

## **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

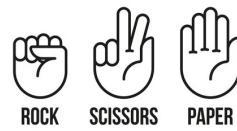
### **The following tools must be used:**

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.

## 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)

The screenshot shows the AWS Management Console. At the top, there's a navigation bar with tabs like 'Instances (1/1)', 'Info', 'Actions', and 'Launch instances'. Below the navigation bar is a search bar labeled 'Filter instances'. A table lists one instance: 'B-Safe' (Instance ID: i-01a8281c8fe7c4147), which is 'Running' and 't2.medium'. The status check is 'Initializing' and there are 'No alarms'. The availability zone is 'us-east-2b' with a public IPv4 DNS of 'ec2-3-15-152-224.us-east-2.compute.amazonaws.com' and a public IPv4 IP of '3.15.152.224'. An 'Elastic IP' is listed as '-'. Below the table is a large modal window titled 'Instance: i-01a8281c8fe7c4147 (B-Safe)'. The modal has tabs for 'Details', 'Security', 'Networking', 'Storage', 'Status checks', 'Monitoring', and 'Tags'. The 'Details' tab is selected. It shows the instance summary with fields for 'Instance ID' (i-01a8281c8fe7c4147 (B-Safe)), 'Public IPv4 address' (3.15.152.224), 'Private IPv4 addresses' (172.31.25.210), 'Instance state' (Running), 'Public IPv4 DNS' (ec2-3-15-152-224.us-east-2.compute.amazonaws.com), and 'Public IPv4 IP' (3.15.152.224).

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

```
resource-agents - Cluster Resource Agents
libapache-mod-jk-doc - Documentation of libapache2-mod-jk package#####
libapache2-mod-jk - Apache 2 connector for the Tomcat Java servlet engine
libjnpnlp-service-java - simple and convenient packaging format for JNLP applications
liblogback-java - flexible logging library for Java
liblogback-java-doc - flexible logging library for Java - documentation
libnetty-tcnative-java - Tomcat native fork for Netty
libnetty-tcnative-jni - Tomcat native fork for Netty (JNI library)
lucene-solr - Enterprise search server based on Lucene - Java libraries
libspring-instrument-java - modular Java/J2EE application framework - Instrumentation
libtcnative-1 - Tomcat native library using the Apache Portable Runtime
libtomcat9-embed-java - Apache Tomcat 9 - Servlet and JSP engine -- embed libraries
libtomcat9-java - Apache Tomcat 9 - Servlet and JSP engine -- core libraries
libtomcatjss-java - JSSE Implementation using JSF for Tomcat
nagios-plugins-contrib - Plugins for nagios compatible monitoring systems
python3-ajpy - Python module to craft AJP requests
solr-common - Enterprise search server based on Lucene3 - common files
solr-jetty - Enterprise search server based on Lucene3 - Jetty integration
solr-xml - Enterprise search server based on Lucene3 - Tomcat integration
tomcat9 - Apache Tomcat 9 - Servlet and JSP engine
tomcat9-admin - Apache Tomcat 9 - Servlet and JSP engine -- admin web applications
tomcat9-common - Apache Tomcat 9 - Servlet and JSP engine -- common files
tomcat9-docs - Apache Tomcat 9 - Servlet and JSP engine -- documentation
tomcat9-examples - Apache Tomcat 9 - Servlet and JSP engine -- example web applications
tomcat9-user - Apache Tomcat 9 - Servlet and JSP engine -- tools to create user instances
yasat - simple stupid audit tool
```

*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

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port	Process
LISTEN	0	4096	127.0.0.53:53	0.0.0.1:*	
LISTEN	0	128	0.0.0.0:22	0.0.0.1:*	
LISTEN	0	100	*	*:8080	
LISTEN	0	128	[::]:22	[::]:*	

*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
```

```
<!-- A "Server" is a collection of one or more "Connectors", that share  
a single "Container". Note: A "Service" is not itself a "Container",  
but rather a collection of "Server"s, which "Server"s "Container"s at this level.  
Documentation at /docs/config/service.html -->  
  
<Service name="Catalina">  
  
    <!-- The connectors can use a shared executor, you can define one or more named thread pools-->  
    <!-->  
    <Executor name="tomcatThreadPool" coreThreads="100" maxThreads="150" minPartitions="4"/>  
    <br>  
  
    <!-- A "Connector" represents an endpoint by which requests are received  
    and responses are returned. Documentation at:  
    the Java EE specification document:http://java.sun.com/javaee/5/docs/api/jsp.html  
    and the Tomcat documentation:http://tomcat.apache.org/tomcat-6.0-doc/  
    Define a standard HTTP/1.1 Connector on port 8088-->  
  
    <Connector port="8088" protocol="HTTP/1.1"  
        connectionTimeout="20000"  
        redirectPort="8443" />  
    <!-- A "Connector" using the shared thread pool-->  
    <!-->  
    <Connector executor="tomcatThreadPool"  
        ports="8009" protocol="HTTP/1.1"  
        connectionTimeout="20000"  
        redirectPort="8443" />  
    <br>  
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443  
    Documentation at /docs/config/ssl.html -->  
    <!-->  
    <Connector port="8443" protocol="HTTP/1.1"  
        SSLEnabled="true"  
        keystoreType="JKS"  
        keystoreFile="conf/keystore"  
        keyAlias="tomcat"  
        defaultContainer="DefaultContainer"/>  
    <br>
```

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

```
tomcat9.service - Apache Tomcat 9 Web Application Server
   Loaded: loaded (/etc/systemd/system/tomcat9.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-08-14 11:21:25 UTC; 4h 35min ago
     Docs: https://tomcat.apache.org/tomcat-9.0-doc/index.html
      Tasks: 36 (limit: 4705)
     Memory: 322.8K
    Corrupt: 0 B
      CPU: 0.000 CPU(s) since start
          └─/usr/local/libexec/tomcat9.service
              ├─└─/usr/lib/jvm/default-java/bin/java -Djava.util.logging.config.file=/var/lib/tomcat9/conf/logging.properties
```

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 :

```
ubuntu@ip-172-21-25-2:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu1~20.04.1
```

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
```

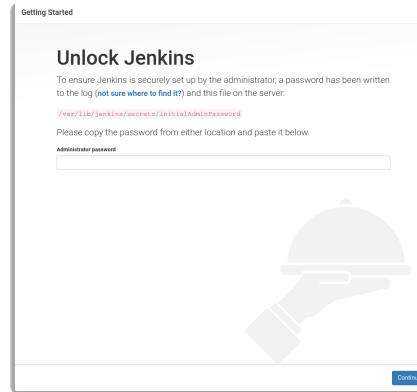
Starting Jenkins and checking the service status :

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

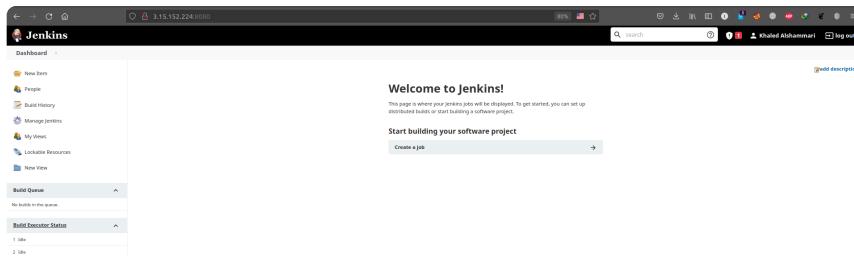
```
ubuntu@ip-172-21-25-2:~$ sudo systemctl status jenkins.service
● jenkins.service - Job start Jenkins at boot time
   Loaded: loaded (/etc/systemd/system/jenkins.service; generated)
   Active: active (exited) since Sat 2021-08-14 11:21:28 UTC; 3h 48min ago
     Tasks: 0 (littles: 0)
    Memory: 0B
      CPU: 0
     CGroup: /system.slice/jenkins.service
```

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:

Docker Commons  
Docker API  
Docker  
docker-build-step  
Loading plugin extensions

Success  
Success  
Success  
Success  
Success

**Configuration**

- Hide credential usage in job output
- Disable performance enhancements
- Preserve second fetch during checkout
- Add git tag action to jobs

**Shell**  
Shell executable

**Docker Builder**

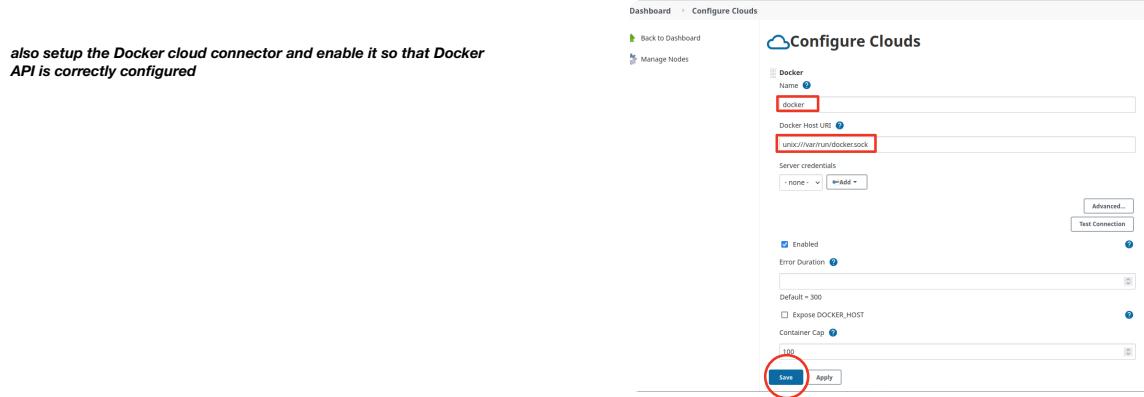
Docker URL: `unix:///var/run/docker.sock`  
Docker server REST API URL:  
Docker version: `20.10.7`  
cert file path

**Extended E-mail Notification**  
SMTP server

**Buttons:** Save (highlighted), Apply

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.



#### 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'

This repository does not have a description.

Last pushed: 2 minutes ago

Docker commands

To push a new tag to this repository.

`docker push i5irh/aetna_webapp:tagname`

Tags and Scans

VULNERABILITY SCANNING - DISABLED

Enable

This repository is empty. When it's not empty, you'll see a list of the most recent tags here.

Automated Builds

Manually pushing Images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

[Upgrade to Pro](#) [Learn more](#)

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 all tests then compile our project to build a WAR for deployment.

On Jenkins dashboard create a job “ freestyle “ name “ Aetna\_webapp “

**Build environment**

- Delete workspace before build starts
  - Advanced...
- Use secret text(s) or file(s)
- Provide Configuration files
- Abort the build if it's stuck

Time-out strategy

Absolute

Timeout minutes

Time-out variable

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

```
cp -r /home/ubuntu/docker/Dockerfile ./var/lib/jenkins/workspace/Arteson_webapp/
```

See the list of available environment variables

**Build / Publish Docker Image**

Directory for Dockerfile

Location to look for the Dockerfile in, which is used to build the image.

Cloud

Docker

Cloud config to build image

Image

(Arteson, webapp:latest)

Push image

Registry Credentials

(git\*\*\*\*\*/DockerHub)

Clean local images

Attempt to remove images when Jenkins deletes the run

Disable caching

Pull base image

Execute shell

Command

```
docker create -t -n arteson_webapp $(ls /tmp/Arteson_webapp)
curl -X POST http://127.0.0.1:8080/computer/$(id -u)/start
curl -X POST http://127.0.0.1:8080/computer/$(id -u)/stop
sudo cp -r /tmp/Arteson_webapp /var/lib/jenkins/workspace/Arteson_webapp/
rm -rf /tmp/Arteson_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*

 **i5irh/aetna\_webapp**

This repository does not have a description 

Last pushed: 20 minutes ago 

---

Tags and Scans	VULNERABILITY SCANNING - DISABLED		
This repository contains 1 tag(s):	 <a href="#">Enable</a>		
TAG	OS	PULLED	PUSHED
 latest		20 minutes ago	20 minutes ago

[See all](#)

---

**Docker commands**

To push a new tag to this repository,

```
docker push i5irh/aetna_webapp:tagname
```

[Public View](#)

---

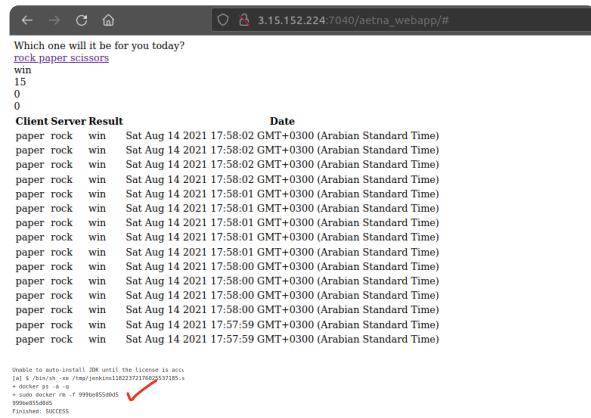
**Automated Builds**

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

[Upgrade to Pro](#) [Learn more](#)

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



Which one will it be for you today?

Rock paper scissors  
win  
15  
0  
0

Client Server Result		Date	
paper	rock	win	Sat Aug 14 2021 17:58:02 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:02 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:02 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:02 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:01 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:00 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:00 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:58:00 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:57:59 GMT+0300 (Arabian Standard Time)
paper	rock	win	Sat Aug 14 2021 17:57:59 GMT+0300 (Arabian Standard Time)

```
unable to auto-install DK until the license is set.
[4] $ ./bin/rsh -w /tmp/jash-test182237219526337185.r
+ docker ps -a -q
+ sudo docker rm -f 999eb855d8d5
999eb855d8d5
Finished: SUCCESS
```

Project URL : [http://3.15.152.224:7040/aetna\\_webapp/#](http://3.15.152.224:7040/aetna_webapp/#)