Project Description
Create a CI/CD Pipeline to convert the legacy development process to a DevOps process.
Background of the problem statement:
A leading US healthcare company, Aetna, with a large IT structure had a 12-week release cycle and their business was impacted due to the legacy process. To gain
true business value through faster feature releases, better service quality, and cost optimization, they wanted to adopt agility in their build and release process.
The objective is to implement iterative deployments, continuous innovation, and automated testing through the assistance of the strategy.

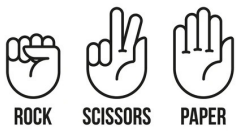
Implementation requirements:

1. Install and configure the Jenkins architecture on AWS instance
2. Use the required plugins to run the build creation on a containerized platform
3. Create and run the Docker image which will have the application artifacts
4. Execute the automated tests on the created build
5. Create your private repository and push the Docker image into the repository
6. Expose the application on the respective ports so that the user can access the deployed application
7. Remove container stack after completing the job

The following tools must be used:
1. EC2
2. Jenkins
3. Docker
4. Git

The following things to be kept in check:
1. You need to document the steps and write the algorithms in them.
2. The submission of your Github repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, executing it, and recording the results.
4. You need to submit the final specification document, which includes:
- *Project and tester details*
- *Concepts used in the project*
- *Links to the GitHub repository to verify the project completion*
- *Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)*

For this project we will be using a simple Rock-paper-scissor web application. Just as a refresher it's a game which is usually played between two parties, in which each player simultaneously forms one of three shapes, rock, paper and scissor.
The game is a zero-sum game, it has only two possible outcomes: a draw, or a win for one player and a loss for the other.



The game algorithm:

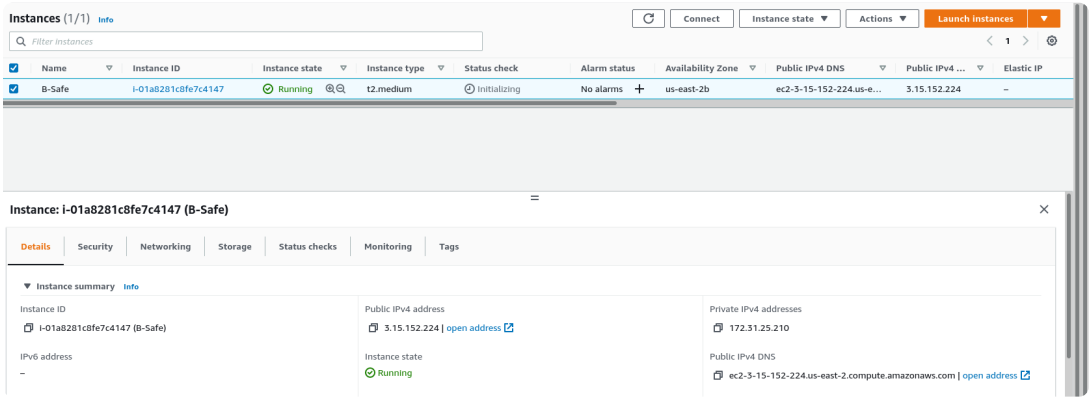
The game uses a simple probability algorithm which counts the previous moves of the opponent to determine if the opponent prefers playing one type of move over the others. Essentially, it keeps a "score" for each of rock, paper, and scissors. After each move, the respective score of the opponent's move is incremented. Our application also displays a summary after each move indicating the scores of win/lose.

Github repo : https://github.com/ik0z/Aetna_webapp.git

Step1 : Setup AWS instance

I prepare a free tier t2.medium AWS instance with Ubuntu 20.04 LTS.
up an appropriate security group to allow traffic for our intended services.

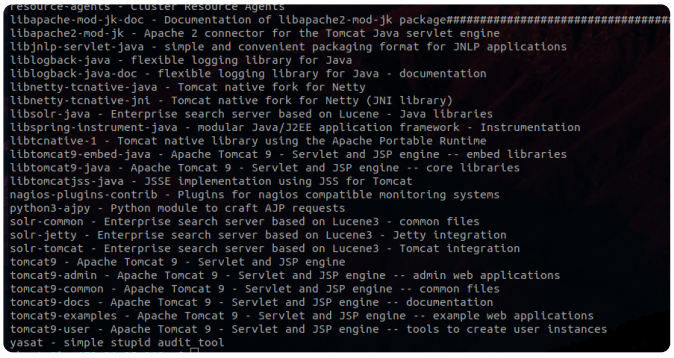
I also create and download a SSH key for accessing the instance SSH. (b-safe.pem)



After connection via ssh then proceed with installing the below :
- *apache tomcat*
- *Docker*
- *Jenkins with necessary plugins*

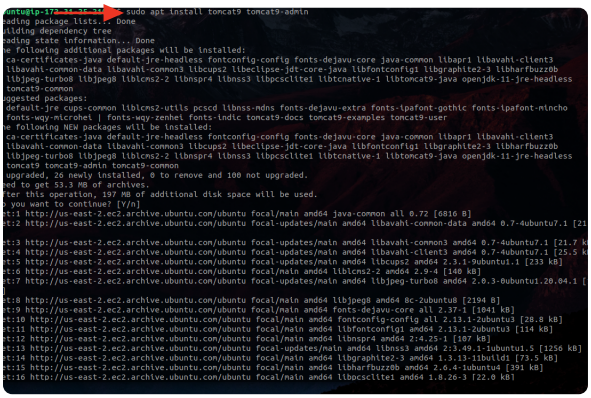
1 - install & setup tomcat :

check tomcat in repo : $ sudo apt-cache search tomcat

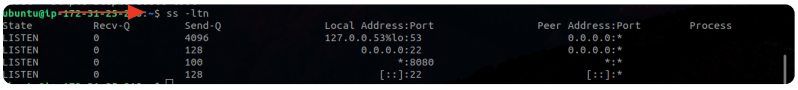


then download the tomcat9 package and the tomcat9 admin package and its dependencies.

$ sudo apt install tomcat9 tomcat9-admin



For verification, type the following ss command, which will show you the 8080 open port number, the default open port reserved for Apache Tomcat Server.
$ ss -ltn

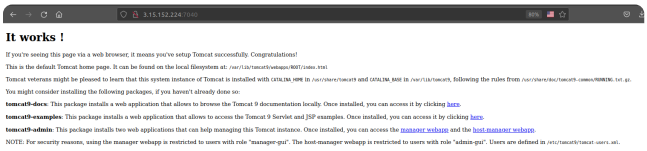


Since I will also install Jenkins on this same instance whose default port is also 8080 we will change our Tomcat port to 7040

I edit the server.xml by via this line
$ sudo nano /etc/tomcat9/server.xml

Save & exit then write this line : $ sudo systemctl restart tomcat9 && sudo systemctl status tomcat9

from the browser open : http://<public ip >:7040



Step 2 : Setup & install docker

first install JDK, set up the JAVA path, install common applications then proceed with download and install Docker

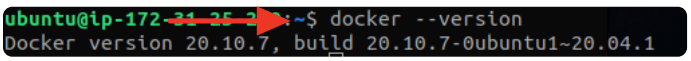
Command :

```
$ sudo apt update && sudo apt install default-jdk
$ export PATH=$PATH:$JAVA_HOME/bin
$ sudo apt-get -y install \
$ apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

$ sudo apt-get install -y docker.io
```

Verify the docker version :



then make ubuntu user to run Docker as admin

```
$ sudo usermod -a -G docker ubuntu
$ sudo chmod 666 /var/run/docker.sock
```

Step 3 : Setup & install Jenkins

run the following commands :

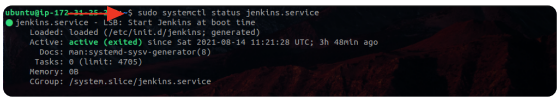
```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

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

$ sudo apt-get update && sudo apt-get install jenkins
```

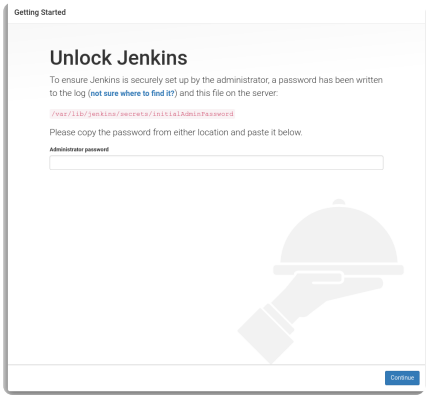
Starting Jenkins and checking the service status :

```
$ sudo systemctl start jenkins && sudo systemctl status jenkins
```

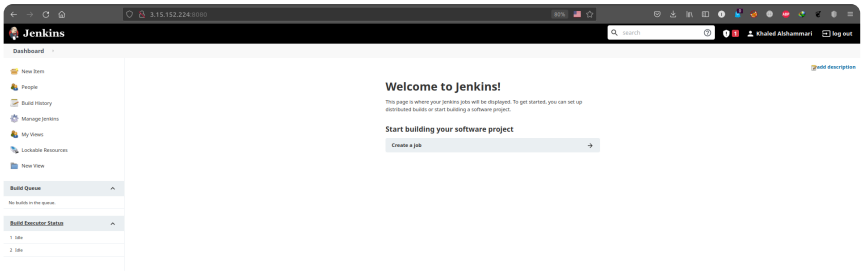


We now make jenkins user(default user Jenkins uses to run job) to run Docker as admin
$ sudo usermod -a -G docker jenkins && sudo chmod 666 /var/run/docker.sock

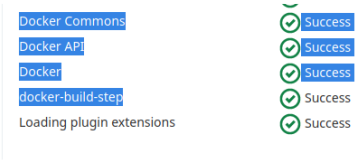
from the browser open : http://<public ip >:8080



after read key via (cat /var/lib...) , I proceed with installation of the default setup and setting up an admin account .
Once everything finished we should see something like below

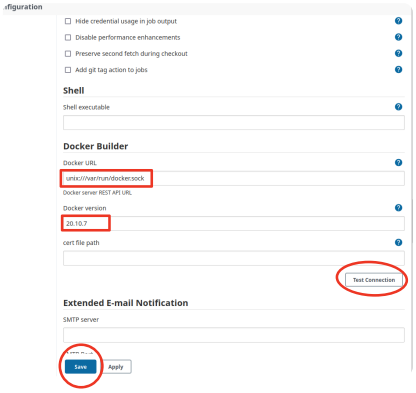


Install docker plugin:

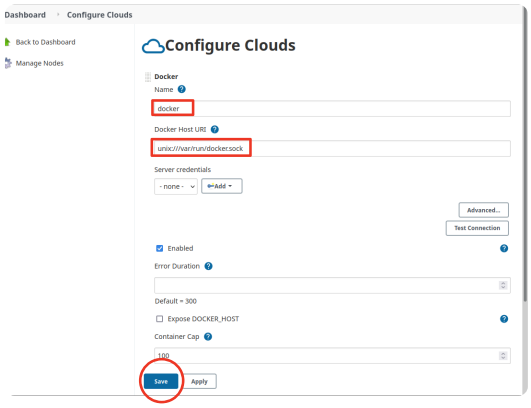


For Jenkins to run local Docker we need to set up the Jenkins Docker integration to allow Jenkins communicate with local Docker.

- navigate to Manage Jenkins > Configure System > Docker Builder
- Enter URL value 'unix:///var/run/docker.sock' then test the connection.



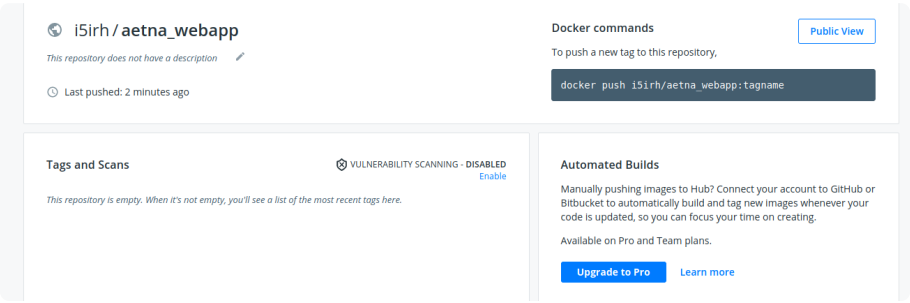
also setup the Docker cloud connector and enable it so that Docker API is correctly configured



Step 4 : Setup Dockerhub & create a job in Jenkins

Setup private repo in dockerhub :

for pushing our web application container after build as per our project requirement.
We login to Docker Hub then create a private repo called 'aetna_webapp'



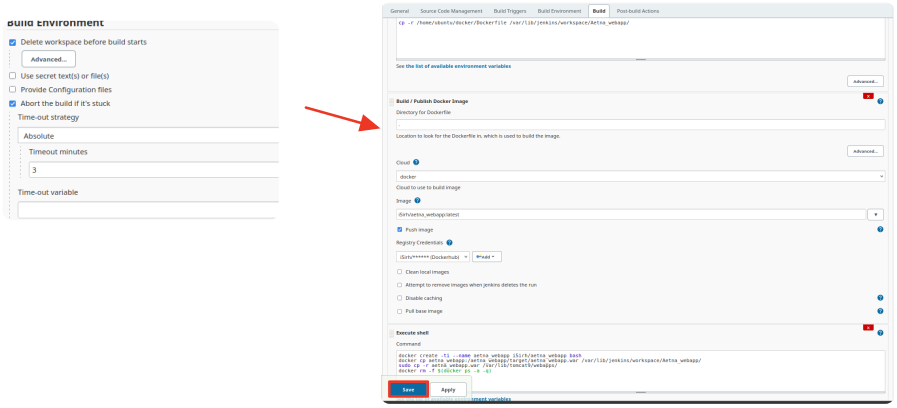
Now we are ready to build our application within the Docker container then deploy using our Jenkins server in automated fashion.

Dockerfile :

```
FROM ubuntu:20.04
RUN apt-get update
RUN apt-get -y upgrade
RUN apt-get install -y default-jdk
RUN export PATH=$PATH:$JAVA_HOME/bin
RUN apt-get -y install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common
RUN apt-get install -y maven
RUN apt-get install -y git
RUN git clone https://github.com/ik0z/Aetna_webapp.git
WORKDIR ./Aetna_webapp
RUN mvn clean install
RUN mvn test
RUN mvn clean package
```

This Dockerfile lists the steps to simply use Ubuntu 20.04 then install all necessary applications for our application build.After install we then run the tests then compile our project to build a WAR for deployment.

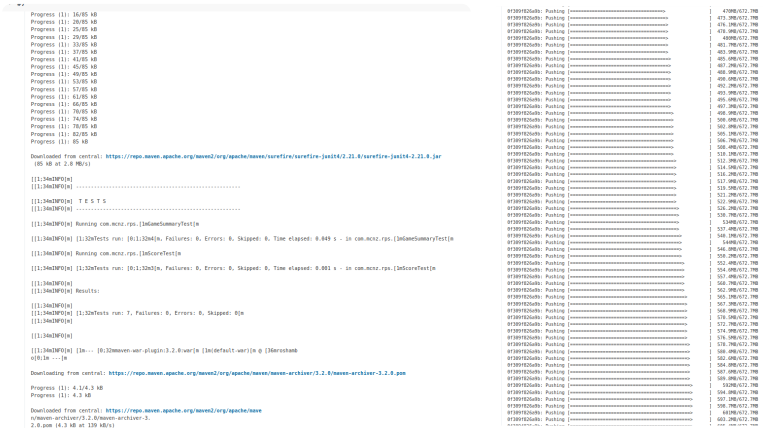
On Jenkins dashboard create a job " freestyle " name " Aetna_webapp "



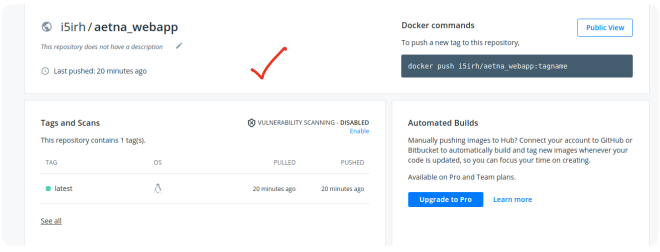
I also set up credentials for our Docker Hub private repo which we've created earlier to push our container after successfully built.

For the 'Build' step we use a mix of 'Execute Shell' and 'Build/Publish Docker image' commands. The reasons are -
- Our previously created Dockerfile is under 'ubuntu' username which 'jenkins' user don't have access to. So we use the SHELL command to ensure we copy this Dockerfile to a workspace which is accessible by user 'jenkins' when running the job.
- Another important step is to make our 'jenkins' user added in system sudoers in order to deploy the build to our tomcat web server

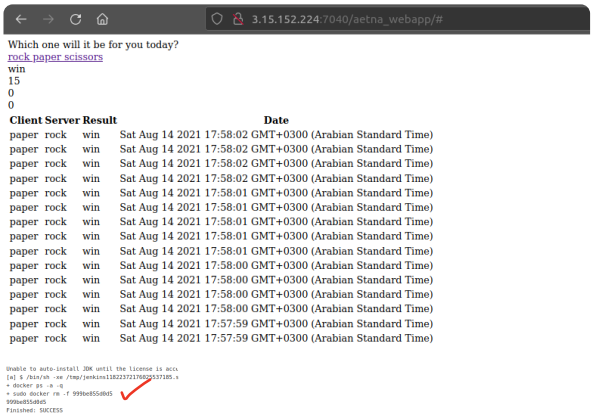
Run build a job :



successful push to our Docker Hub repo.



successful deployment of our web app so that user can access the deployed application



Project URL : http://3.15.152.224:7040/aetna_webapp/#