

Scientific Calculator

WITH DEVOPS

IMT2018037 Amith Sai Konda
amith.sai@iiitb.ac.in | IIITB

1. Introduction
 - PROBLEM
 - DevOps tool chain
 - Tools Used
 - JAVA
 - GIT, GITHUB
2. Testing, building
 - Junit
 - Maven
3. Continuous Integration
 - Definition
 - Jenkins
4. Containerization
 - Docker
 - Docker file
 - Docker hub
5. Deployment
 - Ansible
 - Creating Playbooks
 - Implementing playbooks
6. ELK Stack
7. Pipeline
8. Links
9. References

INTRODUCTION

- **PROBLEM:**

Create a scientific calculator program with user menu driven operations

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

- **DevOps tool chain:**

We can use any set of DevOps tool chains but the pipeline would be same for all. The pipeline I used for this project consists of the following:

1. Using a source control management tool - like GitHub and Bitbucket etc
2. Testing - test your code using either Junit and Selenium etc
3. Build - build your code using tool like Maven and Gradle etc
4. Continuous Integration - Continuous integrate your code using tool like Jenkins and GitHub actions etc
5. Containerize - Containerize your code using Docker.
6. Push your created Docker image to Docker hub .
7. Deployment - Do configuration management and deployment using Ansible and Rundeck etc. We use these to do configuration management and pull your docker image to 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.

- **Tools Used:**

- JAVA
- Maven
- IntelliJ IDE
- Git
- GitHub
- JUnit
- Log4j
- Docker
- Docker Hub
- Jenkins
- Ansible
- ELK Stack

- **JAVA:**

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible.

Installation on mac:

- Download the jdk.dmg file from the Oracle website
- Open the dmg file and complete the required steps
- Configure the java version u need in JAVA_HOME
- Update this path in Global Bash profile of mac

Add The following command in bash_profile. 1.8 is the JDK 8 version of java.

`JAVA_HOME="/usr/libexec/java_home -v 1.8"`

After adding we run the bash_profile using the command "source .bash_profile". It will add the java version or path.

We can check the path of Java as shown in above figure.

```
(base) amithkonda@Amiths-MacBook-Air ~ % echo $JAVA_HOME
/Library/Java/JavaVirtualMachines/jdk1.8.0_321.jdk/Contents/Home
```

• GIT, GITHUB

Git is a tool for managing project source code history and is a distributed version control system. Git is one of the most extensively used version control systems on the market today. Git can be installed on a Mac in a variety of methods. I installed it with brew.

Installation:

```
$ brew install git
```

We can check the version of the git using the command

```
(base) amithkonda@Amiths-MacBook-Air ~ % git --version  
git version 2.34.1
```

Configuration:

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

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

GitHub is an Internet hosting service for Git-based software development and version control. It includes Git's distributed version control and source code management (SCM) features, as well as its own.

We create a repository in the GitHub and add the local remote repository to that repository using the following commands.

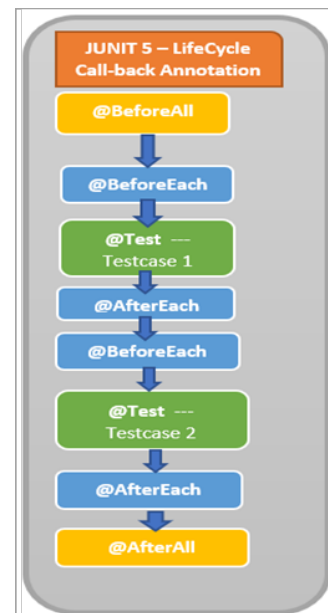
- `git init`
- `git add .`
- `git commit -m "your message"`
- `git remote add origin "GitHub repository link"`
- `git push -u origin master`

TESTING AND BUILDING

- **Junit:**

JUnit is a Java programming language unit testing framework. A JUnit test fixture is a Java object that allows many test cases to be written. The `@Test` annotation is required for test methods. If necessary, the `@BeforeEach` (or `@AfterEach`) and `@BeforeAll` (or `@AfterAll`) annotations can be used to define a method that runs before (or after) each (or all) of the test methods.

```
± Amith *  
public class CalculateTest {  
    8 usages  
    private static final double DELTA = 1e-15;  
    8 usages  
    Calculate calculator = new Calculate();  
    ± Amith *  
    @Test  
    public void square_root_test() {  
        double actual=calculator.root( number: 9.0);  
        double exp=3.0;  
        assertEquals(actual,exp, DELTA);  
    }  
    ± Amith *  
    @Test  
    public void factorial_test() {  
        double actual=calculator.factorial( number: 4);  
        double exp=24;  
        assertEquals(actual,exp, DELTA);  
    }  
    ± Amith *  
    @Test  
    public void lne_test() {  
        double actual=calculator.log( number: 300);  
        double exp=5.703782474656201;  
        assertEquals(actual,exp, DELTA);  
    }  
    ± Amith *  
    @Test  
    public void powerofno_test() {  
        double actual=calculator.power(3.0,3.0);  
        double exp=27.0;  
        assertEquals(actual,exp, DELTA);  
    }  
}
```



- **Maven:**

Maven is a tool for managing and comprehending project development. Builds, dependency management, documentation creation, site publication, and distribution publication are all controlled by the `pom.xml` declarative file, which is based on the concept of a project object model. Plugins for Maven allow it to use a variety of other development tools for reporting and the build process.

Installation:

\$brew install maven

Check Version:

\$mvn -version

Information about maven:

\$brew info maven

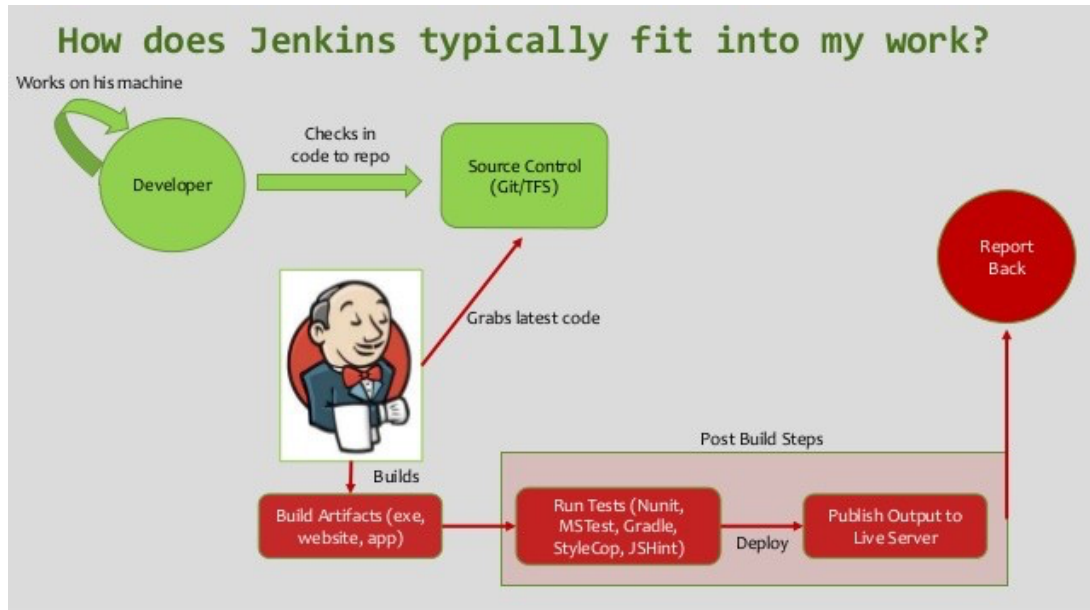
Run maven commands from project root directory.

\$ mvn clean \$ mvn compile \$ mvn test \$ mvn install

CONTINUOUS INTEGRATION

- **Definition:**

When a change is committed to the version control system, it is a process of automating the build and testing of code. Continuous integration's main goals are to detect and fix defects faster, enhance software quality, and speed up the validation and release of new software upgrades.



- **Jenkins:**

Jenkins is a Java-based open source automation platform with plugins designed for Continuous Integration. Jenkins is used to continually create and test your software projects, making it easier for developers to integrate changes to the project and for users to get a fresh build. It also enables you to release your software on a continuous basis by interacting with a variety of testing and deployment technologies.

Installation:

\$brew install Jenkins-lts

Start Jenkins:

\$brew services start Jenkins-lts

Restart Jenkins:

\$brew services restart Jenkins-lts

Stop Jenkins:

\$brew services stop Jenkins-lts

After starting the services, the jenkins server can be accessed in the port localhost:8080. After opening the jenkins server in the browser we need to create a Jenkins user.

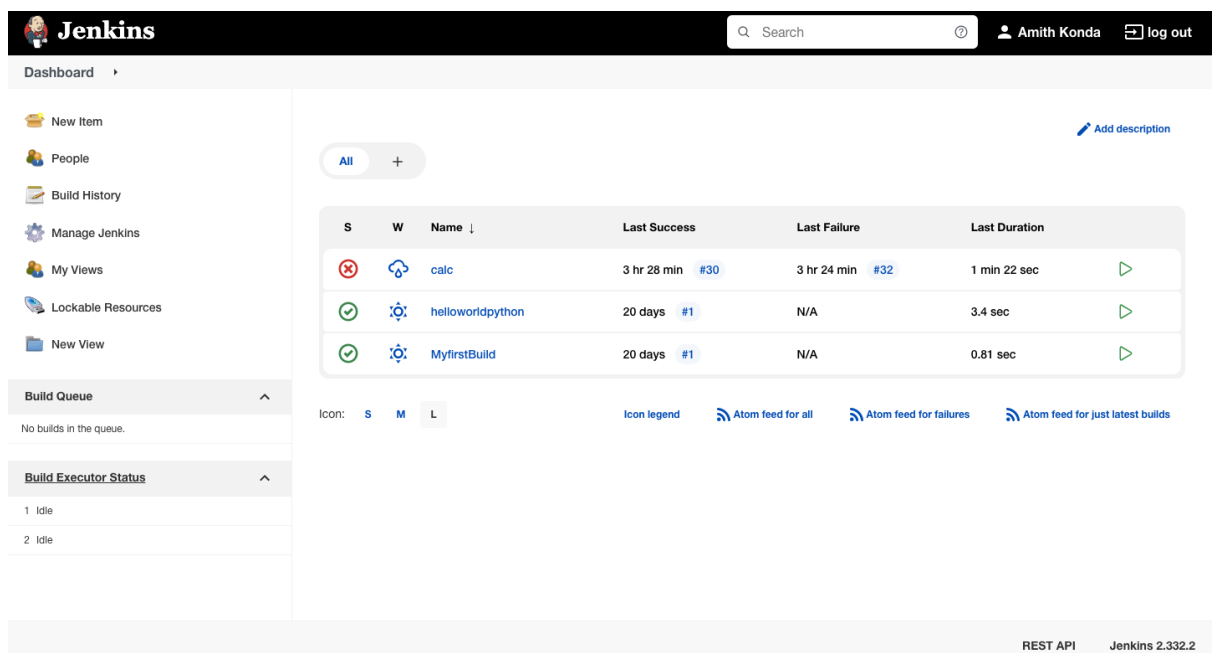
Before that we need to access the default password of Jenkins stored in system using the command
`$ cat /Users/amithkonda/.jenkins/secrets/initialAdminPassword`

We get the default password and use this to login into the jenkins server. After that we need to add the required details to create a user like username, password and email address etc.

The Jenkins dashboard will look like the picture below after we create user.

Here my user name is Amith Konda.

Along with creating user we need to install necessary plugins in jenkins server.



The screenshot shows the Jenkins dashboard interface. At the top, there's a header with the Jenkins logo, a search bar, and the user 'Amith Konda' with a 'log out' button. The left sidebar contains navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. Below these are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing 1 Idle and 2 Idle executors). The main content area displays a table of builds under the 'All' filter. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. There are three builds listed: 'calc' (failed), 'helloworldpython' (succeeded), and 'MyfirstBuild' (succeeded). At the bottom right, there's a footer with 'REST API' and 'Jenkins 2.332.2'.

S	W	Name	Last Success	Last Failure	Last Duration
✖	🔗	calc	3 hr 28 min #30	3 hr 24 min #32	1 min 22 sec
✔	🔗	helloworldpython	20 days #1	N/A	3.4 sec
✔	🔗	MyfirstBuild	20 days #1	N/A	0.81 sec

To create a new project we select New item icon and it will give necessary details to create a project.

After creating a project, we used build command to run the project. We can see the build history and the list of projects which are created before as shown in above figure.

CONTAINERIZATION

- **Docker:**

Docker is a free and open platform for building, delivering, and operating apps. Docker allows you to decouple your apps from your infrastructure, allowing you to swiftly release software. You can manage your infrastructure the same way you control your applications with Docker. Docker allows you to bundle and run an application in a container, which is a loosely isolated environment. Because of the isolation and security, you can operate multiple containers on the same host at the same time.

Installation:

- [Download Docker](#) (Click the link)
- Open the DMG file, and drag-and-drop Docker into your Applications folder.
- Open Docker app to start Docker.
- Docker is running and available, as indicated by the whale in your status bar.
- Docker provides instructions for executing common tasks as well as access to resources.
- The whale in the status bar can be used to access settings and other options. Make sure you have the most recent version of Docker by selecting About Docker.

Check Version:

```
$ docker --version
```

```
$ docker-compose --version
```

```
$ docker-machine --version
```

Commands:

```
$ docker build -t amith1108/miniprojectspe
```

```
$ docker push amith1108/miniprojectspe
```

```
$ docker pull amith1108/miniprojectspe
```

```
$ docker run -it amith1108/miniprojectspe
```

- **Docker File:**

We use docker file to add the jar file of the main java project into the new image from the base image which has jdk8 installed in it.

```
FROM openjdk:8
MAINTAINER Amith KONDA kondaamithsai8@gmail.com
COPY ./target/Calc-Version1-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java", "-jar", "Calc-Version1-jar-with-dependencies.jar"]
```

```
(base) amithkonda@Amiths-MacBook-Air miniproject % docker image build -t amith1108/miniprojectspe:latest .
[+] Building 23.0s (8/8) FINISHED
=> [internal] load build definition from Dockerfile 0.2s
=> => transferring dockerfile: 242B 0.1s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/openjdk:8 4.9s
=> [auth] library/openjdk:pull token for registry-1.docker.io 0.0s
=> [internal] load build context 0.5s
=> => transferring context: 2.03MB 0.5s
=> [1/3] FROM docker.io/library/openjdk:8@sha256:c498405e558f67cea05d04d63c5daf930d0c1fbbf451c8f04f0e2321d740cb83 16.0s
=> => resolve docker.io/library/openjdk:8@sha256:c498405e558f67cea05d04d63c5daf930d0c1fbbf451c8f04f0e2321d740cb83 0.0s
=> => sha256:18f8e41f975e4f0d5ab010b5aab624382b335e6896cc600b1f28287a68a848e 7.81kB / 7.81kB 0.0s
=> => sha256:c498405e558f67cea05d04d63c5daf930d0c1fbbf451c8f04f0e2321d740cb83 1.04kB / 1.04kB 0.0s
=> => sha256:677668e1de865f81f9c18afa6a627033759cc9668b20d8399e437458f5a1d362 1.79kB / 1.79kB 0.0s
=> => sha256:356119a3f4b26e8b6acccb8556234c8eeafa7e179c65325e5170d5e0be329de 106.07MB / 106.07MB 9.3s
=> => sha256:a69de18ffdd4cd971d0a10fde8986a1b4c440bb52bc7b8396cf1ebab3f479e46 210B / 210B 0.6s
=> => extracting sha256:a69de18ffdd4cd971d0a10fde8986a1b4c440bb52bc7b8396cf1ebab3f479e46 0.0s
=> => extracting sha256:356119a3f4b26e8b6acccb8556234c8eeafa7e179c65325e5170d5e0be329de 6.0s
=> [2/3] COPY ./target/Calc-Version1-jar-with-dependencies.jar ./ 1.5s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:2a6314944a70879f9b7371e4c86cd036acd5c51ed4ccefd2d9d8d4040cc9b3819 0.0s
=> => naming to docker.io/amith1108/miniprojectspe:latest 0.0s
```

Building docker image

- **Docker hub:**

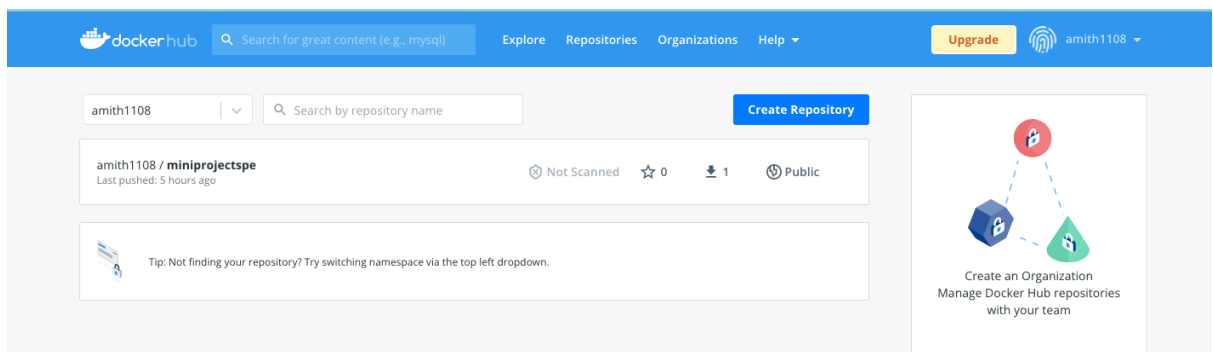
Docker Hub makes it simple for teams to design, manage, and deploy container applications. We can find both official photos generated by companies and customised images created by various users here. We build our own database.

Sign-in into <https://hub.docker.com/account> and create a repository.

How to add the credentials of Docker Hub into Jenkins:

Credentials ->System - > Provide username and password of Docker Hub

-> Provide an ID for this credentials as Docker Hub.

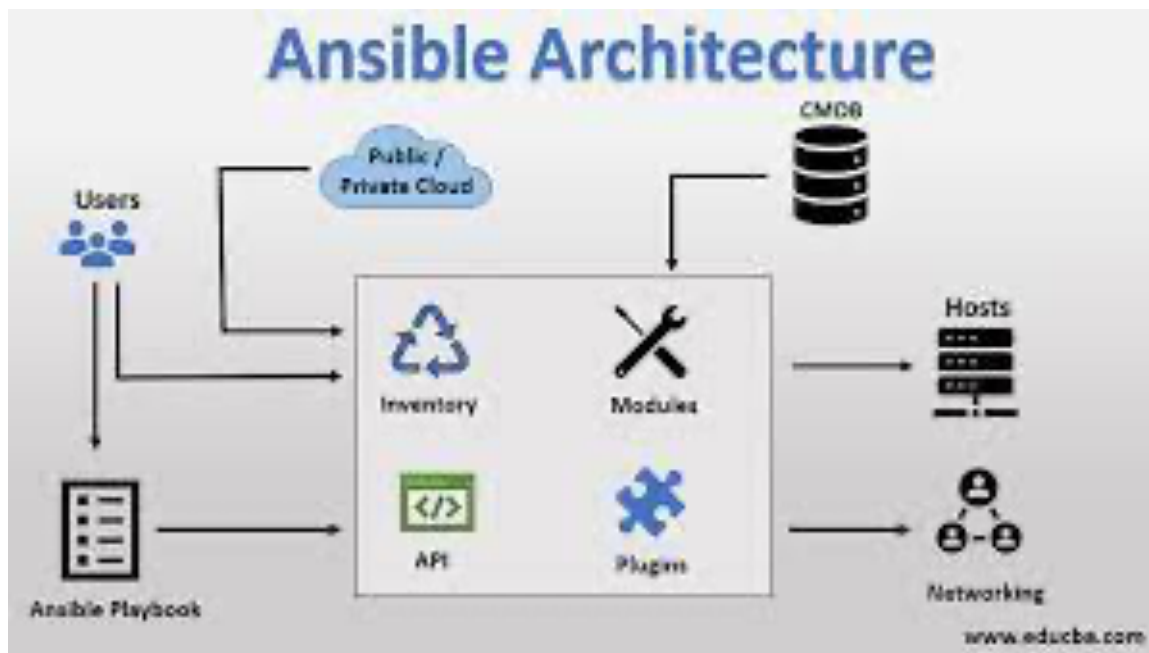


DEPLOYMENT

- **Ansible:**

Ansible is an open source automation platform that includes an Automation Engine for executing Ansible playbooks. Playbooks are defined tasks in which settings and workflows are defined. Ansible uses openssh or winrm to connect to the hosts it controls and run tasks. These are short programmes that are sent to the hosts.

We need to install python and ssh before installing ansible, since ansible is a python library and ssh library is used to communicate with other devices.



Installation:

```
$brew install pyenv
```

```
$pyenv install 3.9.2
```

```
$brew install hudochenkov/sshpas/pass
```

```
$brew install ansible
```

```
(base) amithkonda@Amiths-MacBook-Air miniproject % ansible --version
ansible [core 2.12.4]
  config file = None
  configured module search path = ['/Users/amithkonda/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/Cellar/ansible/5.6.0/libexec/lib/python3.10/site-packages/ansible
  ansible collection location = /Users/amithkonda/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.10.2 (main, Feb 2 2022, 06:19:27) [Clang 13.0.0 (clang-1300.0.29.3)]
  jinja version = 3.1.1
  libyaml = True
```

- **Implementing Playbooks**

Ad hoc commands can perform a single, simple task against a set of targeted hosts as a one-time command. However, learning how to use playbooks to conduct many, sophisticated actions against a group of targeted hosts in a repeatable manner is where Ansible's actual strength lies. A play is a sequence of tasks that are performed against hosts from your inventory in a specific order. A playbook is a text file containing a list of one or more plays that must be performed in a particular order. A playbook is a text file with the yml extension that is written in the YAML format. The playbook uses indentation with space characters to show the structure of its data.

```
---
- name: Pull Docker image of Calculator
  hosts: all
  tasks:
    - name: Pull Cal image
      docker_image:
        name: amith1108/miniprojectspe
        source: pull
```

- **Running Playbooks**

Playbooks are run via the `ansible-playbook` command. The playbook to be run is supplied as a parameter to the command, which is run on the control node:

```
$ ansible-playbook x.yml
```

When you run the playbook, it generates output that shows the play and tasks that are being performed. Each task's outcomes are also reported in the output.

Tasks in Ansible Playbooks are idempotent when you rerun them, so you can execute them as many times as you like. No changes should be done if the targeted managed hosts are already in the correct status.

Syntax Verification:

```
$ ansible-playbook --syntax-check x.yml
```

Executing a dry run:

```
$ ansible-playbook -C x.yml
```

ELK STACK:

Continuous network monitoring gives near-instant input and insight into network performance and interactions.

The abbreviation "ELK" stands for Elasticsearch, Logstash, and Kibana, three open source projects. Elasticsearch is a data analytics and search engine. Logstash is a server-side data processing pipeline that simultaneously ingests data from numerous sources, changes it, and transfers it to a "stash" like Elasticsearch. In Elasticsearch, Kibana allows users to visualise data using charts and graphs.

Logstash configuration file:

```
The bare structure of the configuration file of logstash is input {  
plugin {  
settings  
  
}  
}  
filter {  
plugin {  
settings  
}  
}  
output {  
plugin {  
settings  
}}
```

Kibana is an open source browser-based visualisation application that is primarily used to analyse enormous amounts of logs using line graphs, bar graphs, pie charts, heat maps, region maps, location maps, gauge, goals, timeline, and other visualisations. The graphic makes it simple to predict or see changes in error patterns or other key input source occurrences.

We tell Kibana which Elasticsearch index to look at using the index pattern.

Go to Management Stack Management in Kibana.

Look for Index Patterns in Kibana. Create an index pattern by following the index pattern provided in the logstash configuration file with a *. Select @timestamp as your Time field in the following step. After you've selected Create index pattern, you're ready to begin analysing the data. Investigate the Dashboard and Discover choices.

PIPELINE

Configure Jenkins by installing plugins and make some configuration to run project in automated pipeline manner.

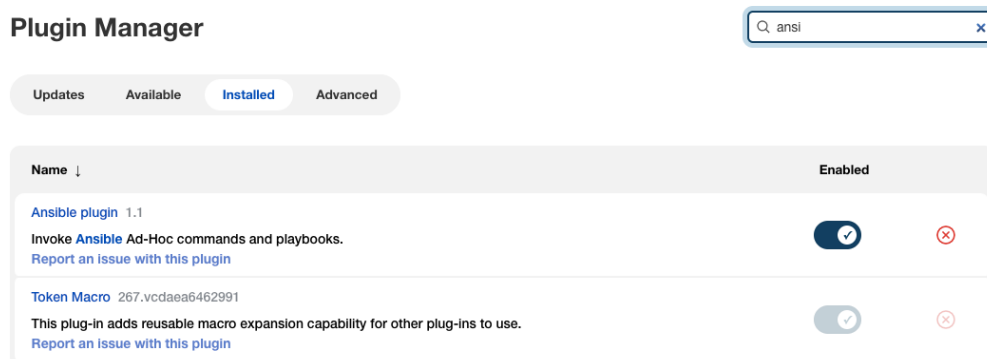
Install Plugins:

If not-vulnerable: Manage Jenkins -> Manage Plugins -> Available -> Filter -> Install without restart

If vulnerable

Download stable plugin file from <https://updates.jenkins-ci.org/download/plugins/> Manage Jenkins -> Manage Plugins -> Advanced -> Upload HPI file -> Install without restart

1. Git
2. GitHub plugin
3. Maven Integration plugin
4. Docker plugin
5. Pipeline
6. Ansible plugin

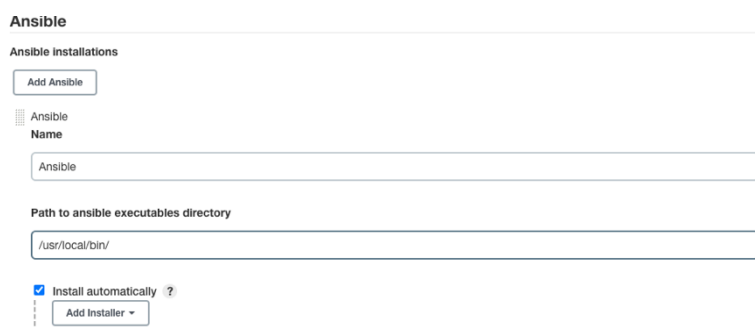


Configure Systems:

Manage Jenkins -> Configure System-> Add Ansible Instances:

Provide name

Provide path - which can be found using \$whereis ansible



Adding the credentials of DockerHub account to Jenkins:

Credentials -> System -> Provide username and password of DockerHub -> Provide an ID for this credentials as DockerHub.

Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	Amith	amith1108/*****

Icon: S M L

Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	 (global)

Create Jenkins Pipeline:

The aim of the pipeline is to trigger whenever a commit happen on GitHub repository, the build should happen using dependencies defined in pom.xml and it should be tested automatically. If test is successful, then it builds an image from Dockerfile and pushes to DockerHub. If this is success, then Ansible is triggered from here, which will deploy the final containerized application to the host node(s).

Create Jenkinsfile -

Jenkins -> New Item -> Enter Item Name-> Pipeline -> OK

Connecting ssh server:

```
(base) amithkonda@Amiths-MacBook-Air miniproject % ssh suchit@172.16.144.254
suchit@172.16.144.254's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-39-generic x86_64)

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

68 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Apr 18 16:02:08 2022 from 172.16.132.43
```

Creating image:

```
(base) amithkonda@Amiths-MacBook-Air miniproject % ansible-playbook dockerdet.yml -i Inventory

PLAY [Pull Docker image of Calculator] *****

TASK [Gathering Facts] *****
ok: [172.16.144.254]

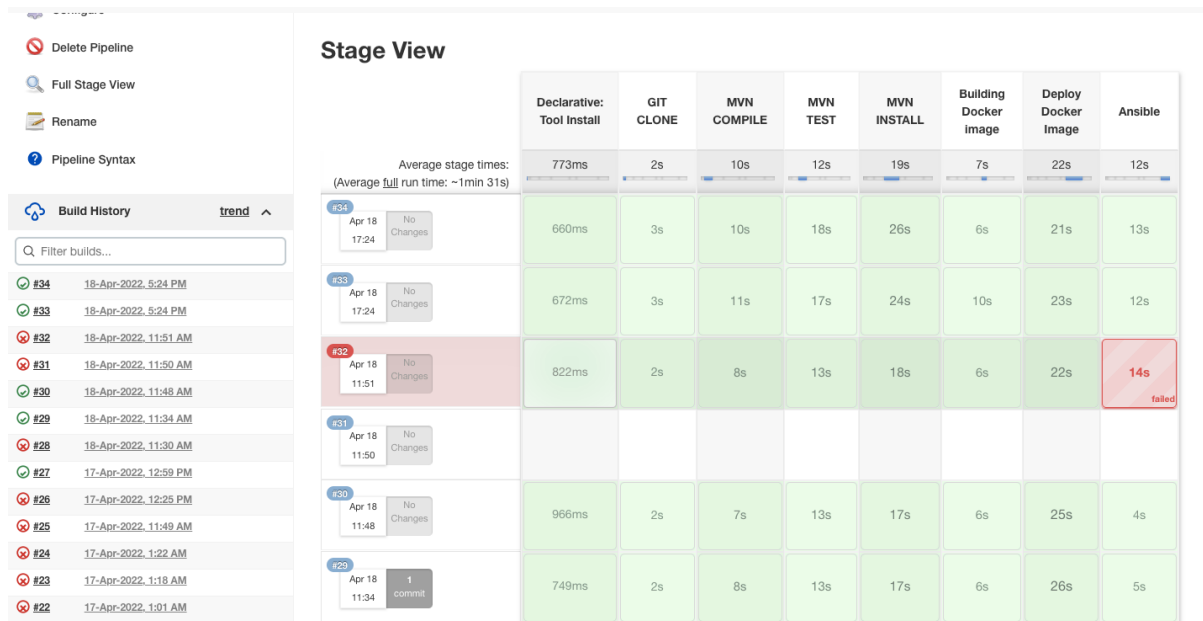
TASK [Pull Cal image] *****
changed: [172.16.144.254]

PLAY RECAP *****
172.16.144.254 : ok=2  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

Viewing images:

```
suchit@suchit-Lenovo-ideapad-330-15IKB:~$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
amith1108/miniprojectspe  latest     2672ce185cfb  5 hours ago  662MB
```

Pipeline Execution:



LINKS

- <https://github.com/Amithsai88/miniprojectspe.git>
- [Docker hub](#)

REFERENCES

- <https://runnable.com/docker/install-docker-on-macos>
- <https://www.jenkins.io/download/lts/macros/>
- https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html
- <https://www.journaldev.com/33645/maven-commands-options-cheat-sheet>

