# 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 - √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

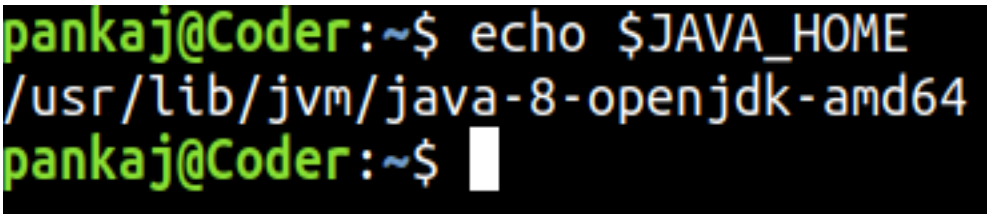
# Part 1 - Setup for Environments and Tools:

## 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 $ JAVE_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



## 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:

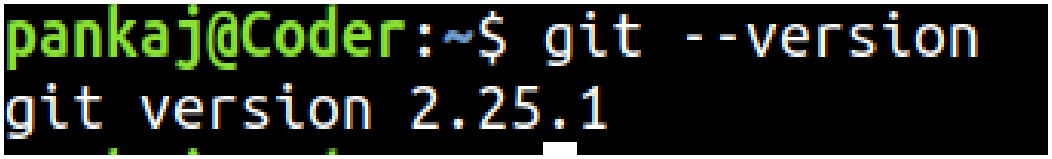
Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to track changes in the source code, enabling multiple developers to work together on non-linear development.

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

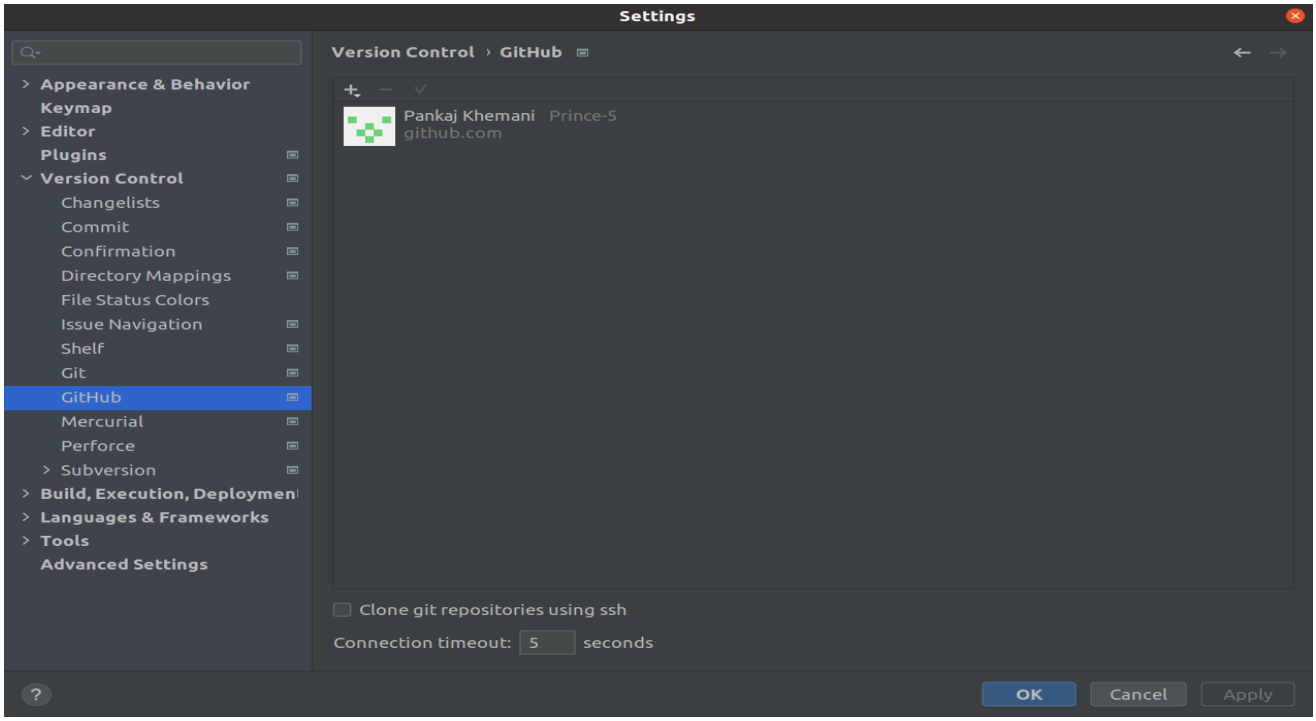


# 4) Github:

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:**

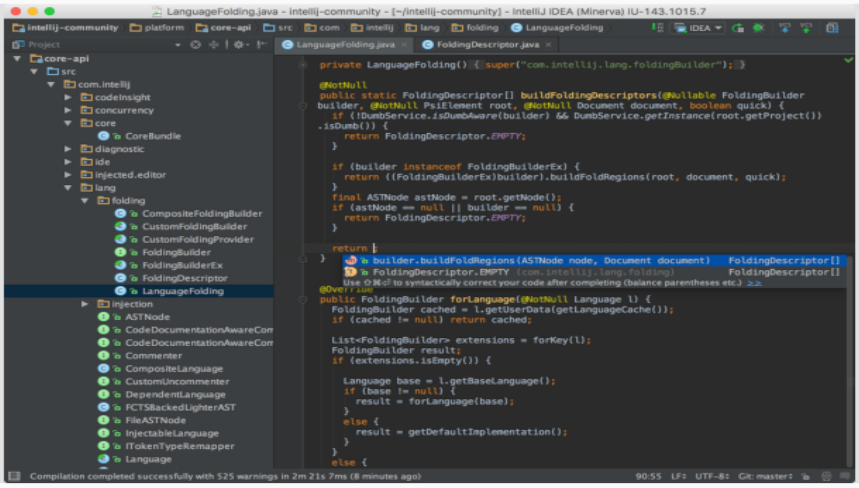
- Download Intellij IDEA from https://www.jetbrains.com/idea/download or directly download it from the ubuntu software store.
- After downloading from the ubuntu store it automatically gets installed.
- Open IntelliJ and go to settings->build, executions, deployment->build tools->maven
- Select use plugin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

# IDEA Community

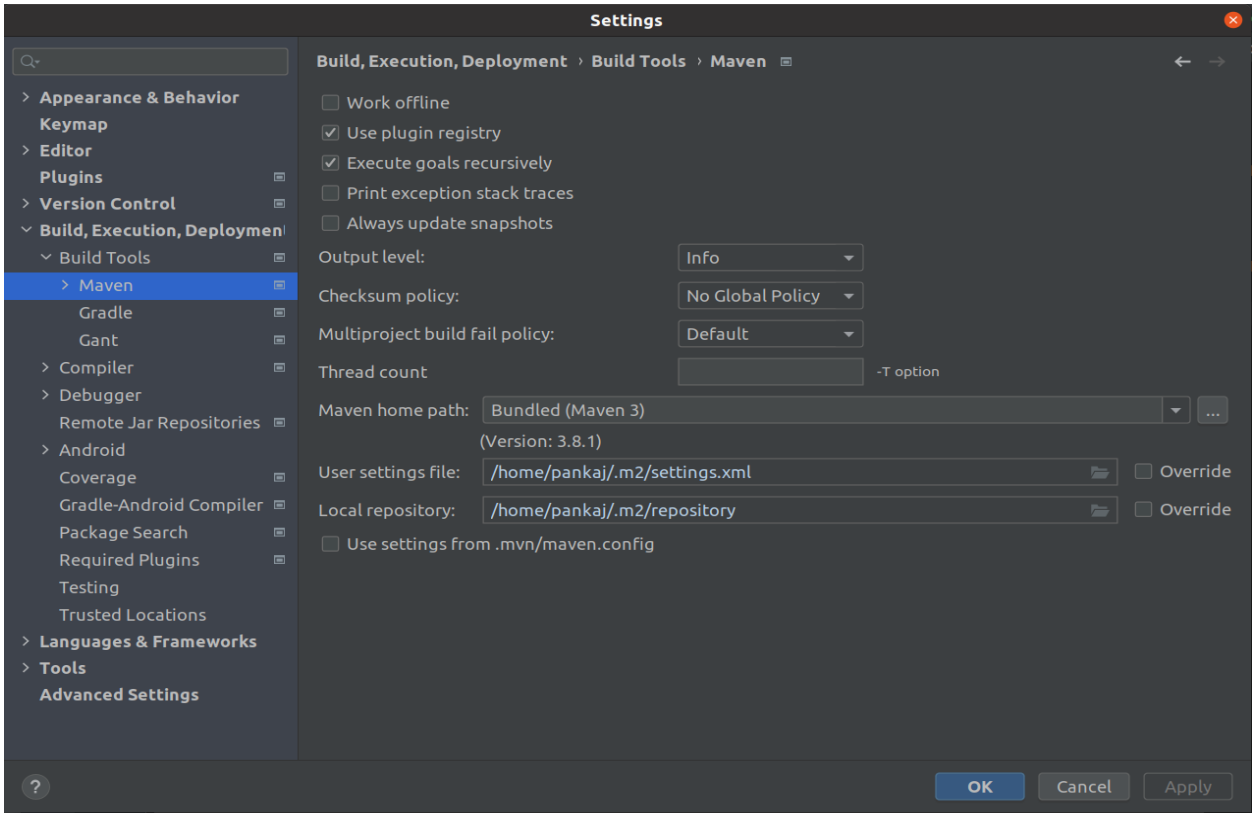
Capable & Ergonomic Java IDE

★★★★☆ (242)

Remove



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.

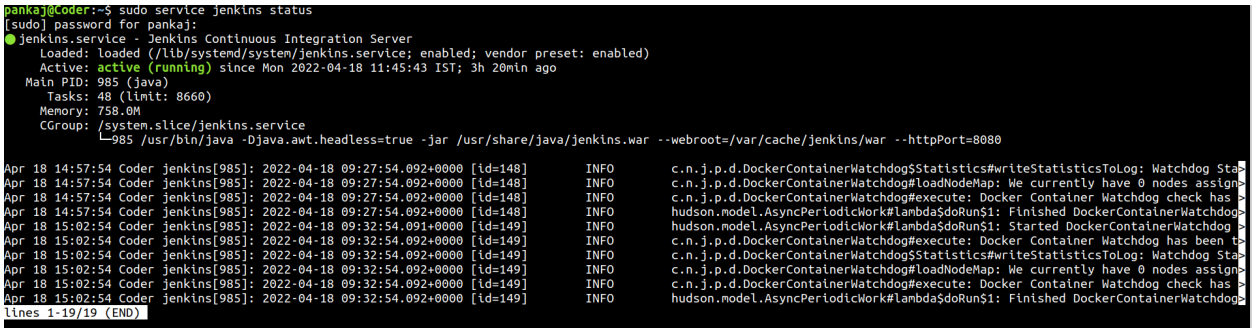


# 6) Jenkins:

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.

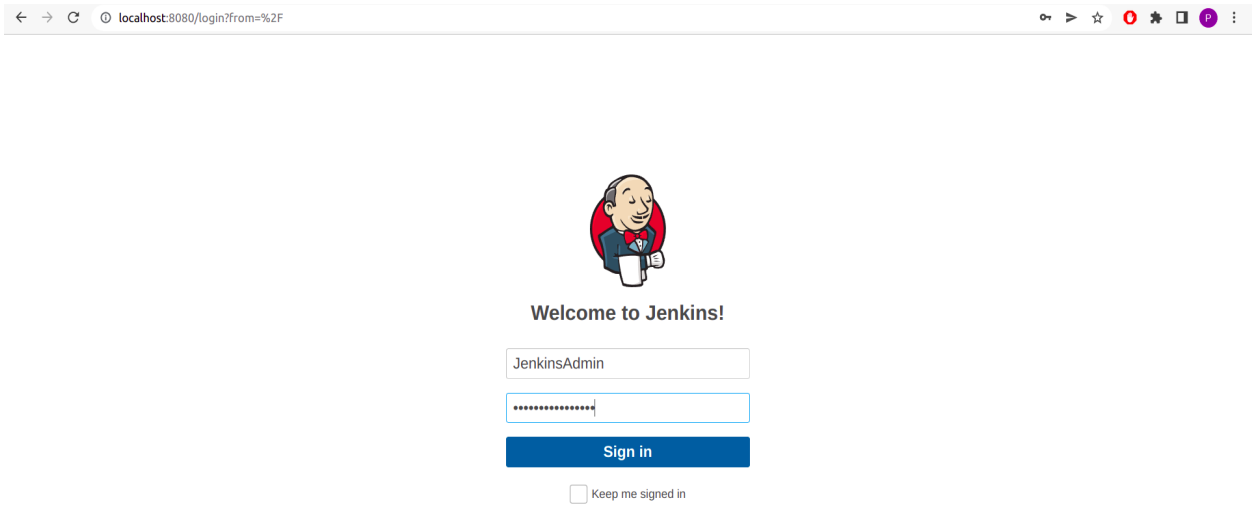
## Steps to install and setup jenkins:

- 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

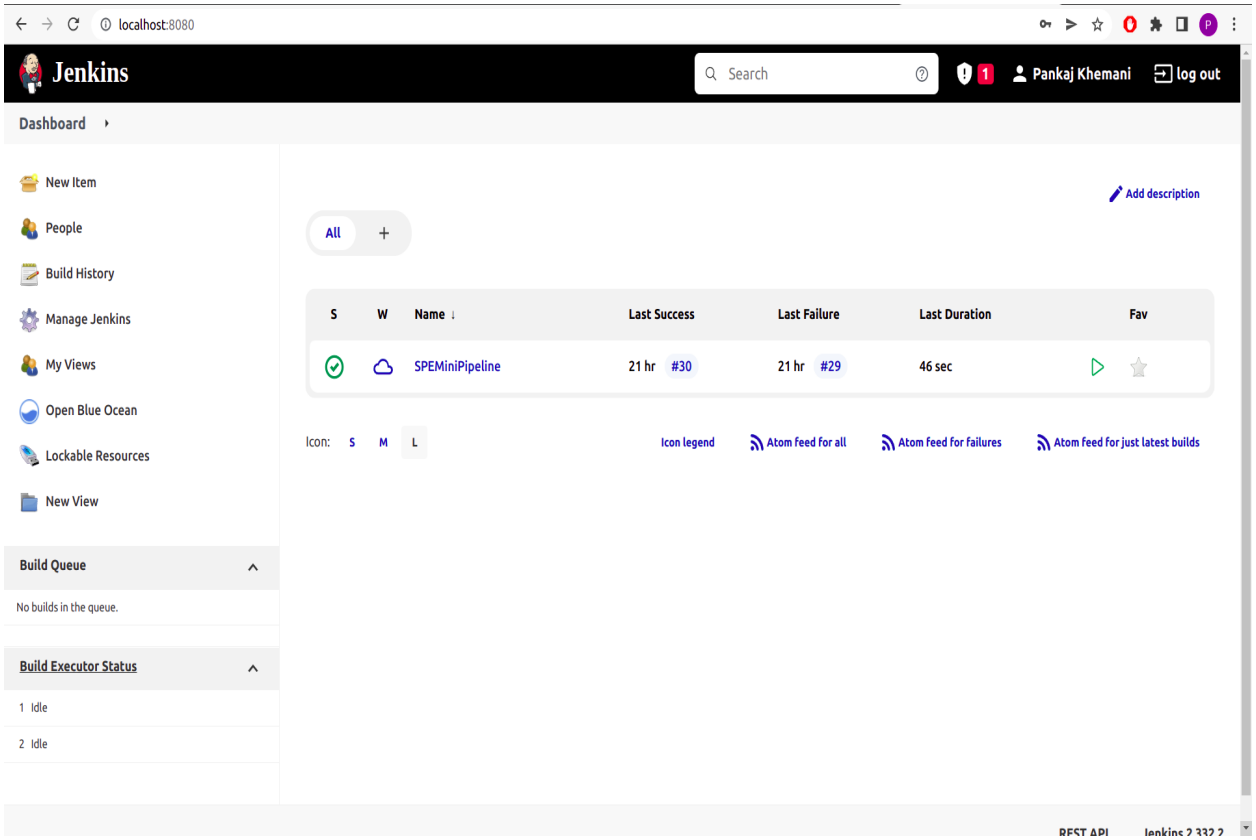



- Jenkins runs on the localhost at port 8080 by default.
- Open localhost:8080 in a browser.
- Print default password to login into jenkins
    - $ cat /var/lib/jenkins/secrets/initialAdminPassword

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




OBJ



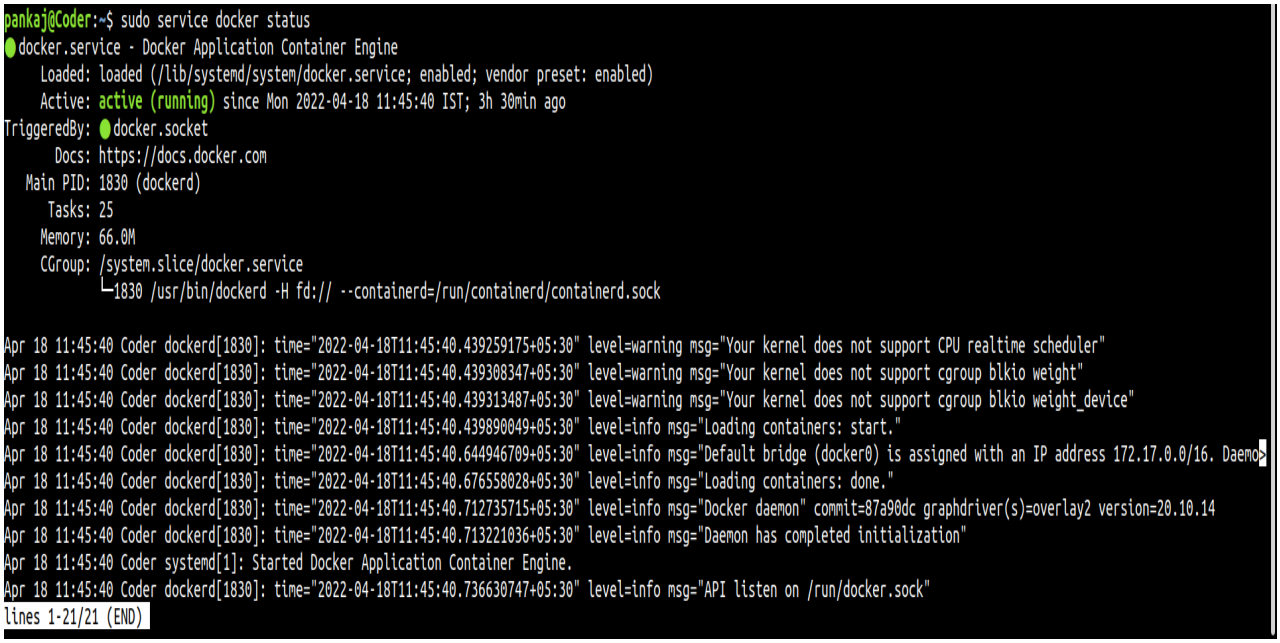

# 7) Docker:

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




- 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/
```

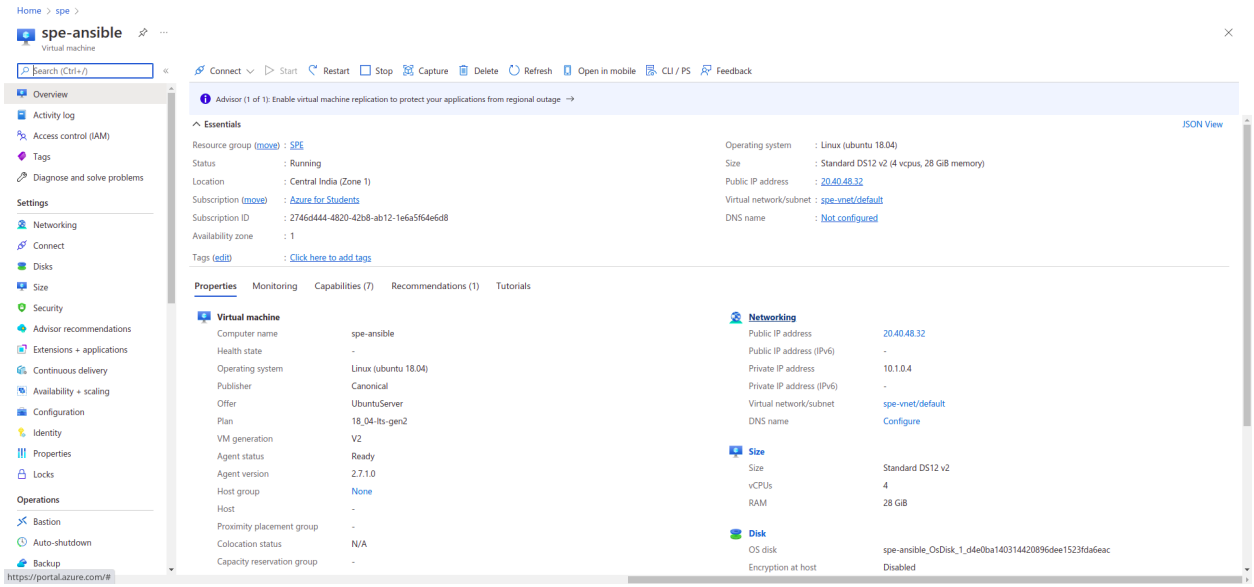
If the above steps for executing docker without sudo don't work then you can also try the below command if docker is giving any permission related error.

- $ 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.

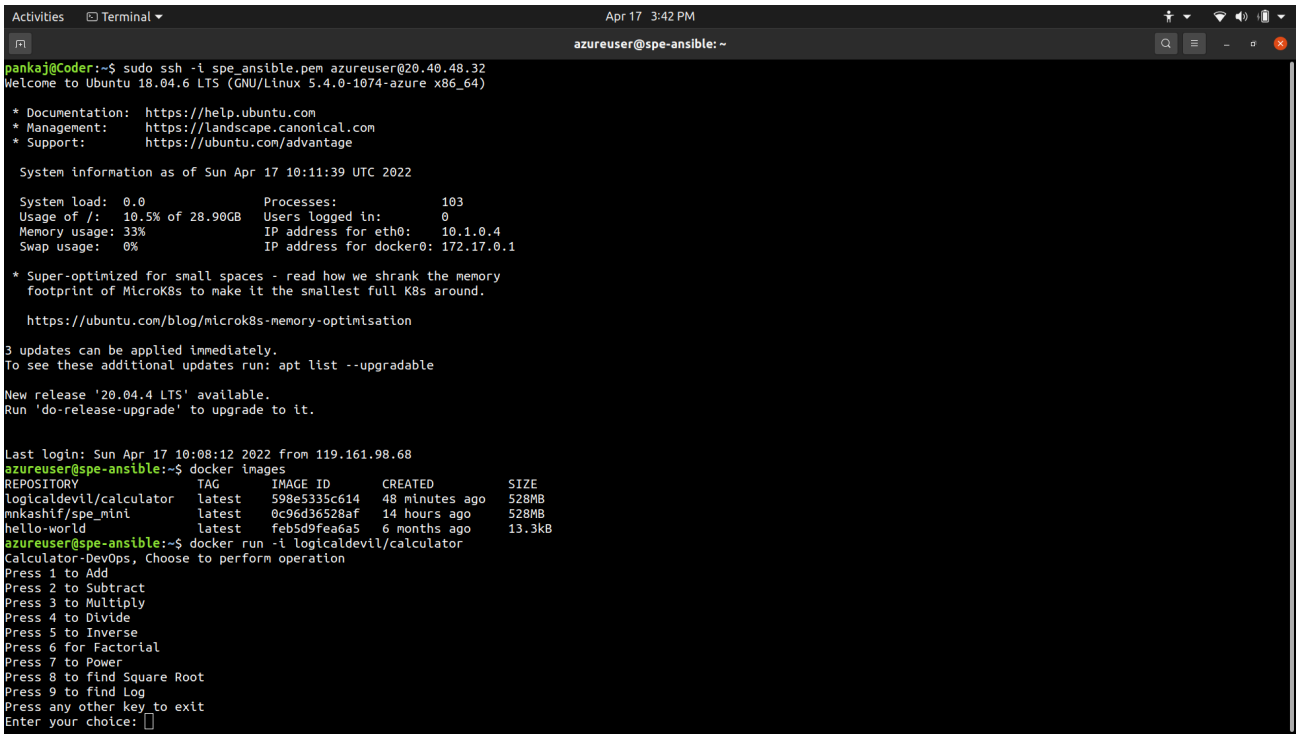



We are creating this VM for checking the deployment of our calculator application. The Jenkins server runs on our local machine and in one of the stages of the jenkins pipeline, we use ansible to pull the docker image of our calculator application on the azure VM.

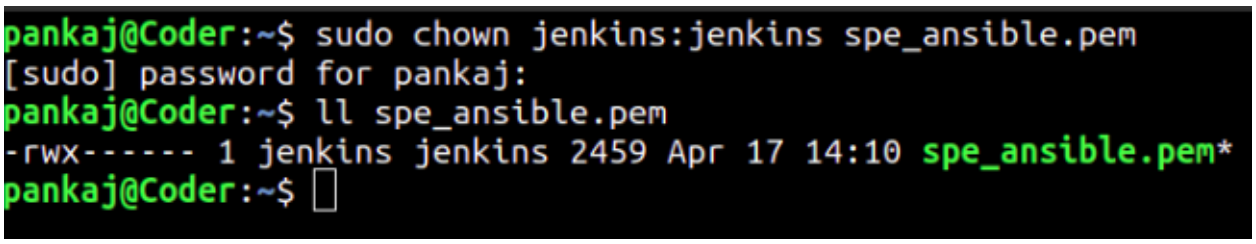
We connect to the azure VM by using the command,
$ sudo ssh -i <path to .pem key file> <VMusername>@<ip>

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>




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.




## 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

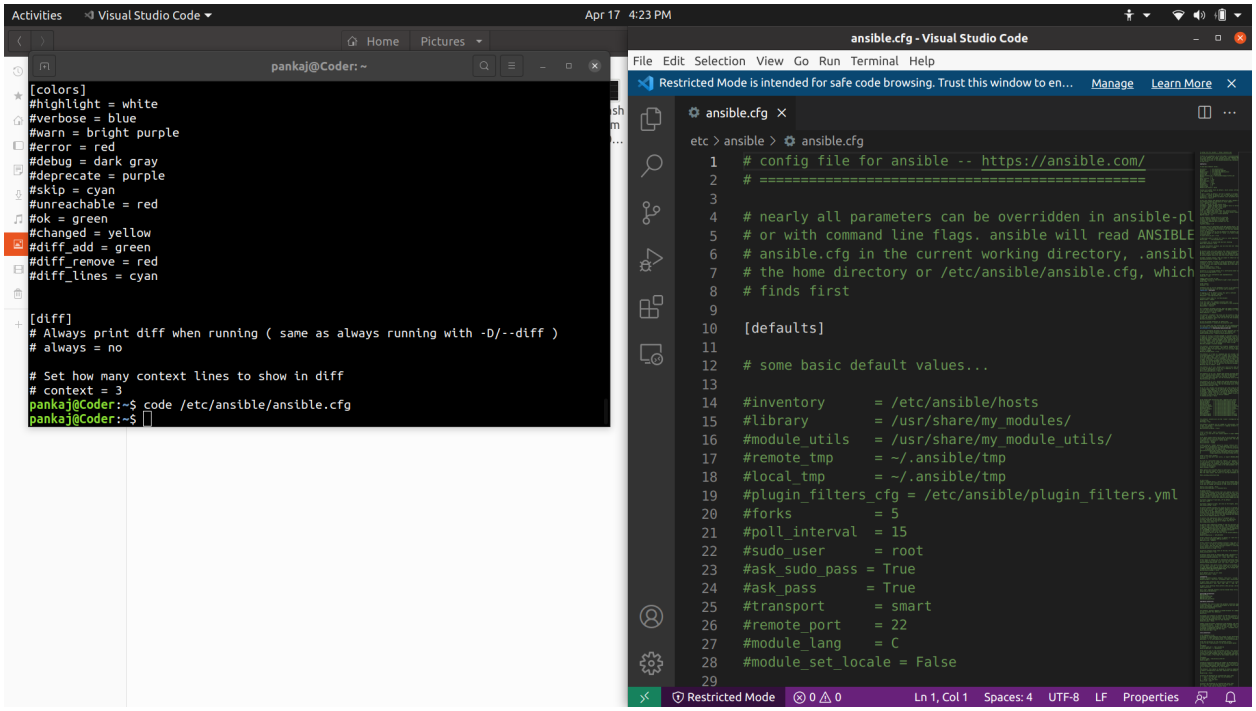
servers. It uses the playbook to describe automation jobs, and the playbook uses a very simple language YAML.

**Steps to install:**

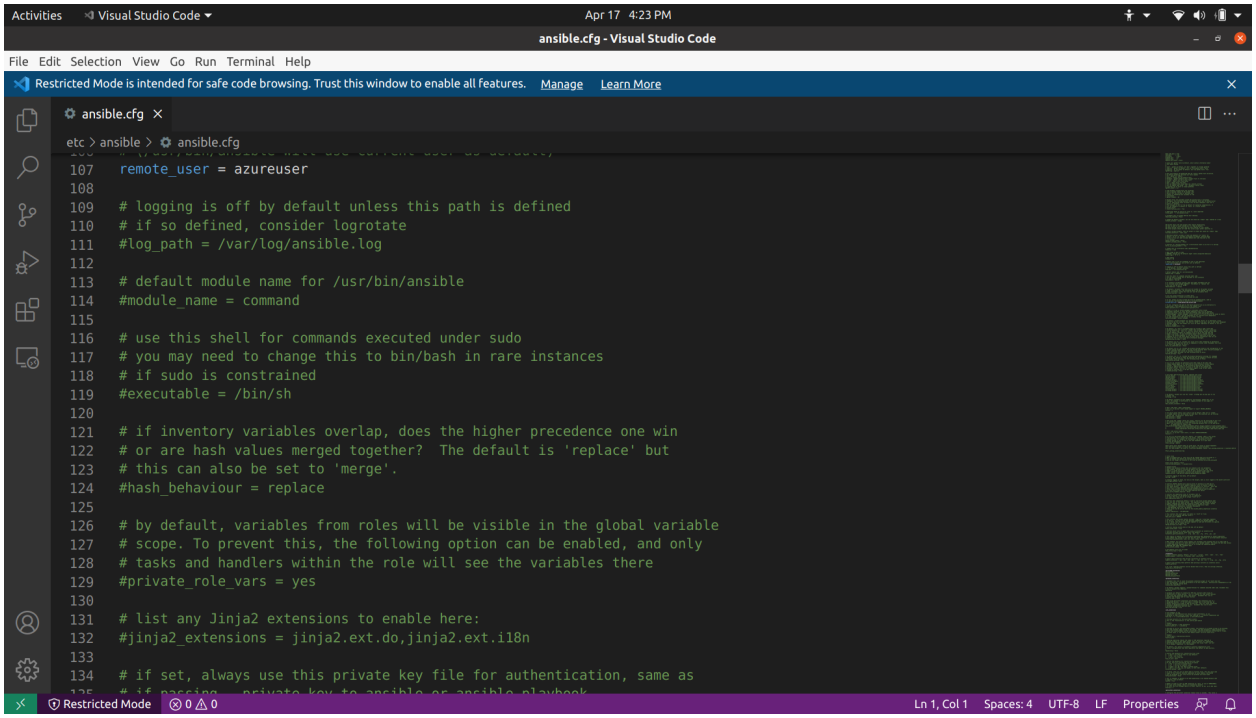
- Install Ansible on Local machine
    - $ sudo apt update
    - $ sudo apt install ansible
- Check Ansible Version
    - $ ansible --version

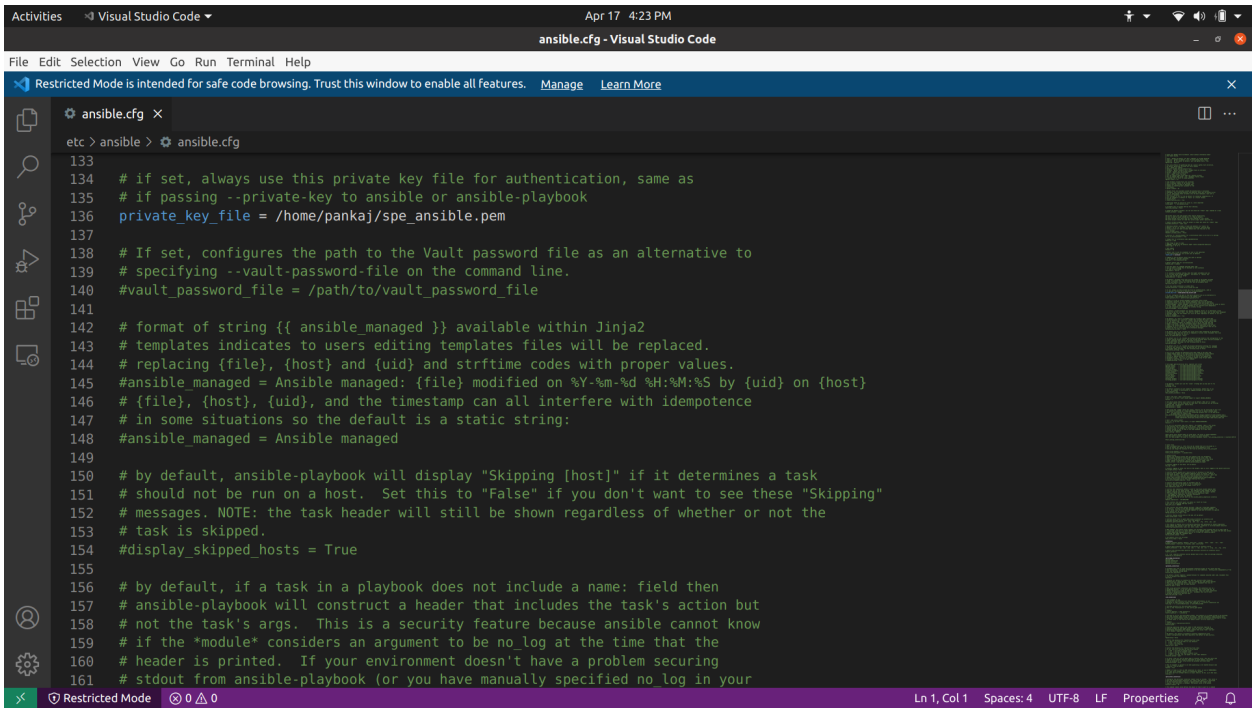
```
pankaj@Coder:~$ ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/pankaj/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /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:




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
Apr 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
remote_user = azureuser
# logging is off by default unless this path is defined
# if so defined, consider logrotate
#log_path = /var/log/ansible.log
# default module name for /usr/bin/ansible
#module_name = command
# use this shell for commands executed under sudo
# you may need to change this to bin/bash in rare instances
# if sudo is constrained
#executable = /bin/sh
# if inventory variables overlap, does the higher precedence one win
# or are hash values merged together? The default is 'replace' but
# this can also be set to 'merge'.
#hash_behaviour = replace
# by default, variables from roles will be visible in the global variable
# scope. To prevent this, the following option can be enabled, and only
# tasks and handlers within the role will see the variables there
#private_role_vars = yes
# list any Jinja2 extensions to enable here:
#jinja2_extensions = jinja2.ext.do,jinja2.ext.i18n
# if set, always use this private key file for authentication, same as
Restricted Mode
0 0
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Properties

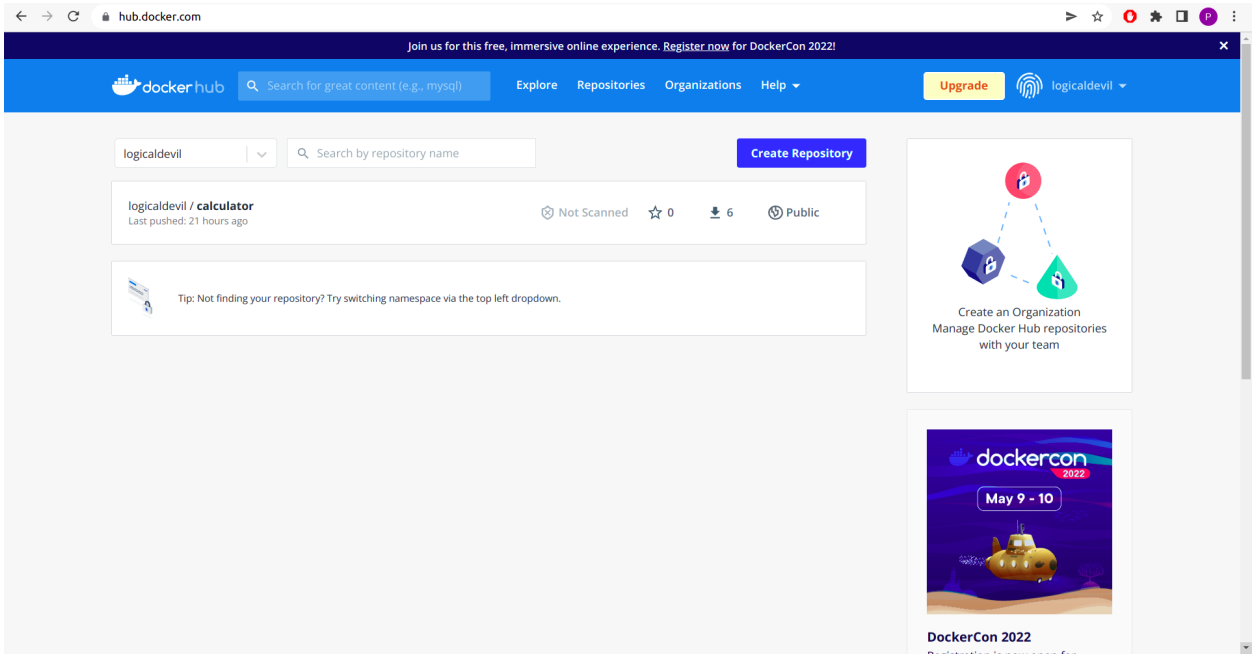

Activities
Visual Studio Code
Apr 17 4:23 PM
ansible.cfg - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
ansible.cfg
etc > ansible > ansible.cfg
# if set, always use this private key file for authentication, same as
# if passing --private-key to ansible or ansible-playbook
private_key_file = /home/pankaj/spe_ansible.pem
# If set, configures the path to the Vault password file as an alternative to
# specifying --vault-password-file on the command line.
#vault_password_file = /path/to/vault_password_file
# format of string {{ ansible_managed }} available within Jinja2
# templates indicates to users editing templates files will be replaced.
# replacing {file}, {host} and {uid} and strftime codes with proper values.
#ansible_managed = Ansible managed: {file} modified on %Y-%m-%d %H:%M:%S by {uid} on {host}
# {file}, {host}, {uid}, and the timestamp can all interfere with idempotence
# in some situations so the default is a static string:
#ansible_managed = Ansible managed
# by default, ansible-playbook will display "Skipping [host]" if it determines a task
# should not be run on a host. Set this to "False" if you don't want to see these "Skipping"
# messages. NOTE: the task header will still be shown regardless of whether or not the
# task is skipped.
#display_skipped_hosts = True
# by default, if a task in a playbook does not include a name: field then
# ansible-playbook will construct a header that includes the task's action but
# not the task's args. This is a security feature because ansible cannot know
# if the *module* considers an argument to be no_log at the time that the
# header is printed. If your environment doesn't have a problem securing
# stdout from ansible-playbook (or you have manually specified no_log in your
Restricted Mode
0 0
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

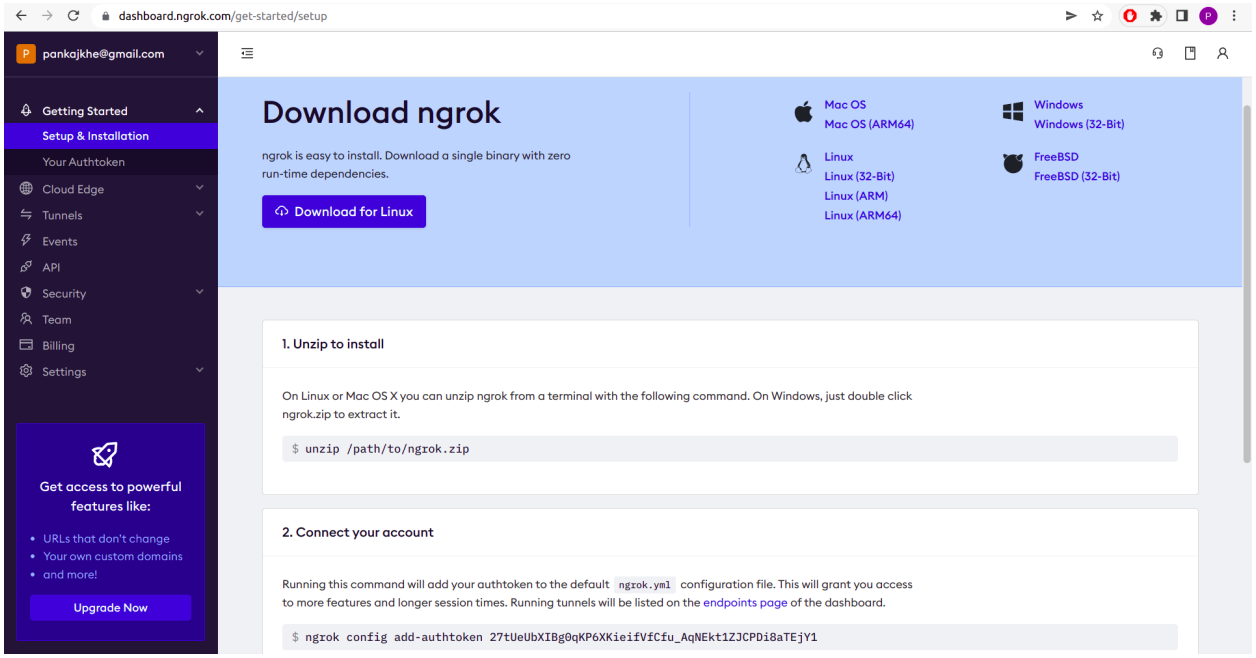


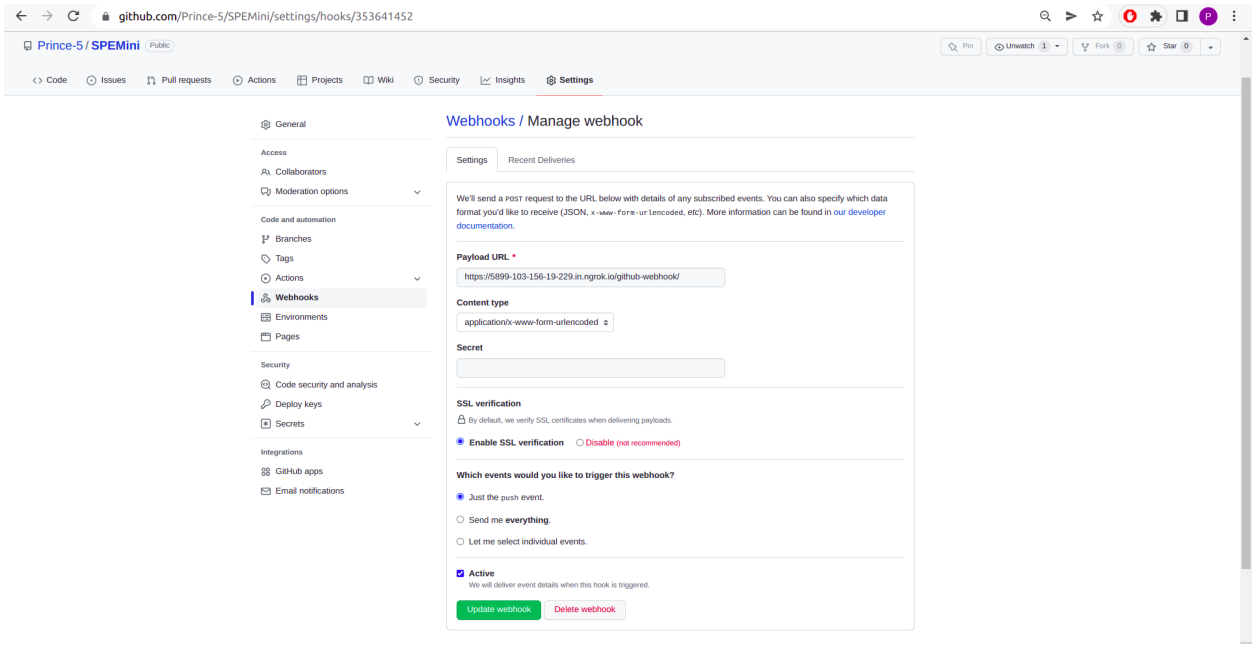

# 11) Ngrok and webhook:

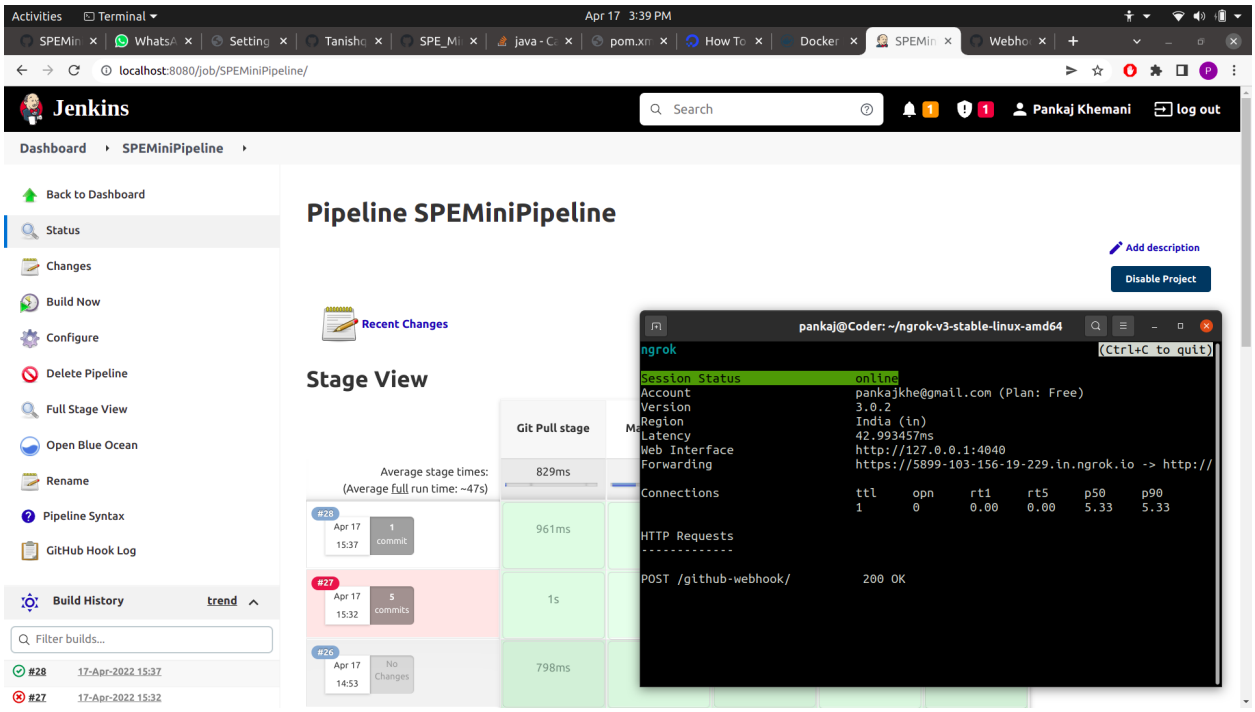
Ngrok is a cross-platform application that enables developers to expose a local development server to the Internet with minimal effort. The software makes your locally-hosted web server appear to be hosted on a subdomain of ngrok.com, meaning that no public IP or domain name on the local machine is needed.

## 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
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Actions
Webhooks
Environments
Pages
Security
Code security and analysis
Deploy keys
Secrets
Integrations
GitHub apps
Email notifications
Webhooks / Manage webhook
Settings
Recent Deliveries
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 *
https://5899-103-156-19-229.in.ngrok.io/github-webhook/
Content type
application/x-www-form-urlencoded
Secret
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

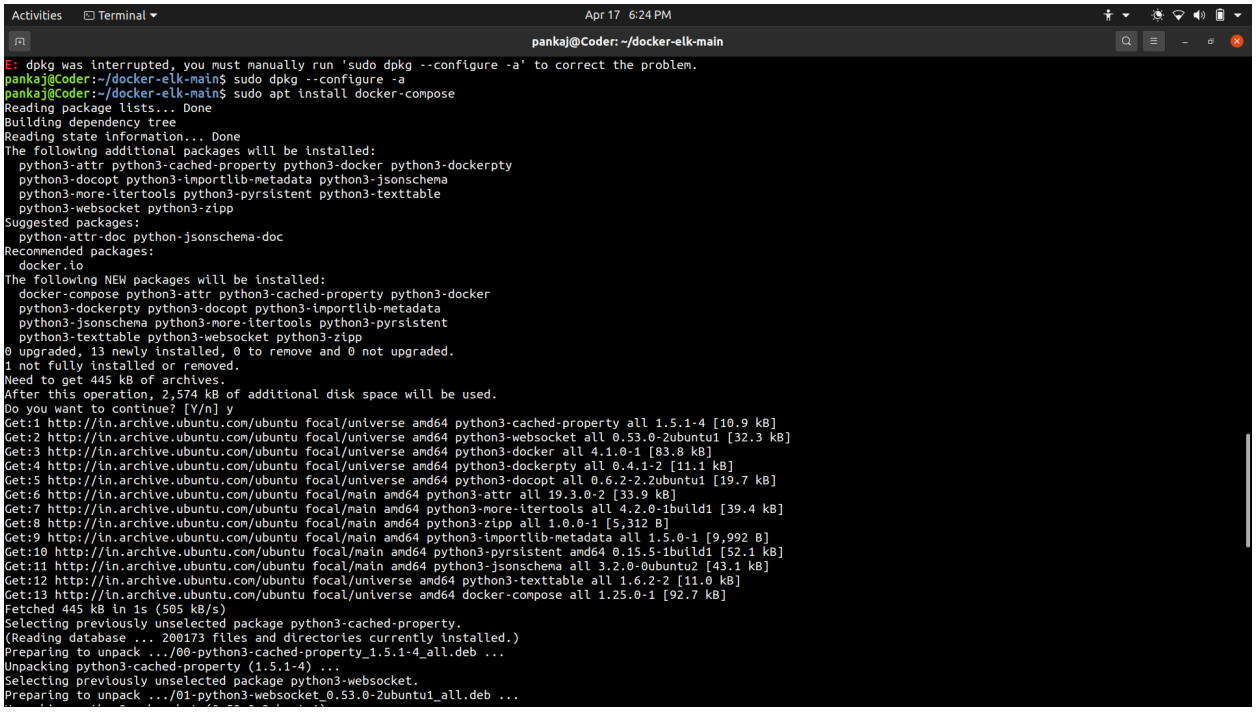

Activities
Terminal
Apr 17 3:39 PM
localhost:8080/job/SPEMiniPipeline/
Jenkins
Search
Pankaj Khemani
log out
Dashboard
SPEMiniPipeline
Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Open Blue Ocean
Rename
Pipeline Syntax
GitHub Hook Log
Build History
trend
Filter builds...
#28 17-Apr-2022 15:37
#27 17-Apr-2022 15:32
Pipeline SPEMiniPipeline
Add description
Disable Project
Recent Changes
Stage View
Git Pull stage
Average stage times:
(Average full run time: ~47s)
829ms
#28
Apr 17
15:37
1 commit
961ms
#27
Apr 17
15:32
5 commits
1s
#26
Apr 17
14:53
No Changes
798ms
pankaj@Coder: ~/ngrok-v3-stable-linux-amd64
ngrok
(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 rt1 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




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.

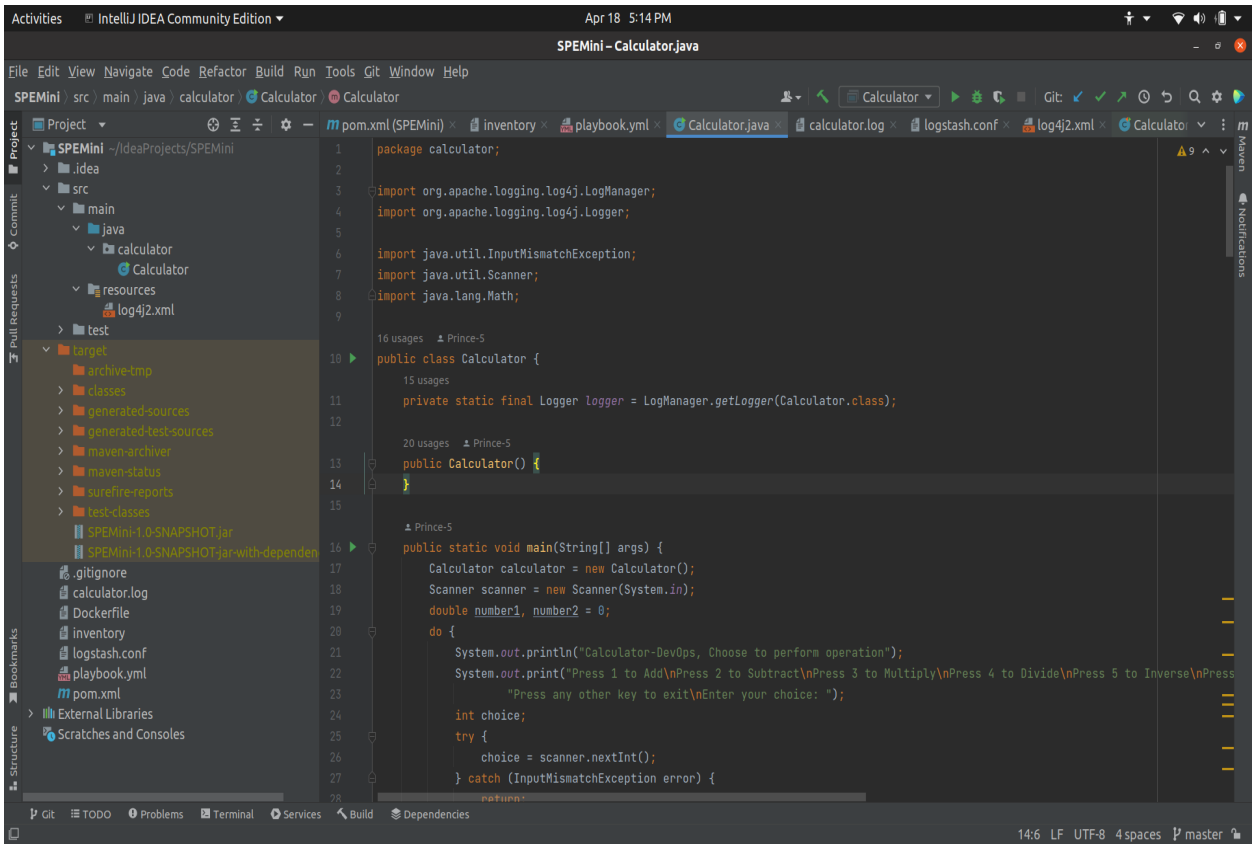
# Part 2 - Developing the Project:

1) Write the code for calculator:

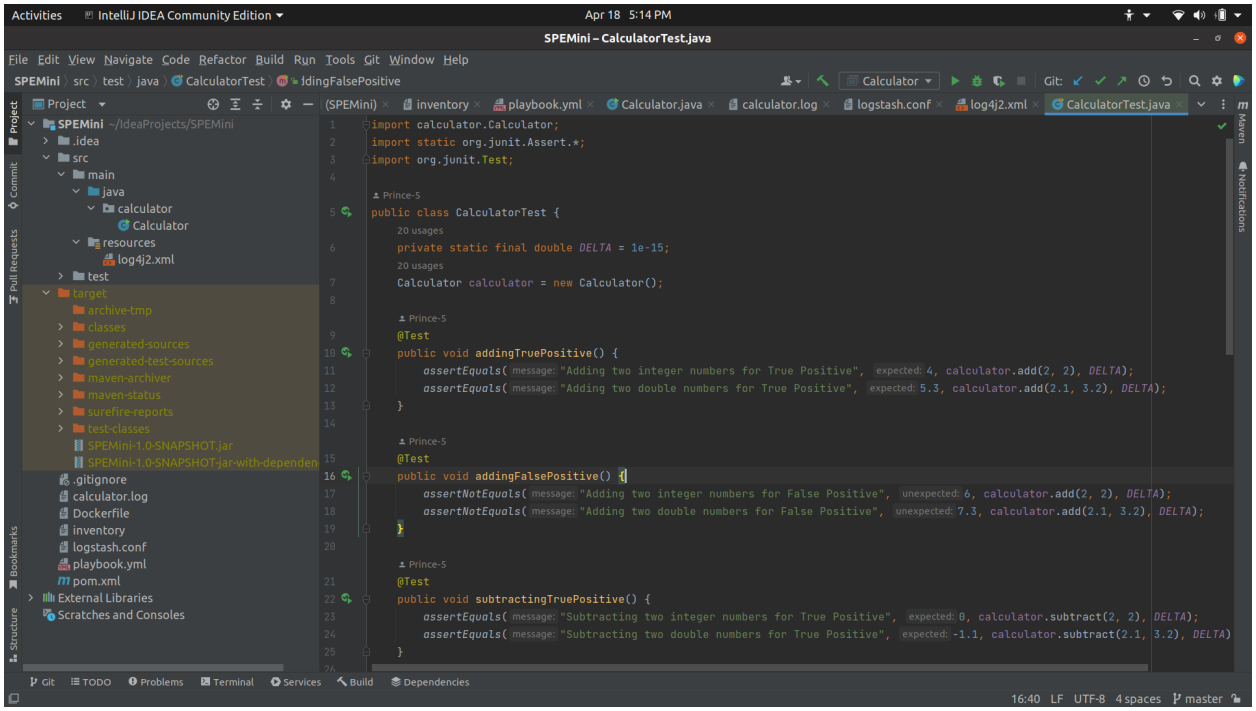
Set up a new project in IntelliJ

a) Open Intellij and select a new project.
b) Select maven from the menu.
c) Give a name and click Finish

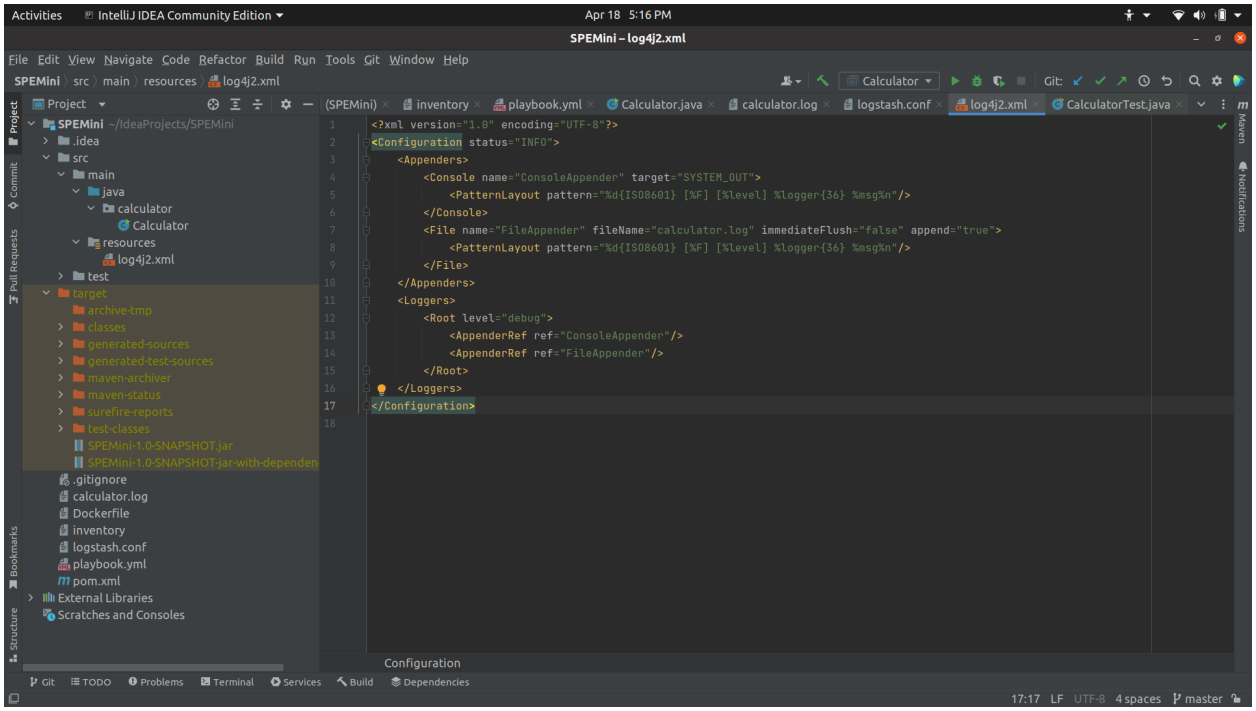
Go to src/main. Right click on java and select new/package.
Write the calculator code here.



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

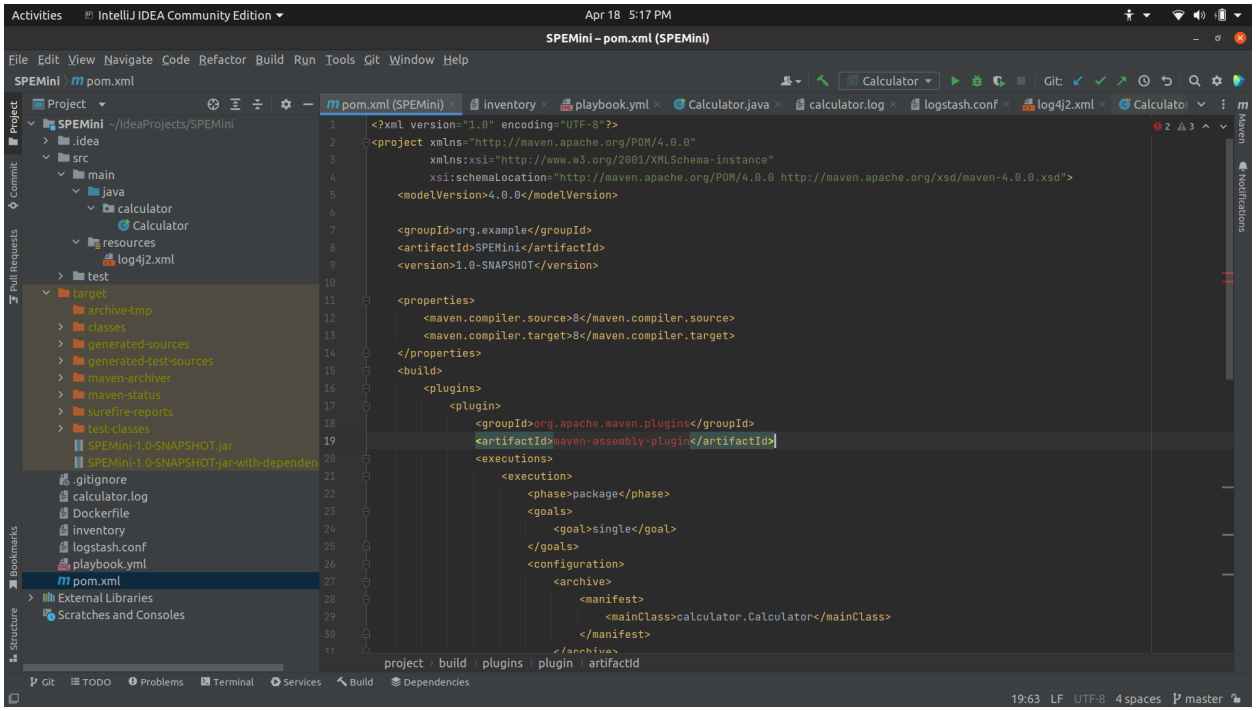



Go to src/main/resources, and create a log4j2.xml file and specify the format for the log file.

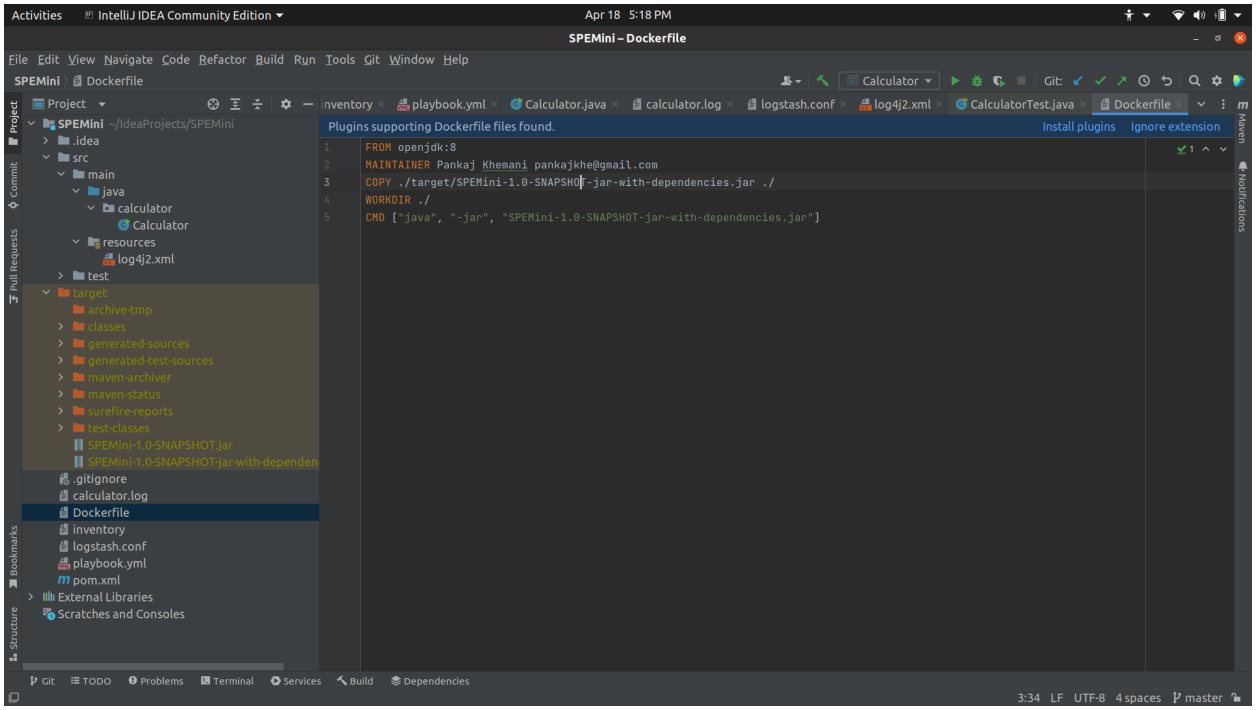


Now create the following files in the project folder: pom.xml, dockerfile, inventory, playbook.yml

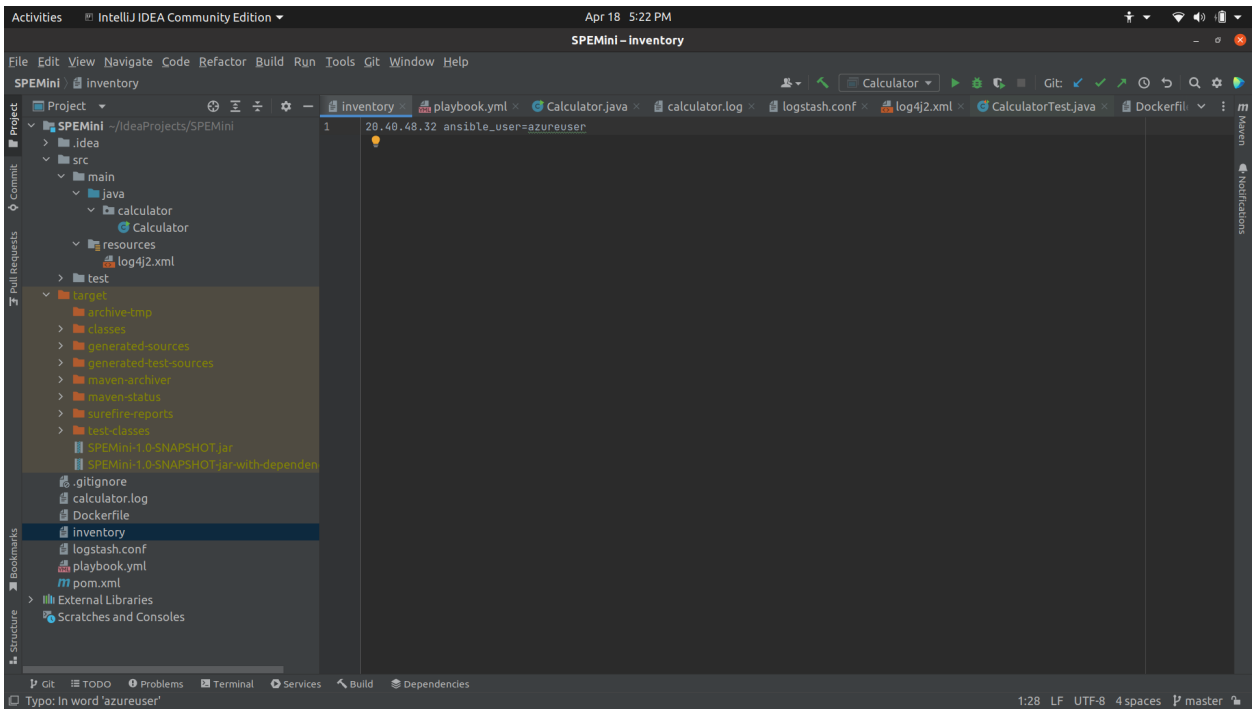
Pom.xml: This contains all the dependencies for the code. It is used by maven to build the code.



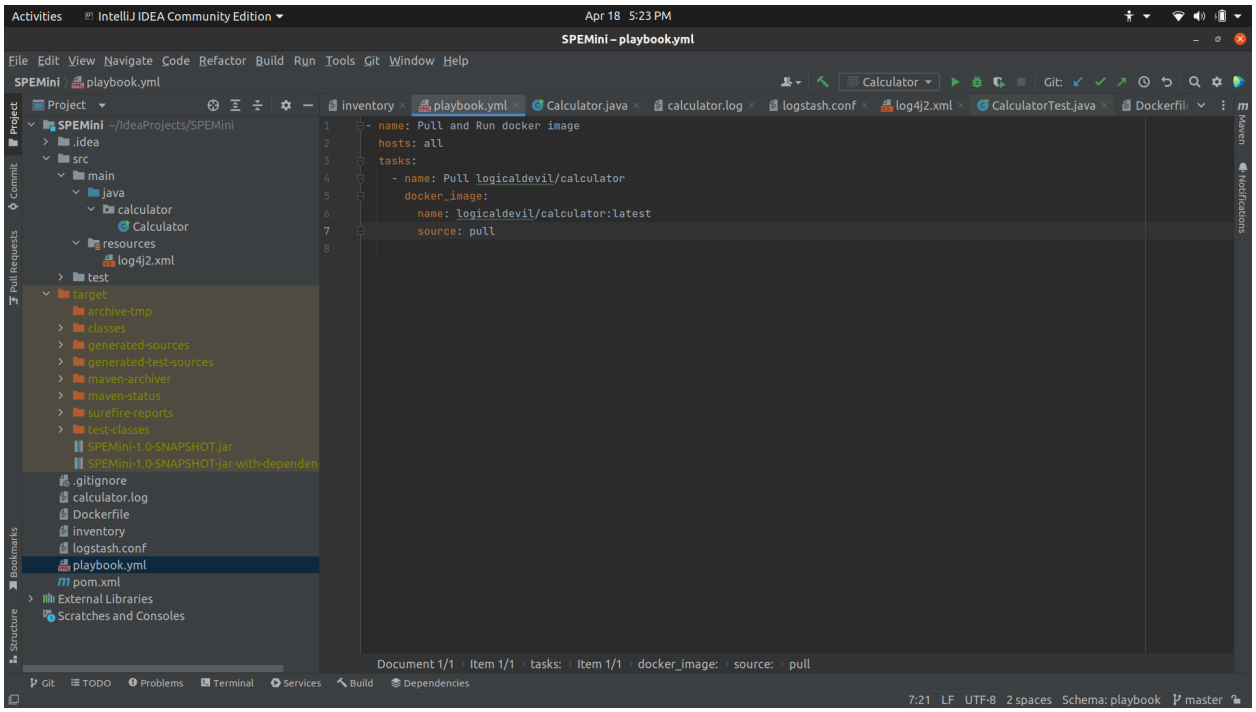

Dockerfile: Docker uses this to construct the image. We're using an openjdk base image here, and after it's up and running, it'll copy and run the calculator jar file.




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



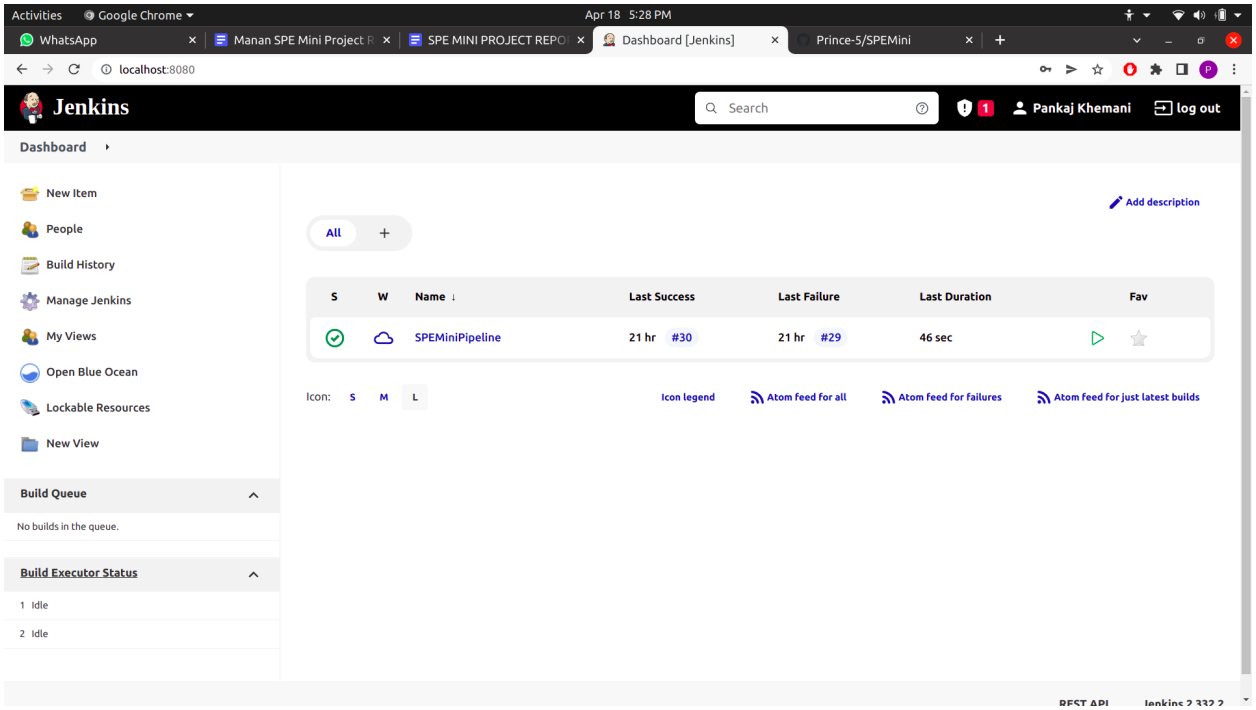
Playbook.yml: It contains the commands to be executed on the remote machine.



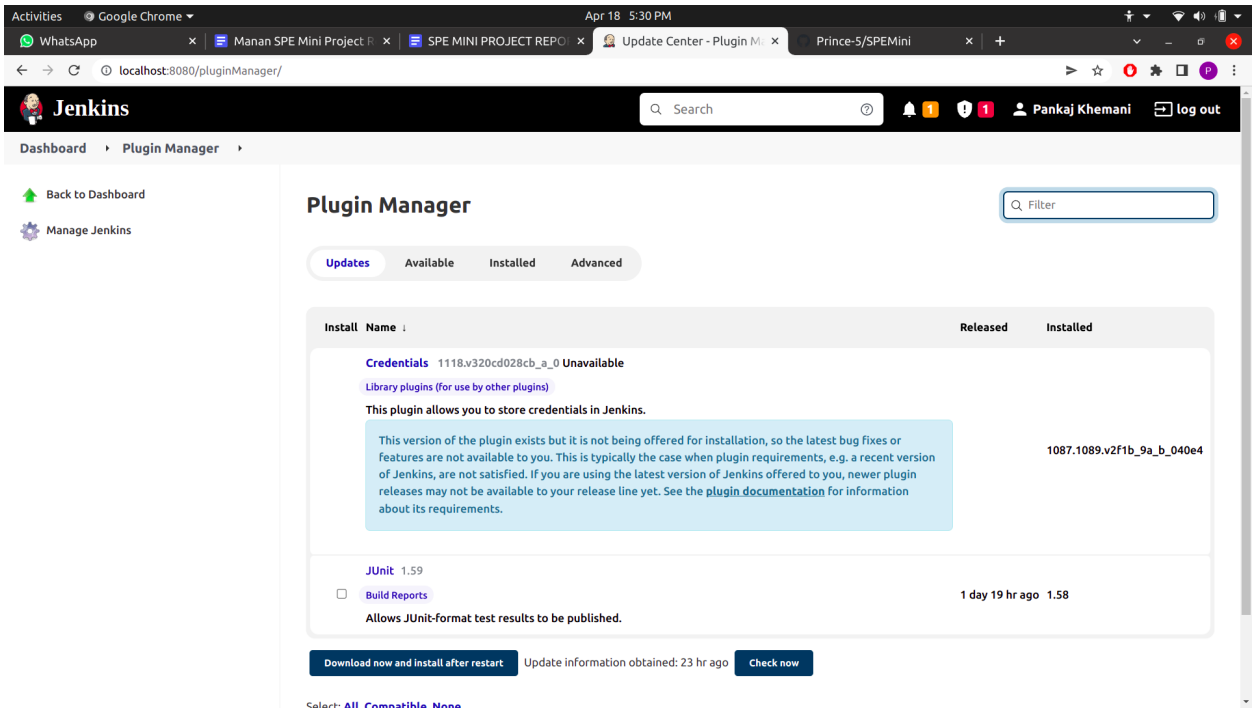

- All the java files can be run separately and checked for faults.
- Click build and run for testing the whole code.
- After this use git add, commit and push and push the code to the github repo.

## 2) Setting up Jenkins PipeLine:

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

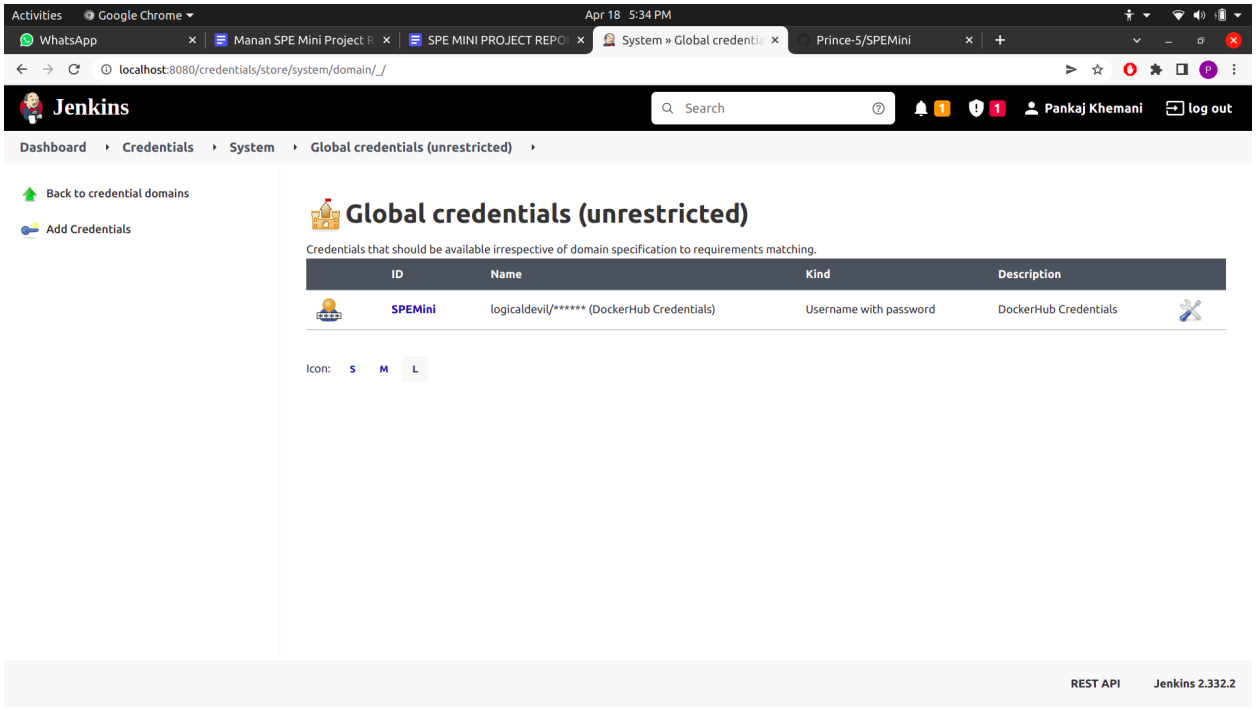



- Go to Dashboard/manage Jenkins/manage plugins/available.

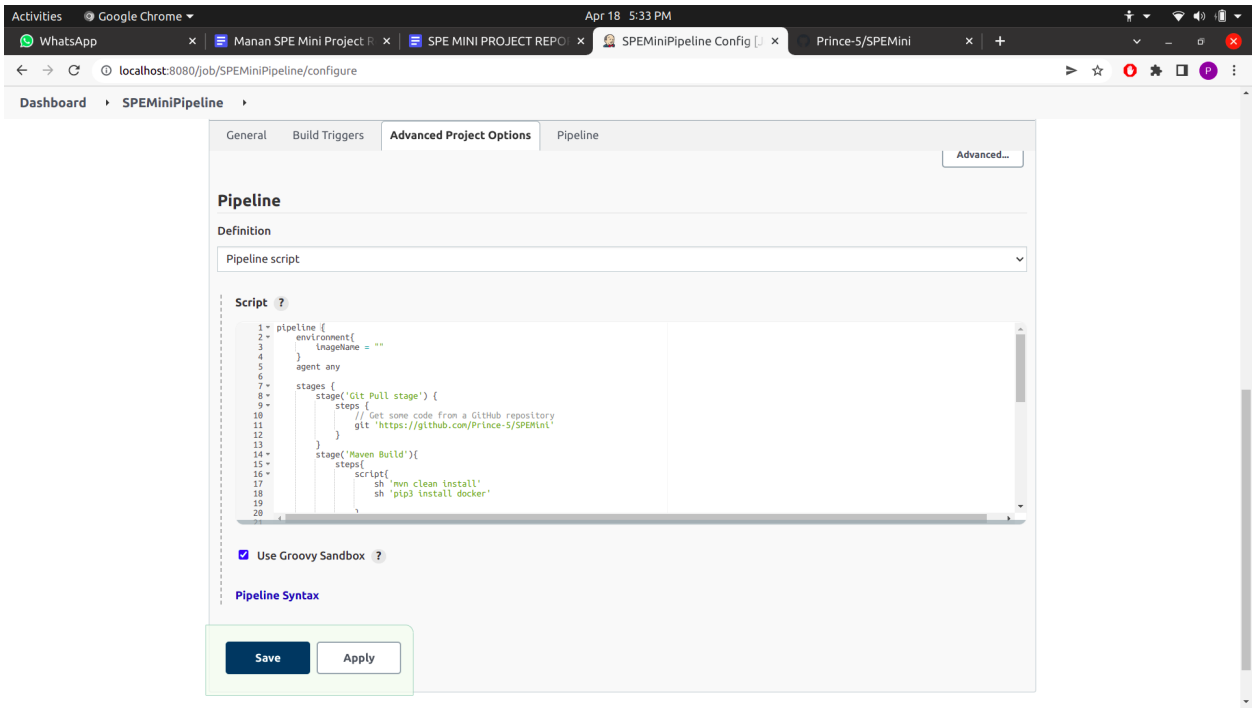



- Install the following plugins
  - Docker
  - Docker plugin

    - 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/



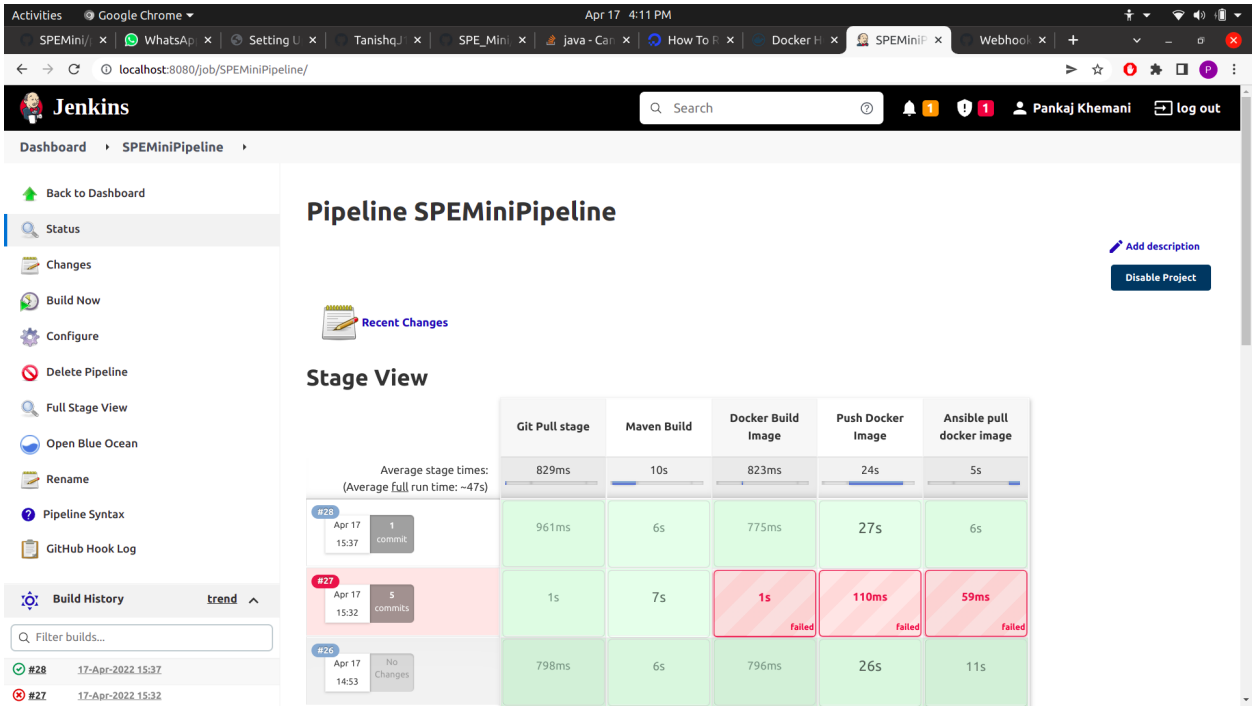

- Click on Add Credentials.
- Enter Docker Hub and Github username, password, and give an ID to these credentials.
- Go to Dashboard/manage Jenkins/Configure system/Git
    - Add the name and path to the git executable.
- Go to Dashboard/manage Jenkins/Configure system/Ansible
    - Add the name and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




- Update the values in the pipeline and save.

## 3) Last Steps:

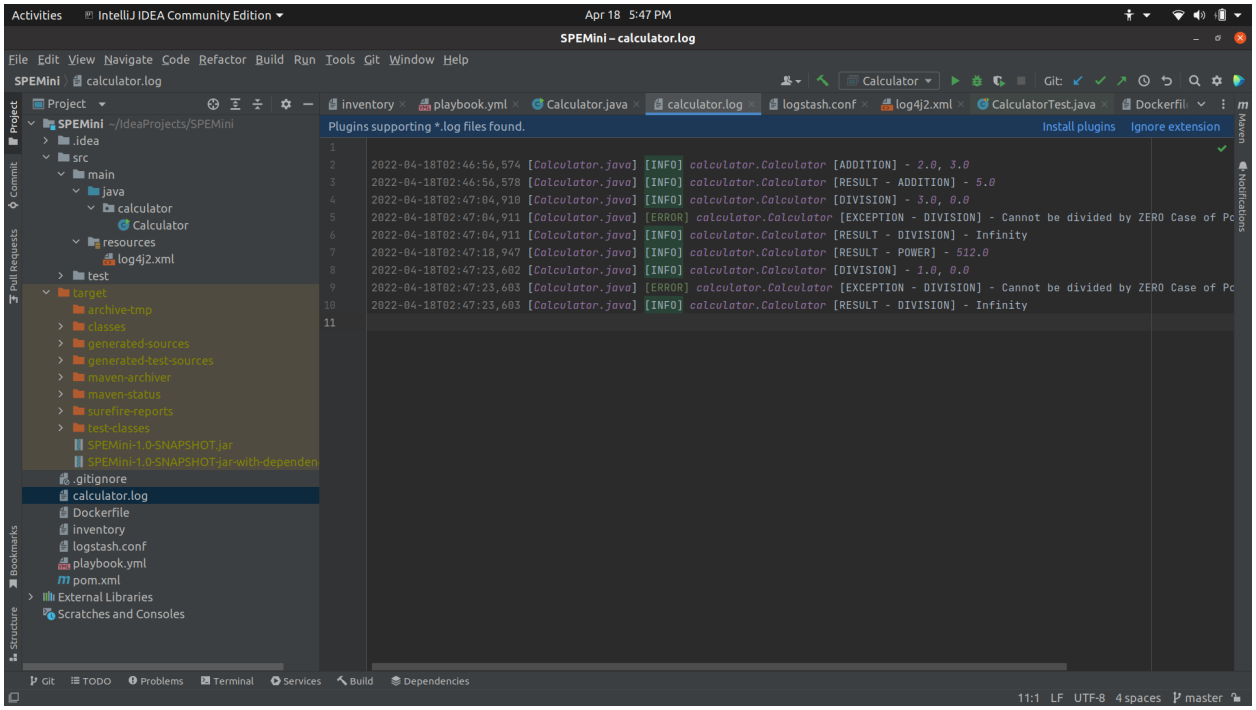
- Now everything is setup-including the jenkins pipeline. The build now option starts the pipeline. The green boxes indicate that the stage is completed successfully.



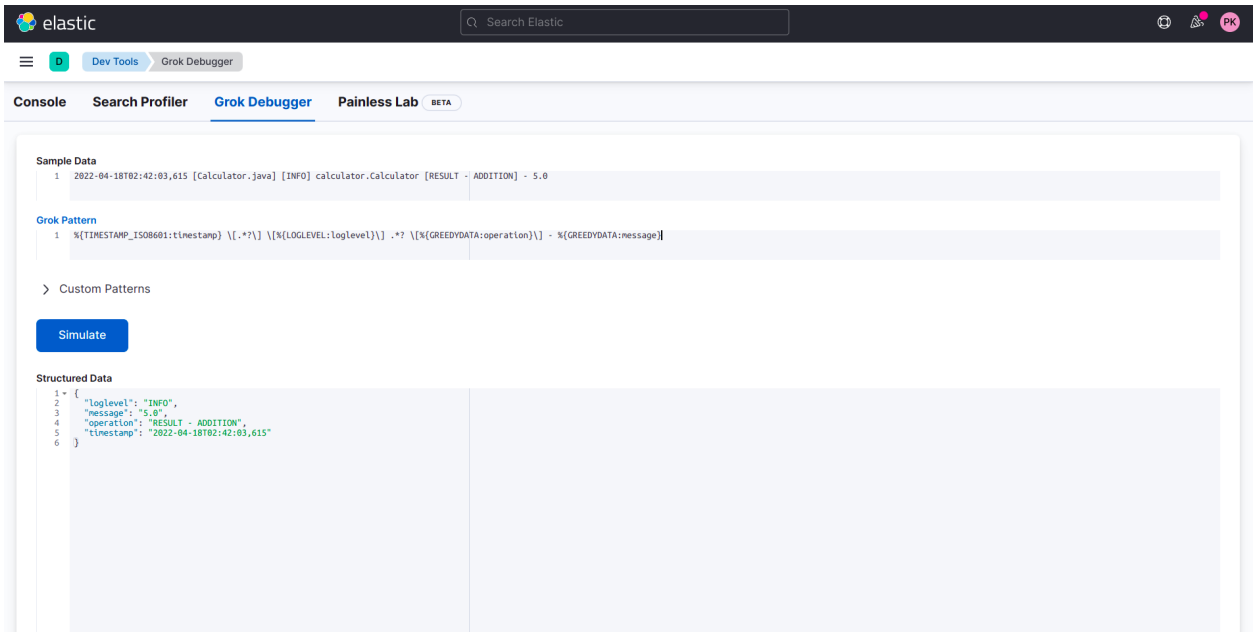

- Now when the execution of the pipeline gets completed, we can connect to the azureuser(VM) and check if our calculator application is running there.
- I have already covered how to run the calculator application on VM in Virtual Machine on Azure part above. The images are also attached there.
- After 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.

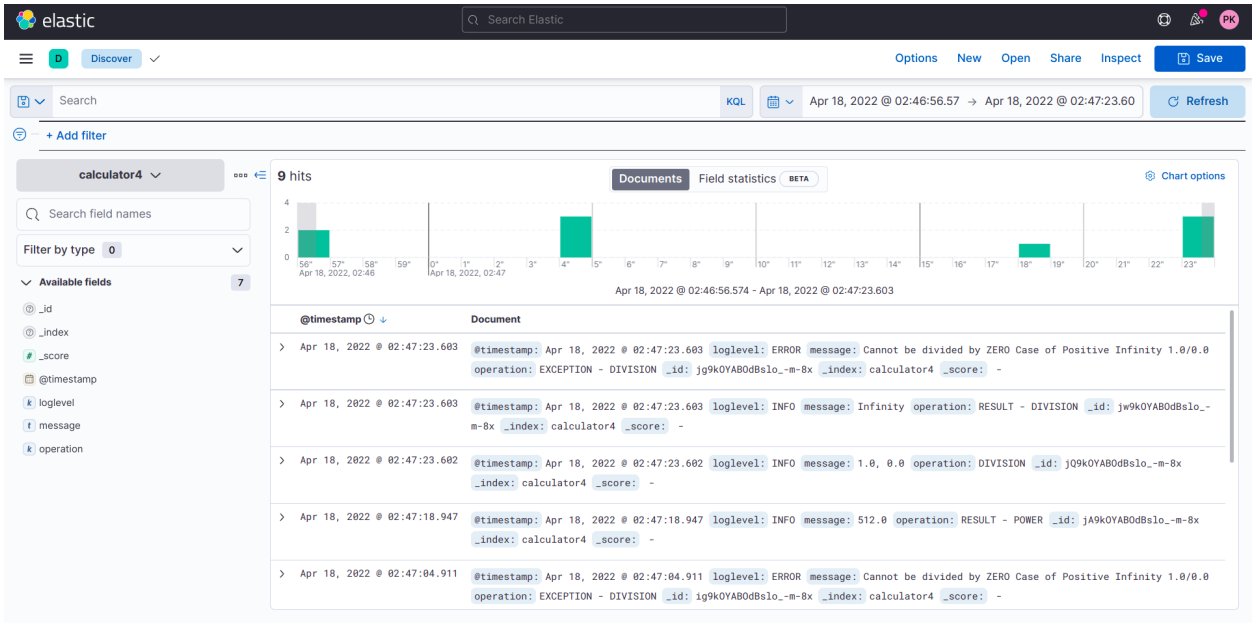


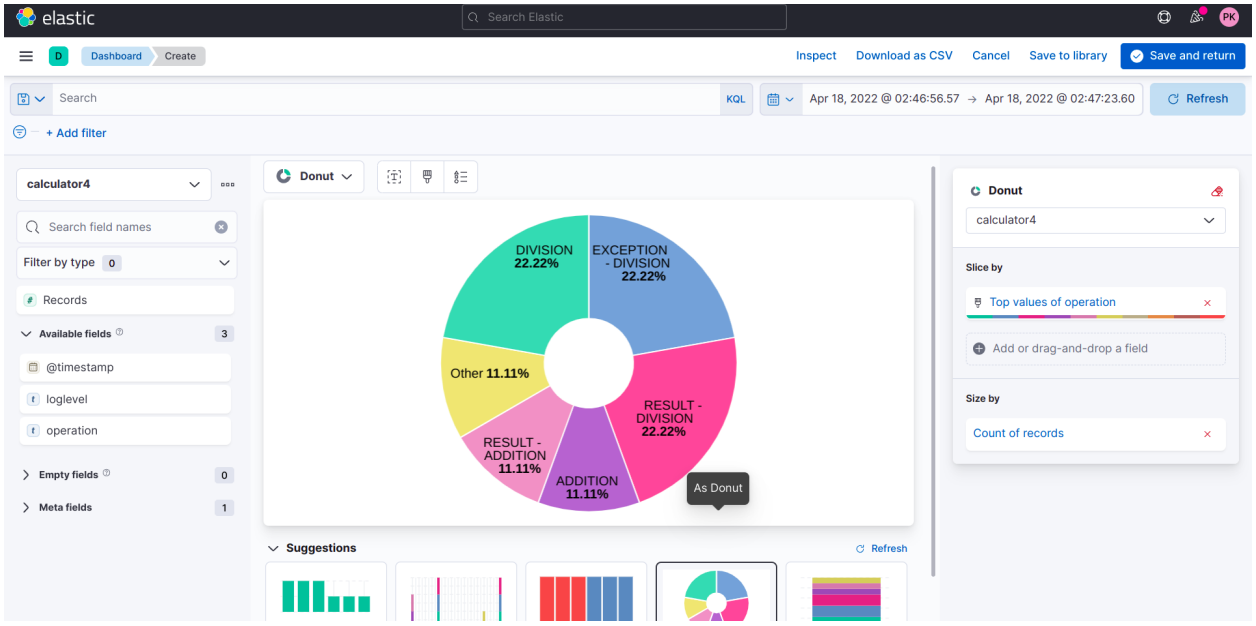

After uploading this file, select or type the appropriate grok pattern for the semi structured text. This can be found out using the grok debugger in the dev tools section for kibana.

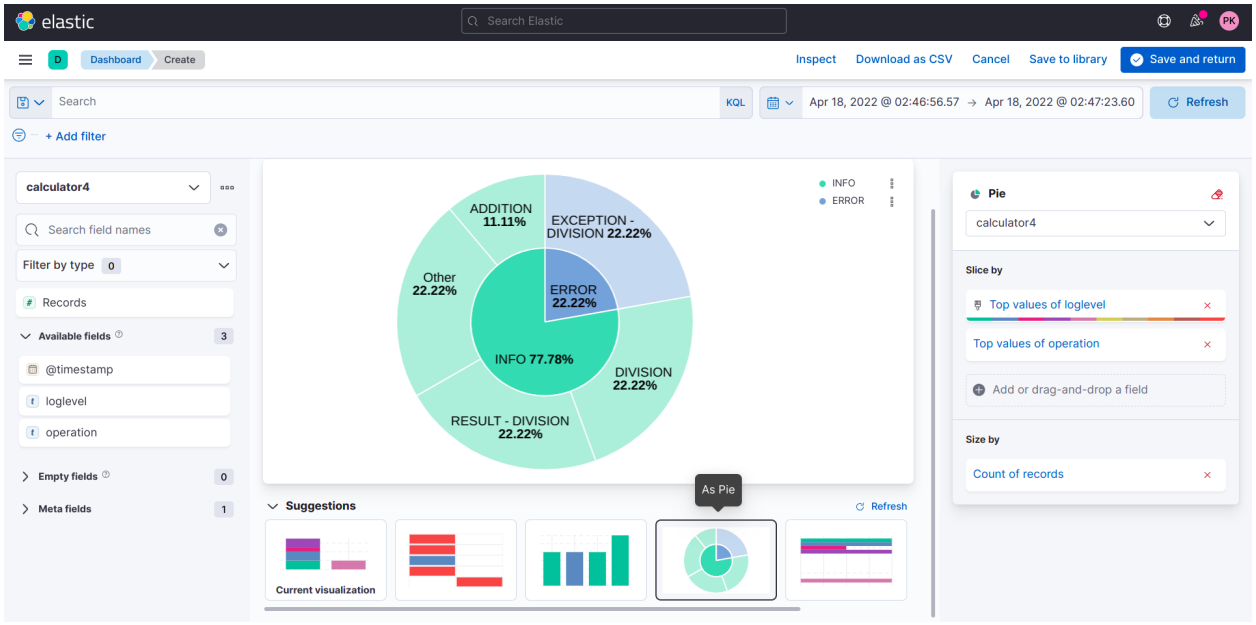


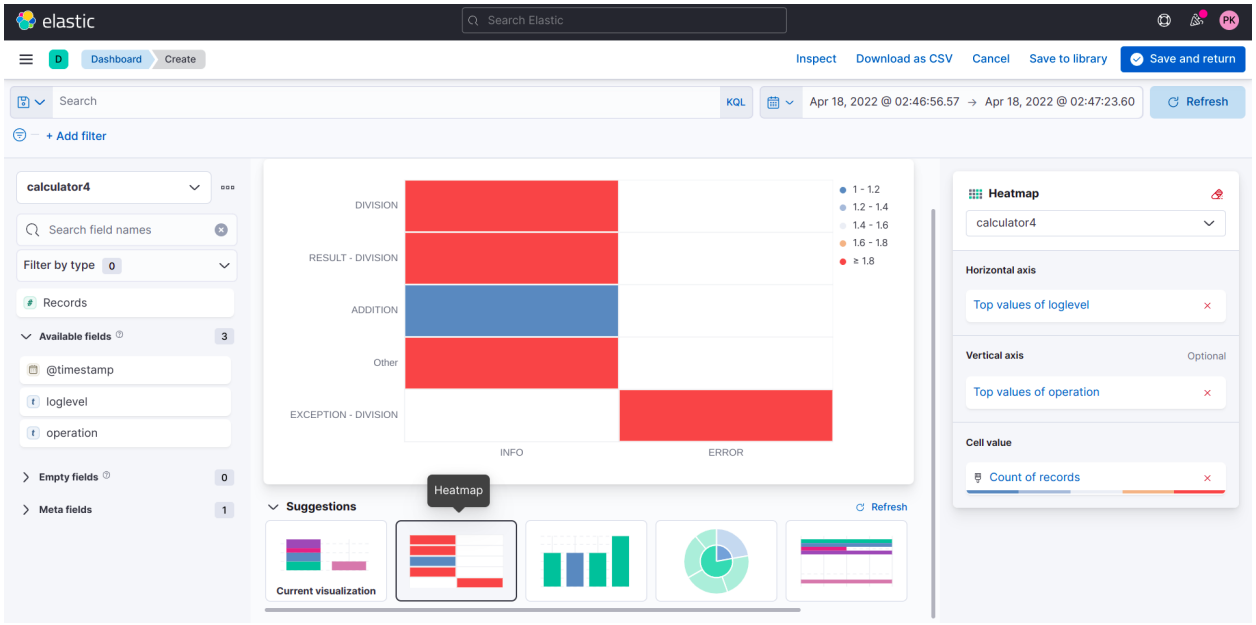
After this we can just use the UI for the monitoring visualizations and can generate many types of visualizations with different fields.

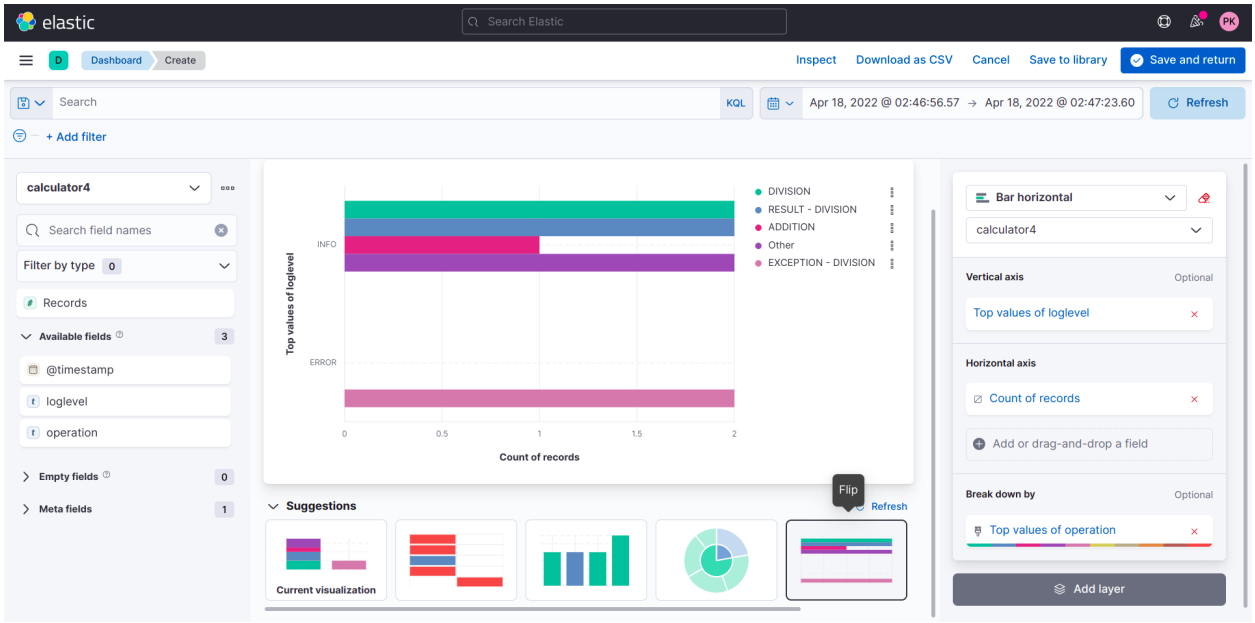
Below are the screenshots of visualizations for my calculator.log file:




elastic
Search Elastic
Dashboard
Create
Inspect
Download as CSV
Cancel
Save to library
Save and return
Search
KQL
Apr 18, 2022 @ 02:46:56.57 → Apr 18, 2022 @ 02:47:23.60
Refresh
+ Add filter
calculator4
Search field names
Filter by type 0
Records
Available fields 3
@timestamp
loglevel
operation
Empty fields 0
Meta fields 1
ADDITION 11.11%
EXCEPTION - DIVISION 22.22%
Other 22.22%
ERROR 22.22%
INFO 77.78%
DIVISION 22.22%
RESULT - DIVISION 22.22%
INFO
ERROR
Suggestions
Refresh
As Pie
Current visualization
Pie
calculator4
Slice by
Top values of loglevel
Top values of operation
Add or drag-and-drop a field
Size by
Count of records


elastic
Search Elastic
Dashboard
Create
Inspect
Download as CSV
Cancel
Save to library
Save and return
Search
KQL
Apr 18, 2022 @ 02:46:56.57 → Apr 18, 2022 @ 02:47:23.60
Refresh
+ Add filter
calculator4
Search field names
Filter by type 0
Records
Available fields 3
@timestamp
loglevel
operation
Empty fields 0
Meta fields 1
DIVISION
RESULT - DIVISION
ADDITION
Other
EXCEPTION - DIVISION
INFO
ERROR
1 - 1.2
1.2 - 1.4
1.4 - 1.6
1.6 - 1.8
≥ 1.8
Suggestions
Refresh
Heatmap
Current visualization
Heatmap
calculator4
Horizontal axis
Top values of loglevel
Vertical axis
Optional
Top values of operation
Cell value
Count of records




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

My DockerHub Repo:
https://hub.docker.com/repository/docker/logicaldevil/calculator