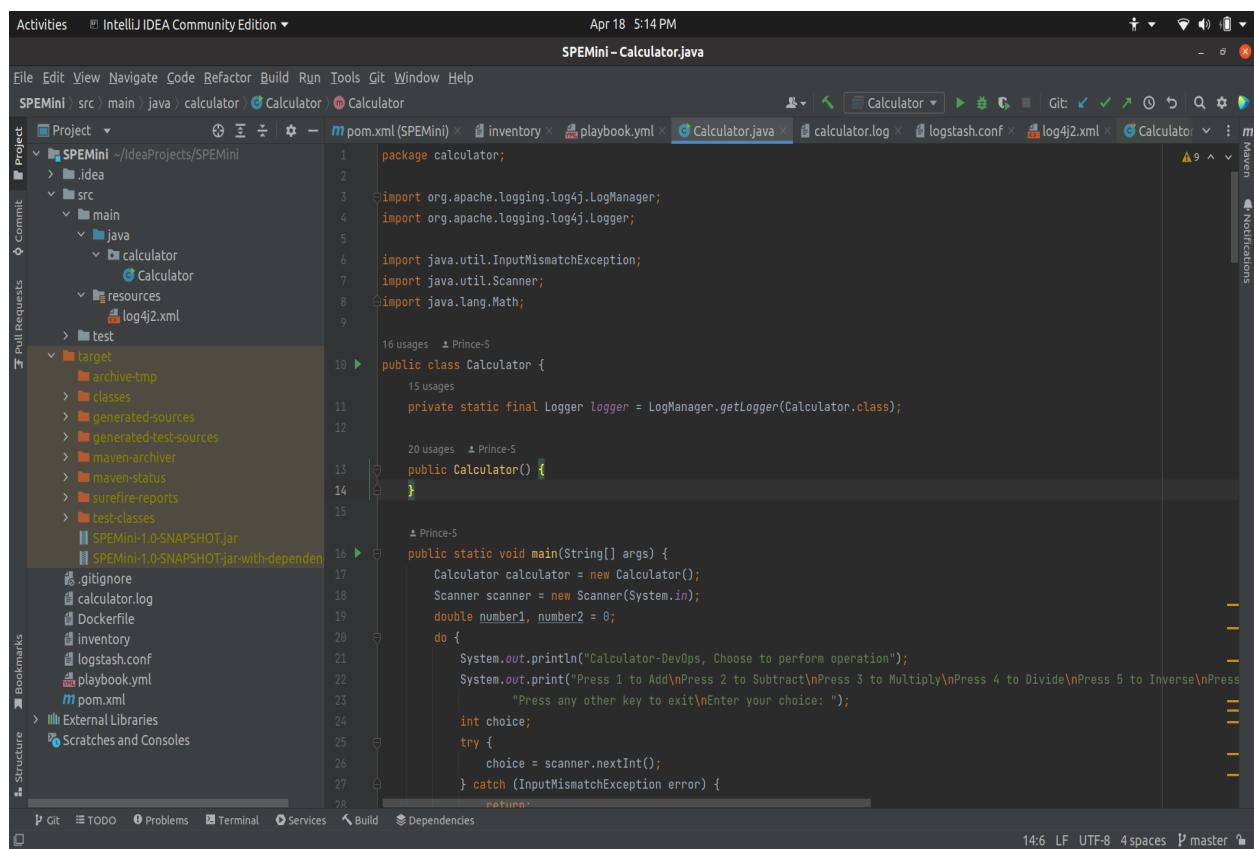
1) Write the code for calculator:

Set up a new project in IntelliJ

- Open IntelliJ and select a new project.
- Select maven from the menu.
- Give a name and click Finish

Go to src/main. Right click on java and select new/package.

Write the calculator code here.



The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Activities IntelliJ IDEA Community Edition, April 18 5:14 PM, SPEMIni-Calculator.java
- File Menu:** File Edit View Navigate Code Refactor Build Run Tools Git Window Help
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Git.
- Project Tool Window:** Shows the project structure under SPEMIni. The target folder is selected. Other visible files include pom.xml, inventory, playbook.yml, calculator.log, logstash.conf, log4j2.xml, and various configuration files.
- Code Editor:** Displays the Calculator.java code. The code is as follows:

```
package calculator;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.util.InputMismatchException;
import java.util.Scanner;
import java.lang.Math;

public class Calculator {

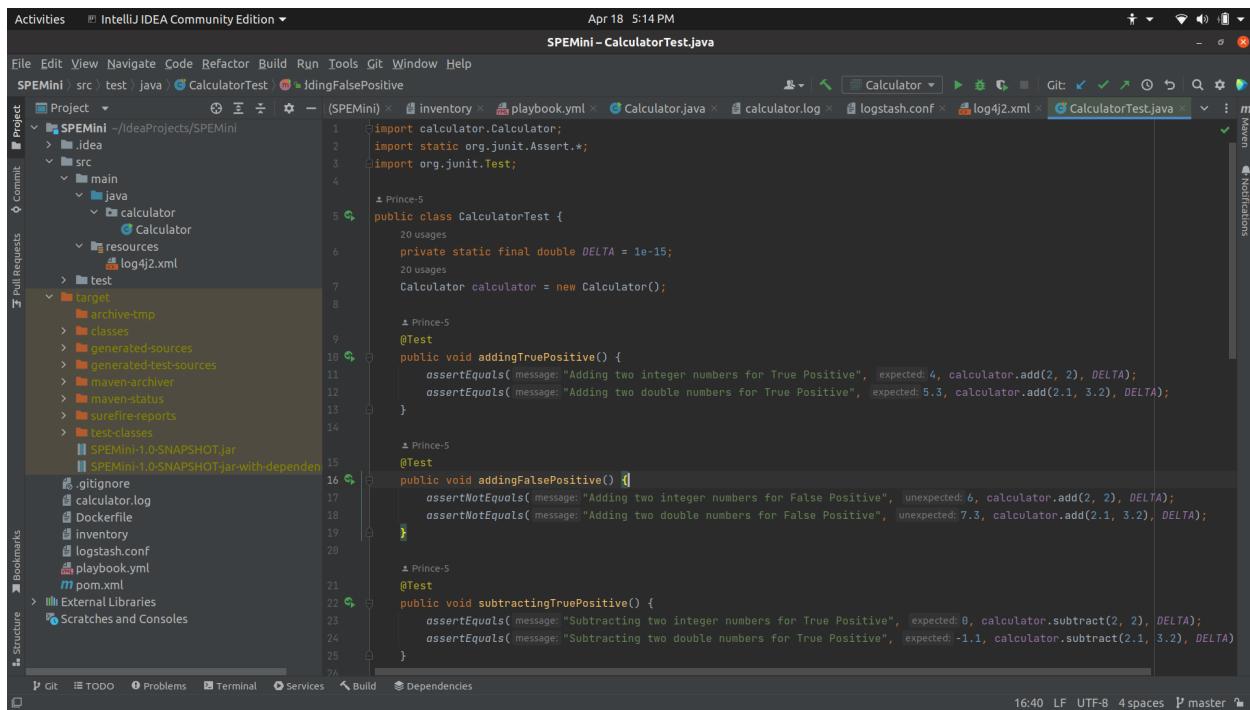
    private static final Logger logger = LogManager.getLogger(Calculator.class);

    public Calculator() {
    }

    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        Scanner scanner = new Scanner(System.in);
        double number1, number2 = 0;
        do {
            System.out.println("Calculator-DevOps, Choose to perform operation");
            System.out.print("Press 1 to Add\nPress 2 to Subtract\nPress 3 to Multiply\nPress 4 to Divide\nPress 5 to Inverse\nPress any other key to exit\nEnter your choice: ");
            int choice;
            try {
                choice = scanner.nextInt();
            } catch (InputMismatchException error) {
                return;
            }
            if (choice == 1) {
                number1 = scanner.nextDouble();
                number2 = scanner.nextDouble();
                System.out.println("Sum is " + (number1 + number2));
            } else if (choice == 2) {
                number1 = scanner.nextDouble();
                number2 = scanner.nextDouble();
                System.out.println("Difference is " + (number1 - number2));
            } else if (choice == 3) {
                number1 = scanner.nextDouble();
                number2 = scanner.nextDouble();
                System.out.println("Product is " + (number1 * number2));
            } else if (choice == 4) {
                number1 = scanner.nextDouble();
                number2 = scanner.nextDouble();
                System.out.println("Quotient is " + (number1 / number2));
            } else if (choice == 5) {
                number1 = scanner.nextDouble();
                System.out.println("Inverse is " + (1 / number1));
            }
        } while (choice != 6);
    }
}
```

The code implements a simple calculator with addition, subtraction, multiplication, division, and inverse operations. It uses Java's Scanner and Math classes, and it logs to the console.

Go to src/java and create a new java file to write all the tests.



The screenshot shows the IntelliJ IDEA interface with the project 'SPEMIni' open. The 'CalculatorTest.java' file is the active editor. The code contains several JUnit test cases for a 'Calculator' class, including methods for addition and subtraction. The IDE's navigation bar at the top includes tabs for 'CalculatorTest.java', 'calculator.log', 'logstash.conf', 'log4j2.xml', and 'Calculator.java'. The left sidebar shows the project structure with 'src/main/java/calculator' selected. The bottom status bar indicates the file is 16:40 LF UTF-8 4 spaces master.

```
import calculator.Calculator;
import static org.junit.Assert.*;
import org.junit.Test;

public class CalculatorTest {
    private static final double DELTA = 1e-15;

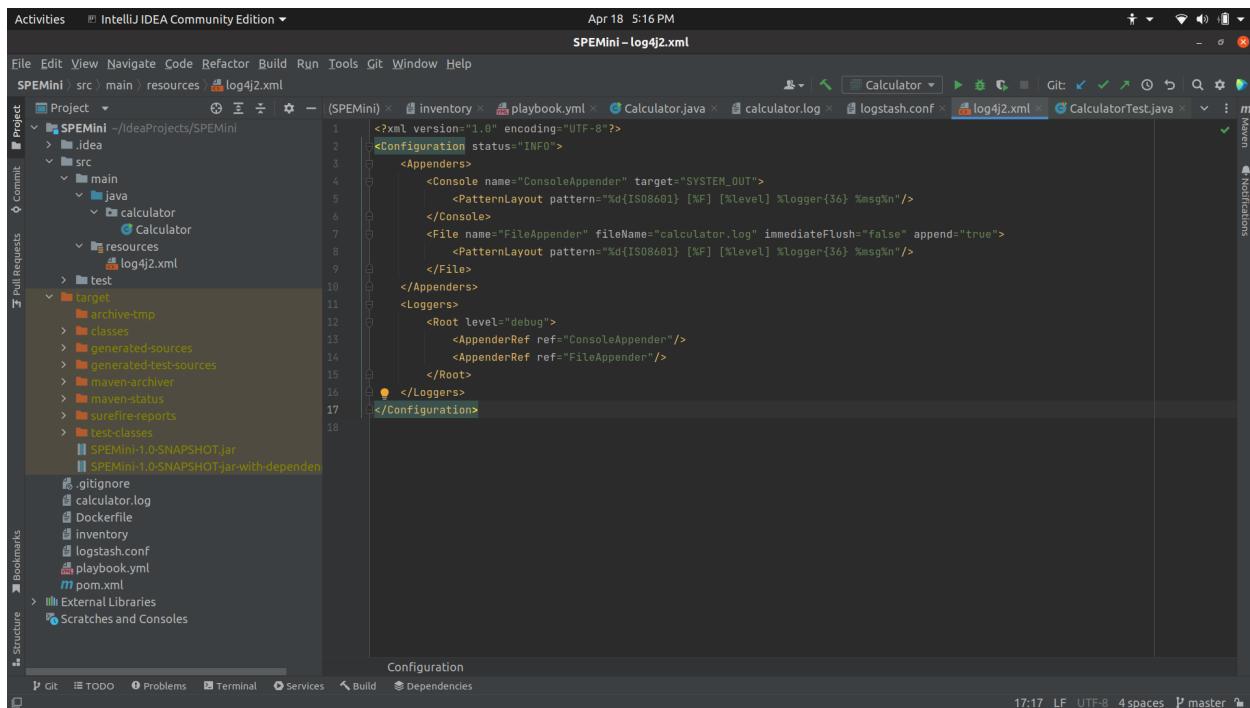
    Calculator calculator = new Calculator();

    @Test
    public void addingTruePositive() {
        assertEquals("Adding two integer numbers for True Positive", expected: 4, calculator.add(2, 2), DELTA);
        assertEquals("Adding two double numbers for True Positive", expected: 5.3, calculator.add(2.1, 3.2), DELTA);
    }

    @Test
    public void addingFalsePositive() {
        assertNotEquals("Adding two integer numbers for False Positive", unexpected: 6, calculator.add(2, 2), DELTA);
        assertNotEquals("Adding two double numbers for False Positive", unexpected: 7.3, calculator.add(2.1, 3.2), DELTA);
    }

    @Test
    public void subtractingTruePositive() {
        assertEquals("Subtracting two integer numbers for True Positive", expected: 0, calculator.subtract(2, 2), DELTA);
        assertEquals("Subtracting two double numbers for True Positive", expected: -1.1, calculator.subtract(2.1, 3.2), DELTA);
    }
}
```

Go to src/main/resources, and create a log4j2.xml file and specify the format for the log file.

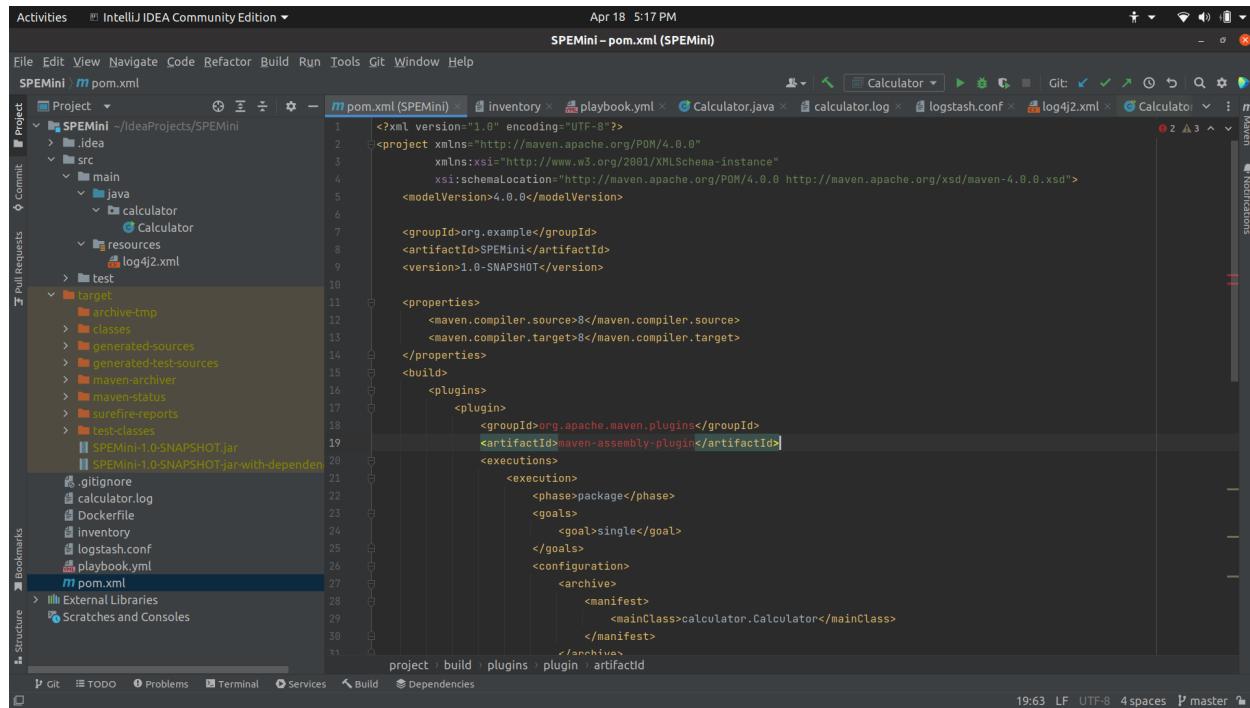


The screenshot shows the IntelliJ IDEA interface with the 'log4j2.xml' file open. The code defines a log4j2 configuration with two appenders: 'ConsoleAppender' and 'FileAppender'. The 'ConsoleAppender' outputs to System.out with a specific pattern. The 'FileAppender' outputs to a file named 'calculator.log' with immediate flush and a different pattern. The IDE's navigation bar at the top includes tabs for 'calculator.log', 'logstash.conf', 'log4j2.xml', and 'CalculatorTest.java'. The left sidebar shows the project structure with 'src/main/resources' selected. The bottom status bar indicates the file is 17:17 LF UTF-8 4 spaces master.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{IS08601} [%F] [%level] %logger{36} %msg%n"/>
        </Console>
        <File name="FileAppender" fileName="calculator.log" immediateFlush="false" append="true">
            <PatternLayout pattern="%d{IS08601} [%F] [%level] %logger{36} %msg%n"/>
        </File>
    </Appenders>
    <Loggers>
        <Root level="debug">
            <AppenderRef ref="ConsoleAppender"/>
            <AppenderRef ref="FileAppender"/>
        </Root>
        </Loggers>
    </Configuration>
```

Now create the following files in the project folder: pom.xml, dockerfile, inventory, playbook.yml

Pom.xml: This contains all the dependencies for the code. It is used by maven to build the code.



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>SPEMini</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-assembly-plugin</artifactId>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                        <configuration>
                            <archive>
                                <manifest>
                                    <mainClass>calculator.Calculator</mainClass>
                                </manifest>
                            </archive>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Dockerfile: Docker uses this to construct the image. We're using an openjdk base image here, and after it's up and running, it'll copy and run the calculator jar file.

Activities IntelliJ IDEA Community Edition ▾ Apr 18 5:18 PM SPEMini – Dockerfile

File Edit View Navigate Code Refactor Build Run Tools Git Window Help

SPEMini ▾ Dockerfile

Project SPEMini ~/ideaProjects/SPEMini

- src
 - main
 - java
 - calculator
 - Calculator
 - resources
 - log4j2.xml
 - test
 - target
 - archive-tmp
 - classes
 - generated-sources
 - generated-test-sources
 - maven-archiver
 - maven-status
 - surefire-reports
 - test-classes
 - SPEMini-1.0-SNAPSHOT.jar
 - SPEMini-1.0-SNAPSHOT-jar-with-dependencies.jar
 - .gitignore
 - calculator.log
 - Dockerfile
 - inventory
 - logstash.conf
 - playbook.yml
- pom.xml

Bookmarks External Libraries Scratches and Consoles

Git TODO Problems Terminal Services Build Dependencies

3:34 LF UTF-8 4 spaces master

```
FROM openjdk:8
MAINTAINER Pankaj Khemani pankajkhe@gmail.com
COPY ./target/SPEMini-1.0-SNAPSHOT-jar-with-dependencies.jar .
WORKDIR .
CMD ["java", "-jar", "SPEMini-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

Inventory: It specifies the username and ip address on which ansible needs to work on.

Activities IntelliJ IDEA Community Edition ▾ Apr 18 5:22 PM SPEMini – inventory

File Edit View Navigate Code Refactor Build Run Tools Git Window Help

SPEMini ▾ inventory

Project SPEMini ~/ideaProjects/SPEMini

- src
 - main
 - java
 - calculator
 - Calculator
 - resources
 - log4j2.xml
 - test
 - target
 - archive-tmp
 - classes
 - generated-sources
 - generated-test-sources
 - maven-archiver
 - maven-status
 - surefire-reports
 - test-classes
 - SPEMini-1.0-SNAPSHOT.jar
 - SPEMini-1.0-SNAPSHOT-jar-with-dependencies.jar
 - .gitignore
 - calculator.log
 - Dockerfile
 - inventory
 - logstash.conf
 - playbook.yml
- pom.xml

Bookmarks External Libraries Scratches and Consoles

Git TODO Problems Terminal Services Build Dependencies

1:28 LF UTF-8 4 spaces master

```
28.48.48.32 ansible_user=azureuser
```

Playbook.yml: It contains the commands to be executed on the remote machine.

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Menu:** Activities, IntelliJ IDEA Community Edition, File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Title Bar:** SPEMini – playbook.yml, Date: Apr 18 5:23 PM.
- Toolbars:** Standard toolbar with icons for Open, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, etc.
- Project Tree:** Shows the project structure under SPEMini, including src/main/java/calculator/Calculator.java, src/main/resources/log4j2.xml, src/test/java, target, and various build artifacts like SPEMini-1.0-SNAPSHOT.jar.
- Code Editor:** The current file is playbook.yml, which contains the following YAML code:

```
name: Pull and Run docker image
hosts: all
tasks:
  - name: Pull logicaldevil/calculator
    docker_image:
      name: logicaldevil/calculator:latest
    source: pull
```

The code editor has syntax highlighting for YAML. The status bar at the bottom shows the document is 1/1, item 1/1, tasks: 1/1, docker_image: 1/1, source: 1/1, and the time is 7:21.

- All the java files can be run separately and checked for faults.
- Click build and run for testing the whole code.
- After this use git add, commit and push and push the code to the github repo.

2) Setting up Jenkins PipeLine:

- Login to jenkins on <http://localhost:8080>

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Open Blue Ocean', 'Lockable Resources', and 'New View'. The main area has a table with columns: S (Status), W (Workflow), Name, Last Success, Last Failure, Last Duration, and Fav. A single row is shown for 'SPEMiniPipeline', which is green (Success) and has a duration of 46 sec. Below the table, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 idle). At the bottom right, there are links for 'REST API' and 'Jenkins 2.33.2'.

- Go to Dashboard/manage Jenkins/manage plugins/available.

The screenshot shows the Jenkins Plugin Manager. The top navigation bar includes 'Dashboard', 'Plugin Manager', and 'Available'. The main area has tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. Under the 'Available' tab, the 'Credentials' plugin is selected. It shows the version '1118.v320cd028cb_a_0' is 'Unavailable' due to Jenkins requirements. The 'JUnit' plugin is also listed with version '1.59' and a note about publishing test results. At the bottom, there are buttons for 'Download now and install after restart' and 'Check now'.

- Install the following plugins
 - Docker
 - Docker plugin

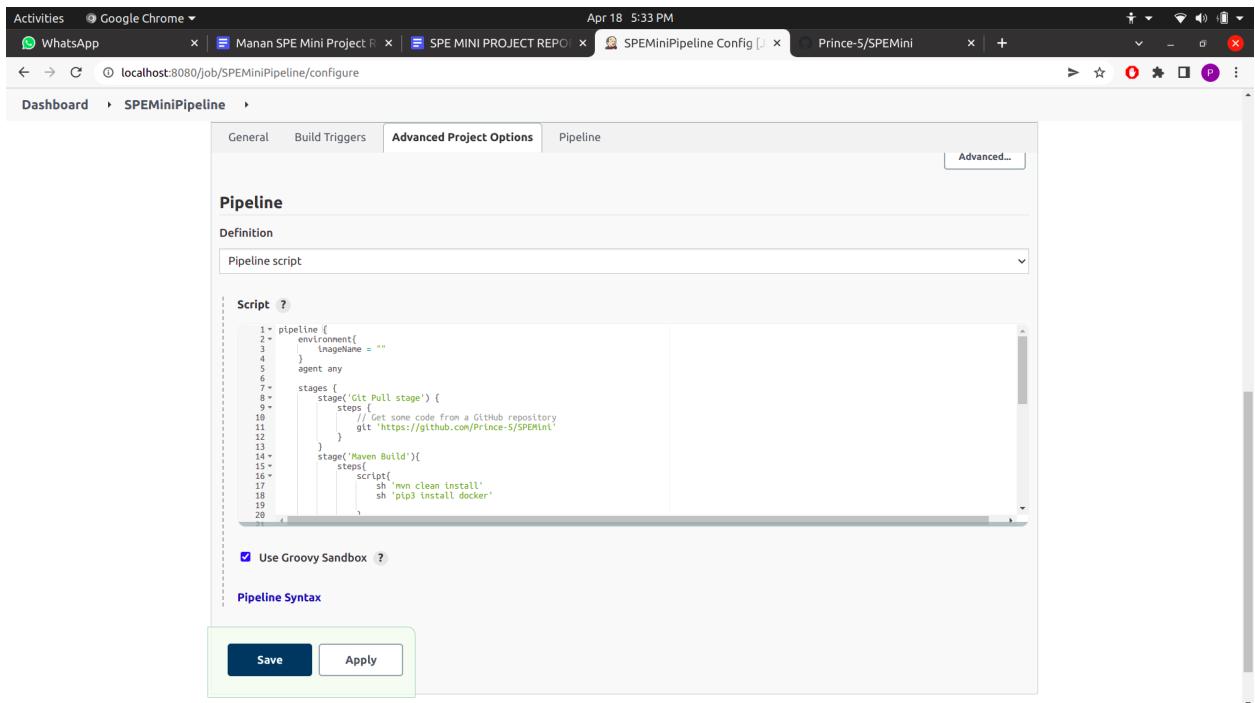
- Git
- Github Integration Plugin
- Junit plugin
- Maven integration
- Pipeline
- Ansible plugin
- Click on install without restart after selecting plugins.
- Go to manage jenkins/manage credentials/ jenkins/ Global Credentials/

The screenshot shows a Google Chrome browser window with multiple tabs open. The active tab is 'System » Global credentials' on the Jenkins server at 'localhost:8080'. The page title is 'Global credentials (unrestricted)'. It displays a table with one row of data:

ID	Name	Kind	Description
SPEMini	logicaldevil/***** (DockerHub Credentials)	Username with password	DockerHub Credentials

At the bottom right of the page, it says 'REST API Jenkins 2.332.2'.

- Click on Add Credentials.
- Enter Docker Hub and Github username, password, and give an ID to these credentials.
- Go to Dashboard/manage Jenkins/Configure system/Git
 - Add the name and path to the git executable.
- Go to Dashboard/manage Jenkins/Configure system/Ansible
 - Add the name and path to the git executable.
- Click on new item -> pipeline
- Create pipeline script:



The screenshot shows a Jenkins pipeline configuration page. At the top, there are tabs for General, Build Triggers, Advanced Project Options (which is selected), and Pipeline. Below the tabs, the Pipeline section is titled "Pipeline". Under "Definition", a dropdown menu is set to "Pipeline script". A large text area contains the following Groovy script:

```
1 pipeline {
2     environment{
3         imageName = ""
4     }
5     agent any
6
7     stages {
8         stage('Git Pull stage') {
9             steps {
10                 // Get some code from a GitHub repository
11                 git "https://github.com/Prince-5/SPEMin"
12             }
13         }
14         stage('Maven Build'){
15             steps{
16                 script{
17                     sh 'mvn clean install'
18                     sh 'pip3 install docker'
19                 }
20             }
21         }
22     }
23 }
```

A checkbox labeled "Use Groovy Sandbox" is checked. At the bottom, there are "Save" and "Apply" buttons.

- Update the values in the pipeline and save.

3) Last Steps:

- Now everything is setup-including the jenkins pipeline. The build now option starts the pipeline. The green boxes indicate that the stage is completed successfully.

- Now when the execution of the pipeline gets completed, we can connect to the azureuser(VM) and check if our calculator application is running there.
- I have already covered how to run the calculator application on VM in Virtual Machine on Azure part above. The images are also attached there.
- After we configure the ngrok as I had mentioned above in the ngrok setup, we don't even need to click on build now. Everytime something is pushed into the github repo the pipeline starts running and the code is built again.

4) ELK Stack:

The setup is already mentioned above.
We just need to upload our log file (i.e calculator.log for us) to the kibana server(localhost:5601) that generates the monitoring visualizations for us.

```

1 2022-04-18T02:46:56,574 [Calculator.java] [INFO] calculator.Calculator [ADDITION] - 2.0, 3.0
2 2022-04-18T02:46:56,578 [Calculator.java] [INFO] calculator.Calculator [RESULT - ADDITION] - 5.0
3 2022-04-18T02:47:04,918 [Calculator.java] [INFO] calculator.Calculator [DIVISION] - 3.0, 0.0
4 2022-04-18T02:47:04,911 [Calculator.java] [ERROR] calculator.Calculator [EXCEPTION - DIVISION] - Cannot be divided by ZERO Case of PC
5 2022-04-18T02:47:04,911 [Calculator.java] [INFO] calculator.Calculator [RESULT - DIVISION] - Infinity
6 2022-04-18T02:47:18,947 [Calculator.java] [INFO] calculator.Calculator [RESULT - POWER] - 512.0
7 2022-04-18T02:47:23,602 [Calculator.java] [INFO] calculator.Calculator [DIVISION] - 1.0, 0.0
8 2022-04-18T02:47:23,603 [Calculator.java] [ERROR] calculator.Calculator [EXCEPTION - DIVISION] - Cannot be divided by ZERO Case of PC
9 2022-04-18T02:47:23,603 [Calculator.java] [INFO] calculator.Calculator [RESULT - DIVISION] - Infinity

```

After uploading this file, select or type the appropriate grok pattern for the semi structured text. This can be found out using the grok debugger in the dev tools section for kibana.

Sample Data

```
1 2022-04-18T02:42:03,615 [Calculator.java] [INFO] calculator.Calculator [RESULT - ADDITION] - 5.0
```

Grok Pattern

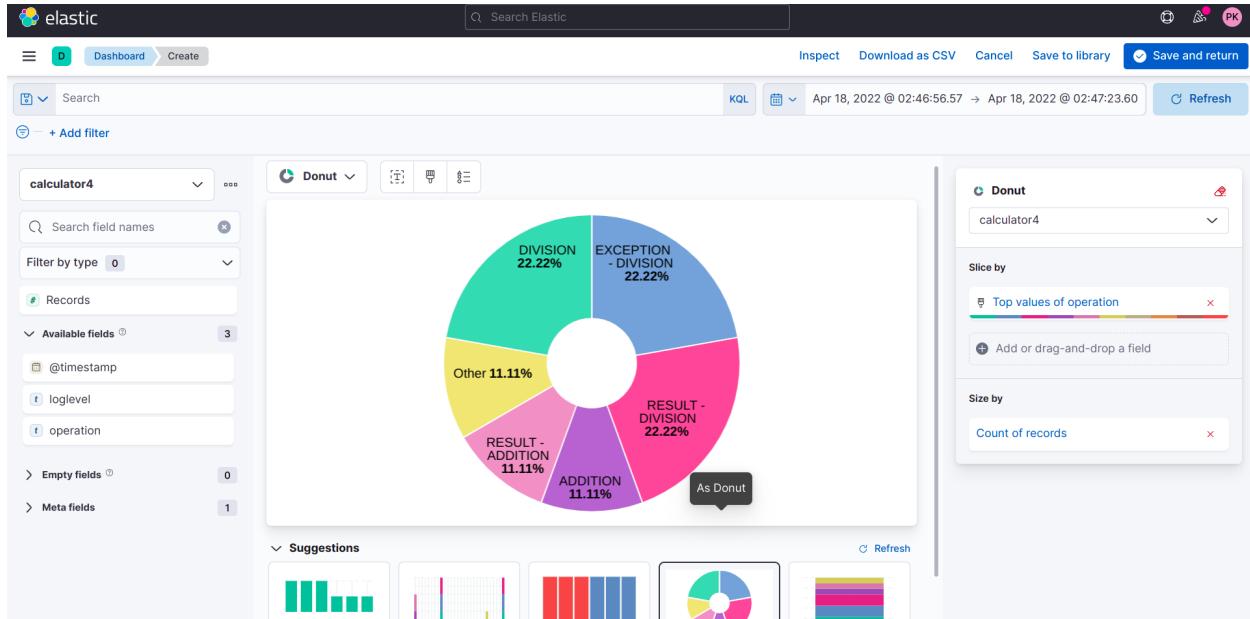
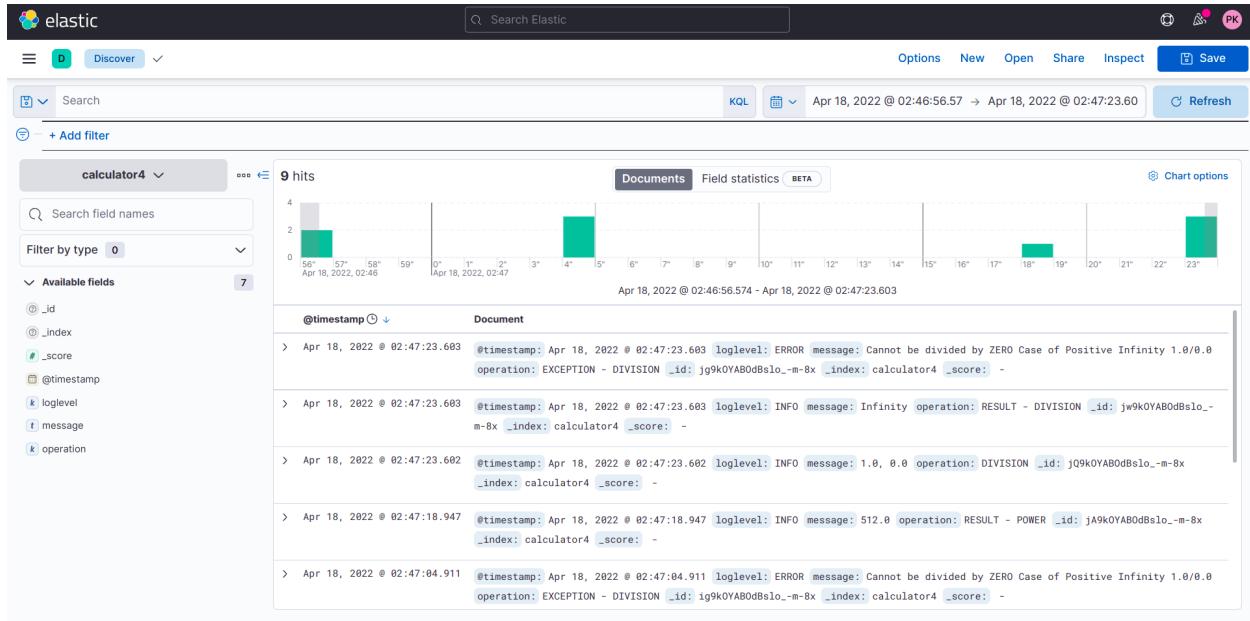
```
1 %{TIMESTAMP_ISO8601:timestamp} \[%{LOGLEVEL:loglevel}\] .%* \[%{GREEDYDATA:operation}\] - %{GREEDYDATA:message}
```

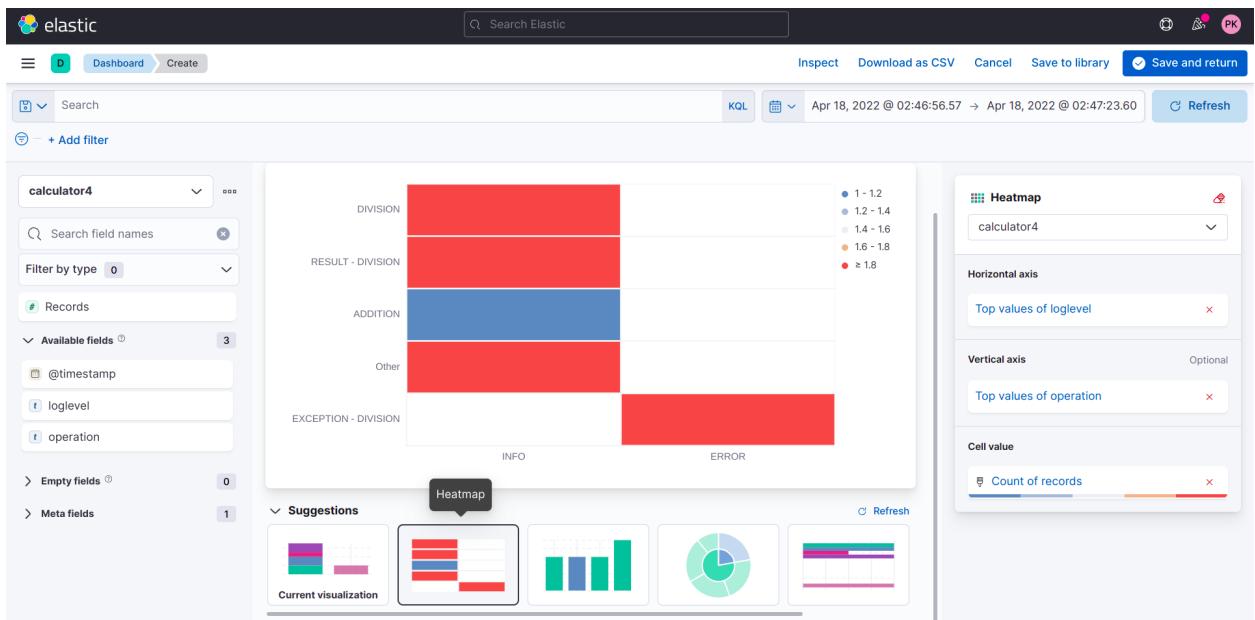
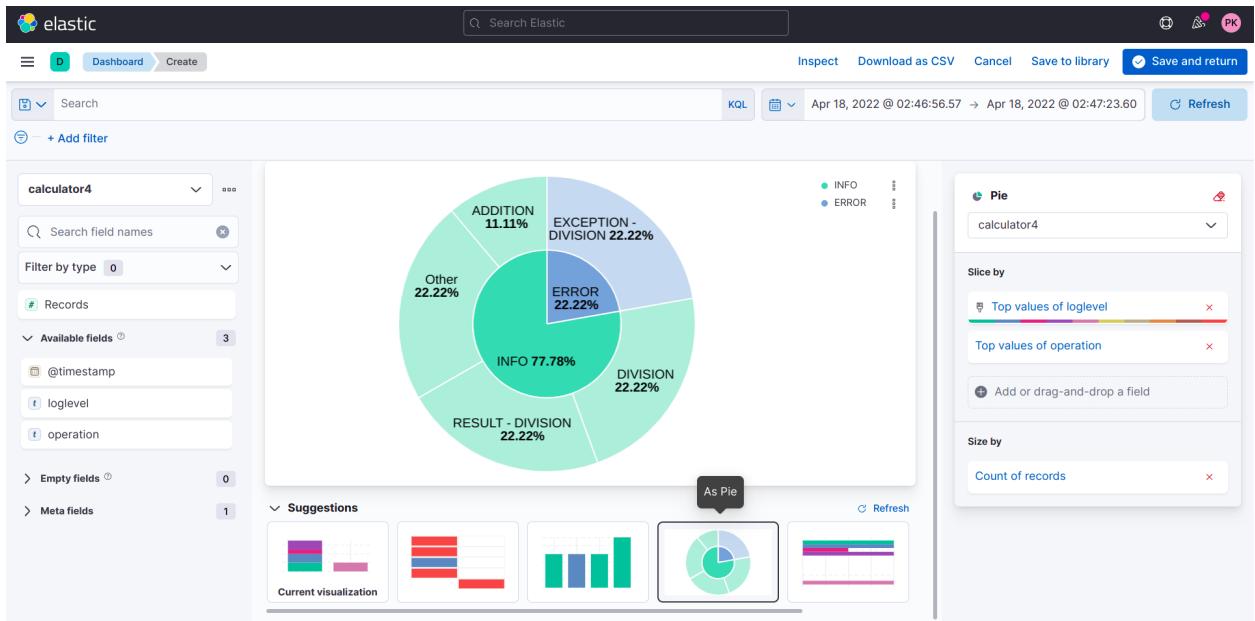
Structured Data

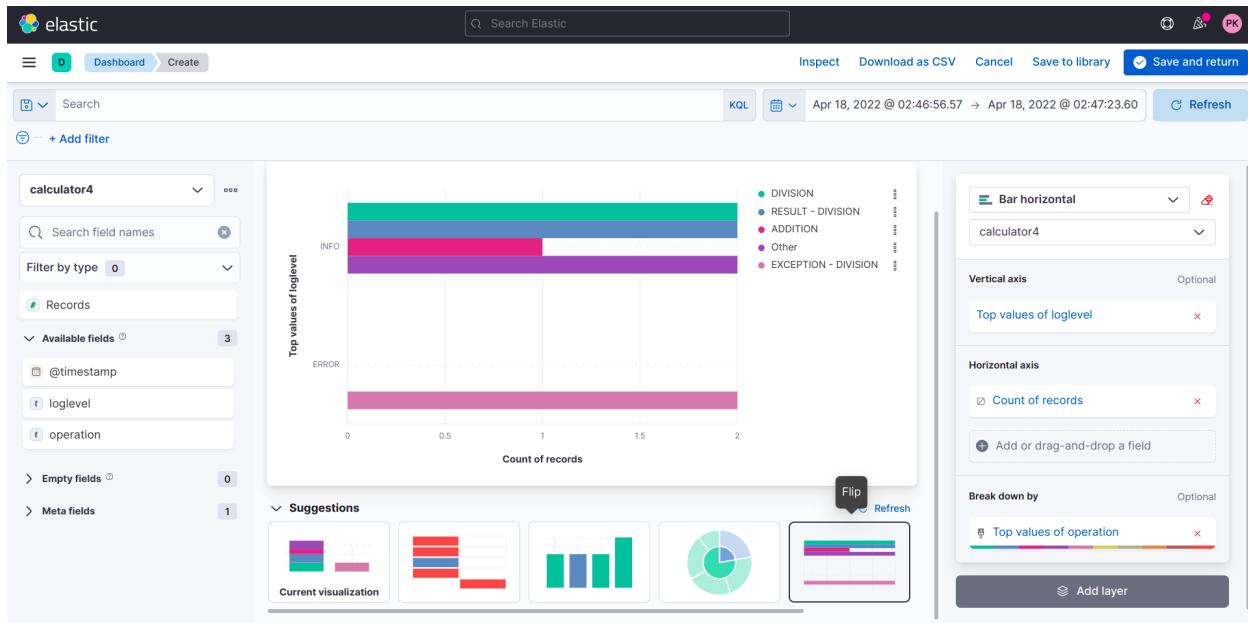
```
1 + {
2   "loglevel": "INFO",
3   "message": "5.0",
4   "operation": "RESULT - ADDITION",
5   "timestamp": "2022-04-18T02:42:03,615"
6 }
```

After this we can just use the UI for the monitoring visualizations and can generate many types of visualizations with different fields.

Below are the screenshots of visualizations for my calculator.log file:







My Github Repo: <https://github.com/Prince-5/SPEMini>

My DockerHub Repo:

<https://hub.docker.com/repository/docker/logicaldevil/calculator>

