Introduction to Project

Waterfall method, which is most dominantly used by many software developers, especially in old organizations has many disadvantages such as: -

- The waterfall model doesn't support making changes.
- It can invalidate the work you've previously accomplished.
- This method excludes end-users and clients.
- It delays testing until after the completion of the project.
- The waterfall model can promote longer delivery times.
- It typically works better for small projects.
- Working models aren't available until the latter stages of a project.

Agile methodology offers many benefits over waterfall method. Customer satisfaction is rapid, continuous development and delivery of useful software. Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools. Product is developed fast and frequently delivered (weeks rather than months.) It offers daily and close cooperation between business, people, and developers.

In an agile software development environment, the working model and the operations need to be super flexible to the ever-changing needs of the company.

DevOps – A combination of tools and practices aid in software development along with IT operations and go hand-in-hand. This cross functional working mode aims in minimizing the duration of the system's development life cycle and provides continuous deployment and delivery.

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.

DevOps enables formerly siloed roles—development, IT operations, quality engineering and security—to coordinate and collaborate to produce better, more reliable products. 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.

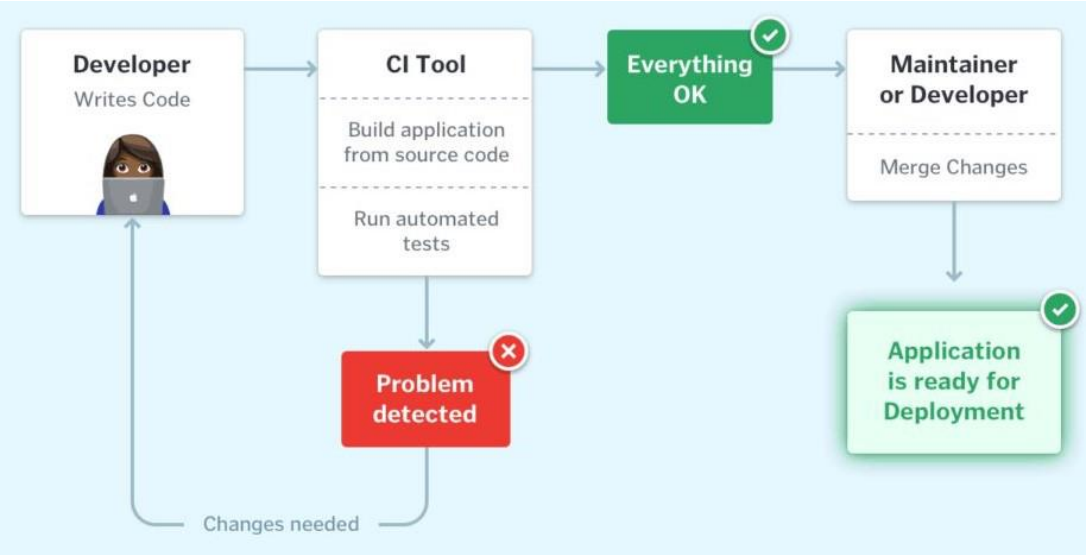
Teams that adopt DevOps culture, practices and tools become high performing, building better products faster for greater customer satisfaction. This improved collaboration and productivity is also integral to achieving business goals like these:

- Accelerating time to market
- Adapting to the market and competition
- Maintaining system stability and reliability
- Improving the mean time to recovery

CI or **Continuous Integration** is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently into the central code repository (Github or Stash). Then there are automated tools that build the newly committed code and do a code review, etc as required upon integration.
The key goals of Continuous Integration are to find and address bugs quicker, make the process of integrating code across a team of developers easier, improve software quality and reduce the time it takes to release new feature updates. Some popular CI tools are Jenkins, TeamCity, and Bamboo.

How CI Works?

Below is a pictorial representation of a CI pipeline- the workflow from developers checking in their code to its automated build, test, and final notification of the build status.



Once the developer commits their code to a version control system like Git, it triggers the CI pipeline which fetches the changes and runs automated build and unit tests. Based on the status of the step, the server then notifies the concerned developer whether the integration of the new code to the existing code base was a success or a failure.

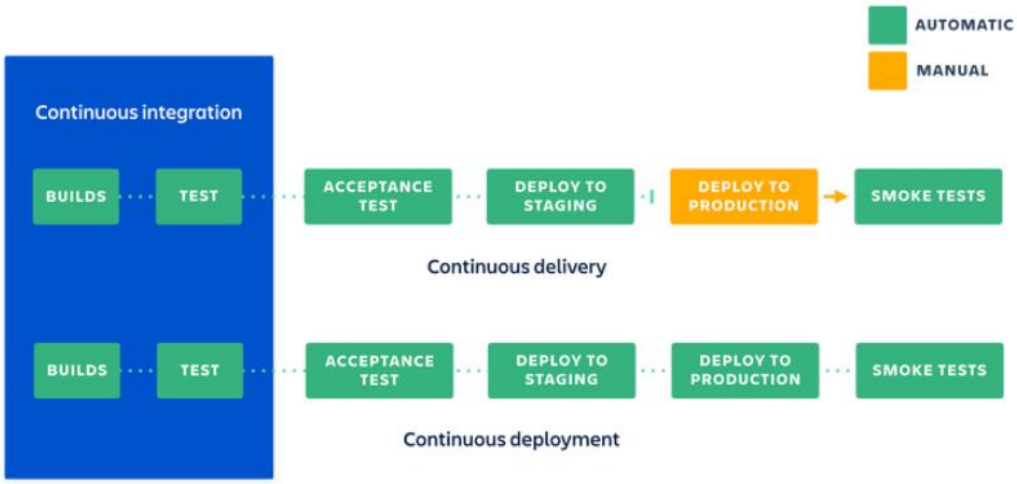
This helps in finding and addressing the bugs much quickly, makes the team more productive by freeing the developers from manual tasks, and helps teams deliver updates to their customers more frequently. It has been found that integrating the entire development cycle can reduce the developer's time involved by ~25 – 30%.

CD or **Continuous Delivery** is carried out after Continuous Integration to make sure that we can release new changes to our customers quickly in an error-free way. This includes running integration and regression tests in the staging area (like the production environment) so that the final release is not broken in production. It ensures to automate the release process so that we always have a release-ready product, and we can deploy our application at any point in time.

Continuous Delivery automates the entire software release process. The final decision to deploy to a live production environment can be triggered by the developer/project lead as required. Some popular CD tools are AWS CodeDeploy, Jenkins, and GitLab.

How CI and CD work together?

The below image describes how Continuous Integration combined with Continuous Delivery helps quicken the software delivery process with lower risks and improved quality.



CI / CD workflow

Incorporating CI/CD into your organization's development process reduces the number of non-critical defects in your backlog. These small defects are detected prior to production and fixed before being released to end-users. The benefits of solving non-critical issues ahead of time are many.

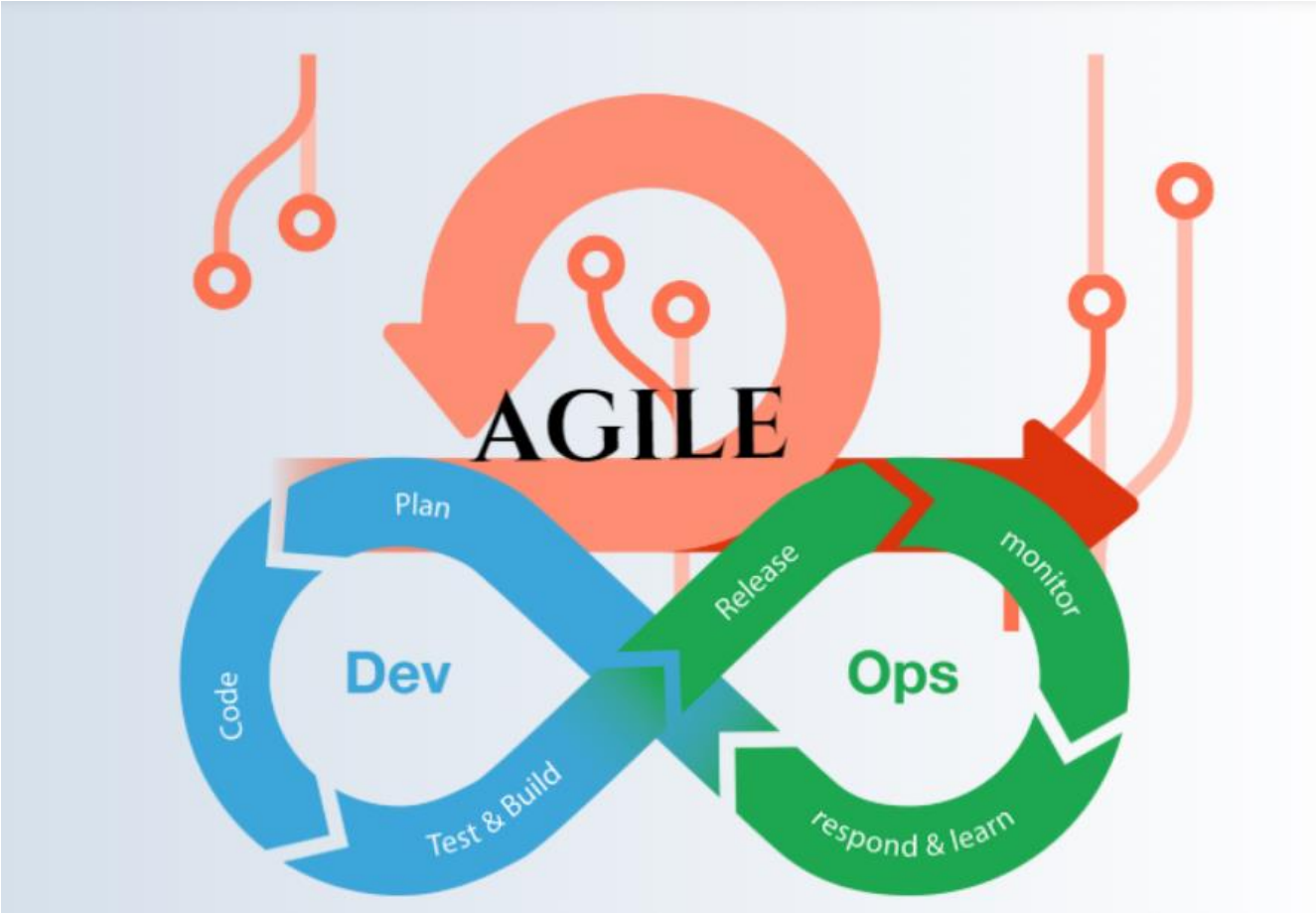
In DevOps, automation means eliminating the need for human engineers to intervene manually to facilitate DevOps practices.

Agile and DevOps, work towards a shared goal, i.e., 'Enhancing Business Productivity.
Agile and DevOps together execute the lean approach on a huge scale, which is evident through their communication process.

Agile, along with DevOps, has a collaborative working style, irrespective of the method implemented.

Both the methodologies rely on continual feedback and routine updates about the work progress from internal and external stakeholders.

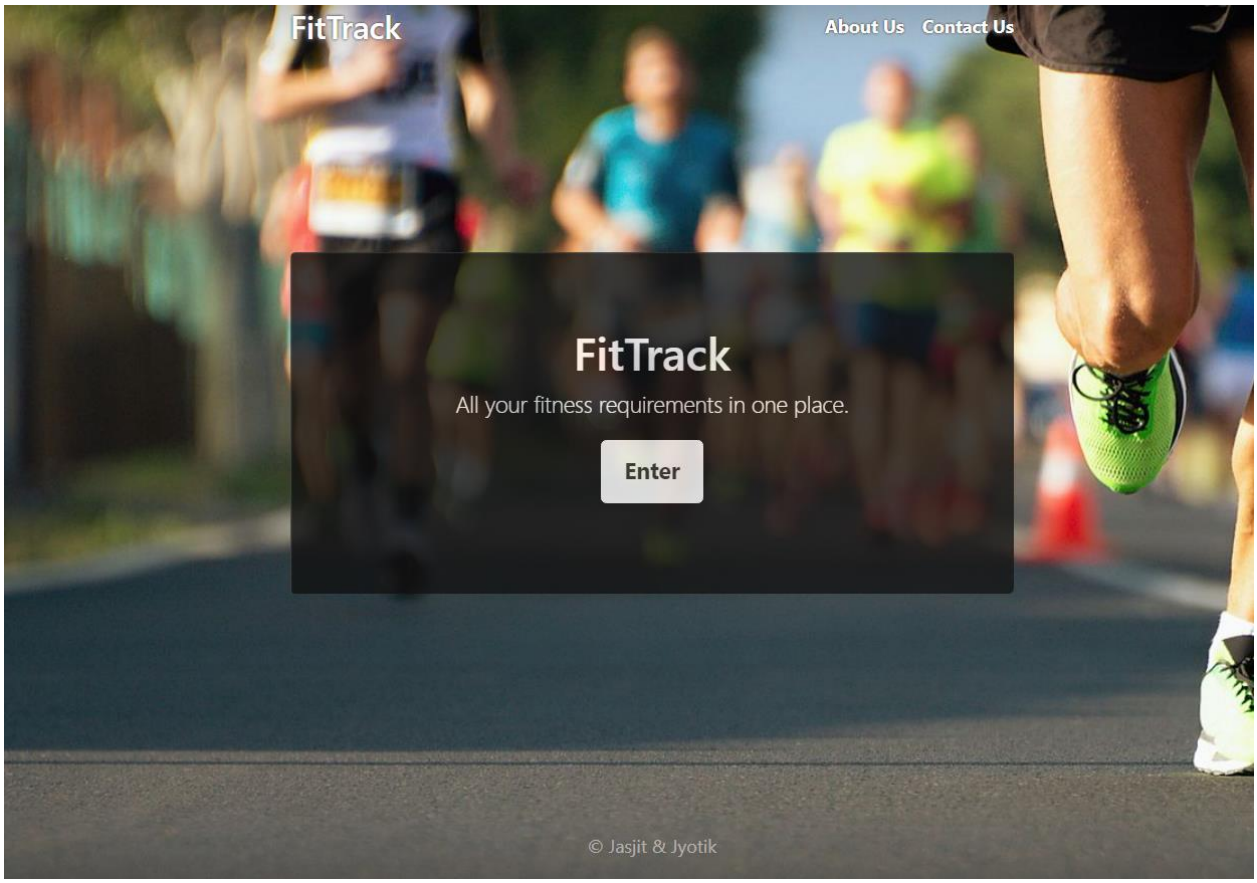
Both Agile and DevOps focus on developing the product at a fast pace by keeping smaller teams and using a risk-free approach. Both methods adapt to the business requirements and continually improve the products to fulfil customer expectations.



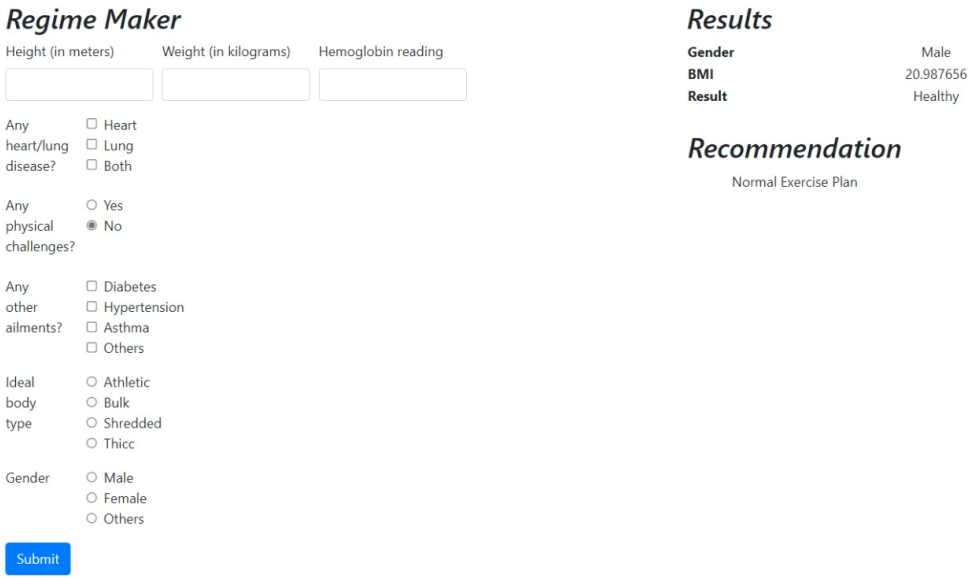
To demonstrate the concept of CI/CD and DevSecOps practice, our project uses a .war file (Java Web Application), where it is pushed into a github repository that automatically triggers a webhook everytime any change is pushed. The webhook on the other side is connected to an Amazon EC2 server that runs Jenkins. On Jenkins, a pipeline is configured with which the code is built, test cases are run, and the packaged war is finally deployed to a docker host and is available live to be accessed by users on internet.

A) The source program

For the simplicity of understanding and implementing concept of pipeline, we've made a simple java web application that based on inputs entered by users suggests them an exercise plan and calculates their Body Mass Index. It contains an interactive UI, and few web pages (index page, home page, about & a contact us page)



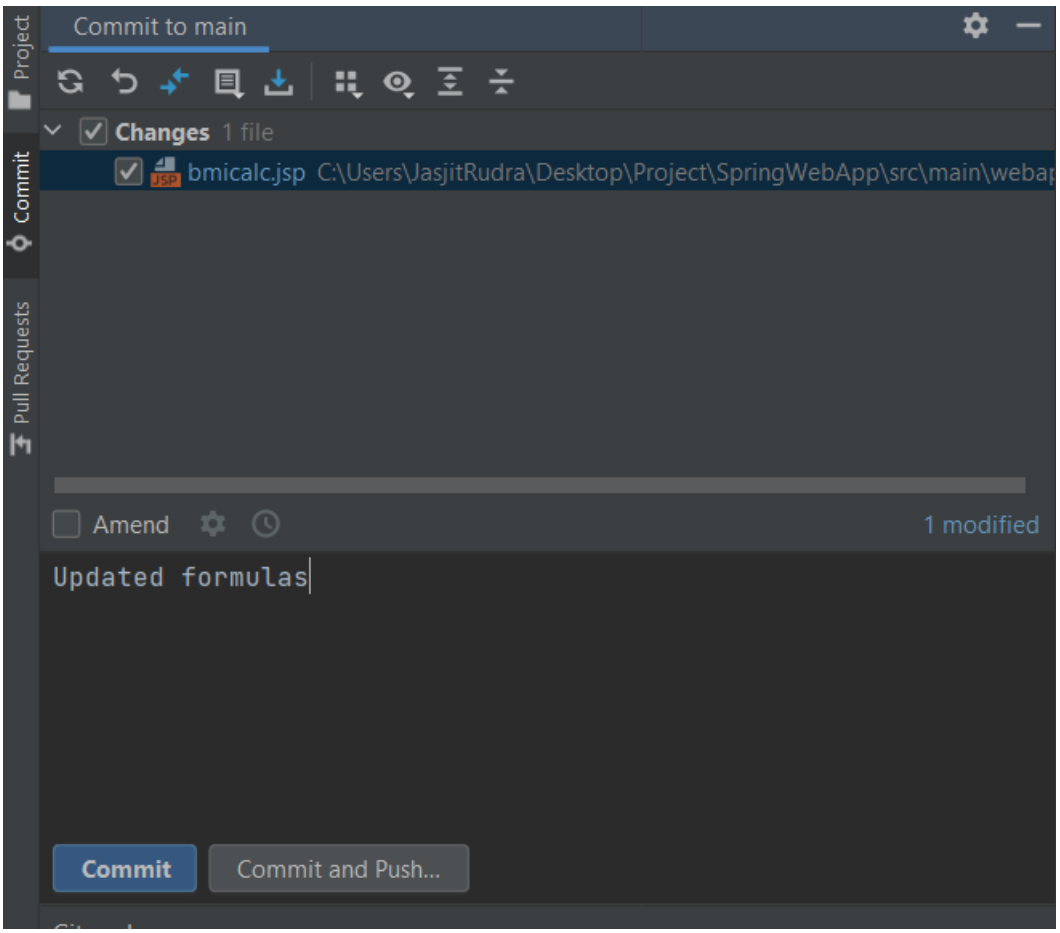
Index Page



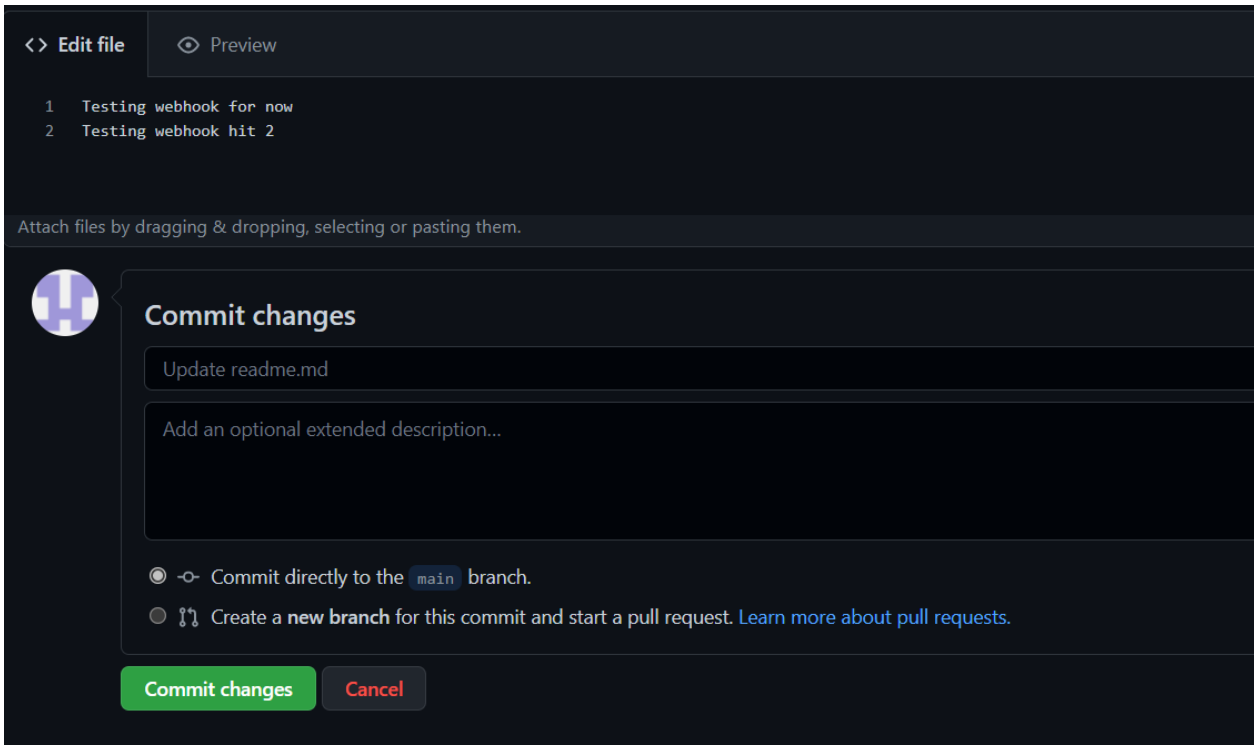
Home Page (Calculates BMI and Recommends Exercise Plan)

B) Git

Git commands were used to push code into a github repository. It is vital for any agile project to have a centralized repository. On github, we have the ability to use webhooks that performs certain actions whenever any new change is made in code.



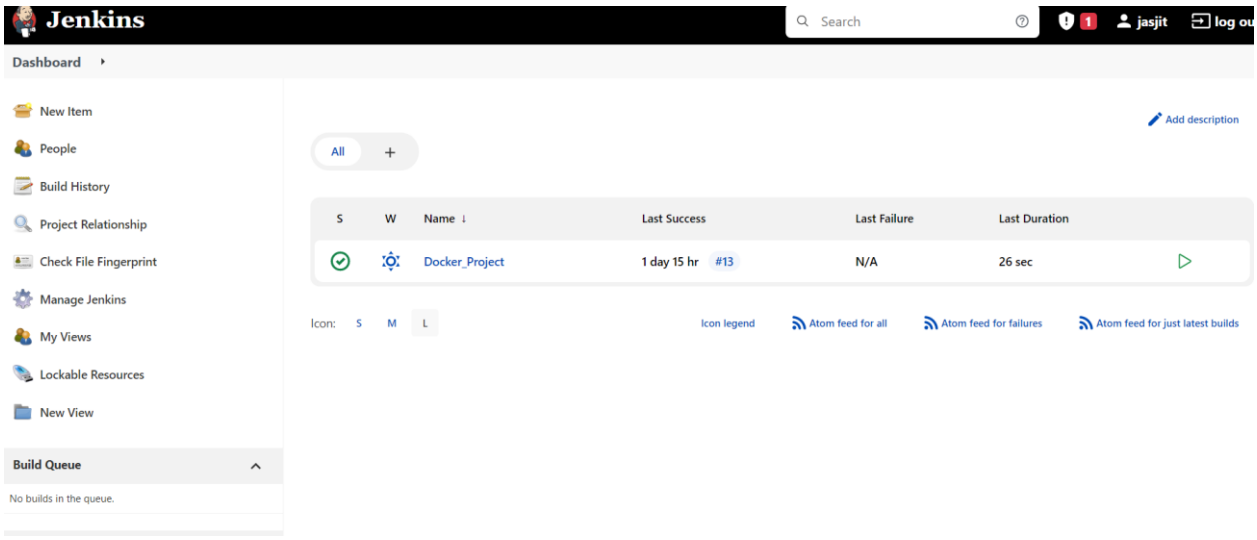
Code being pushed into Github repository from IntelliJ

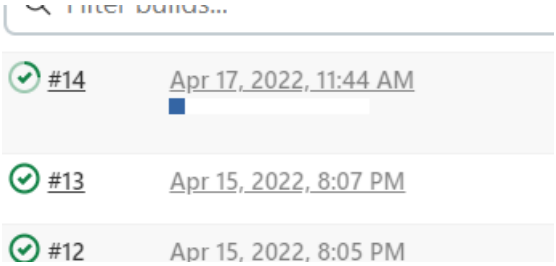


Changes being committed to trigger a webhook

C) Jenkins

Jenkins is an open-source server that is written entirely in Java. It lets you execute a series of actions to achieve the continuous integration process, that too in an automated fashion. We have made a maven project on our Jenkins server, hosted on Amazon EC2 that runs maven commands-clean install package to build our war file from code being pushed onto git repository.





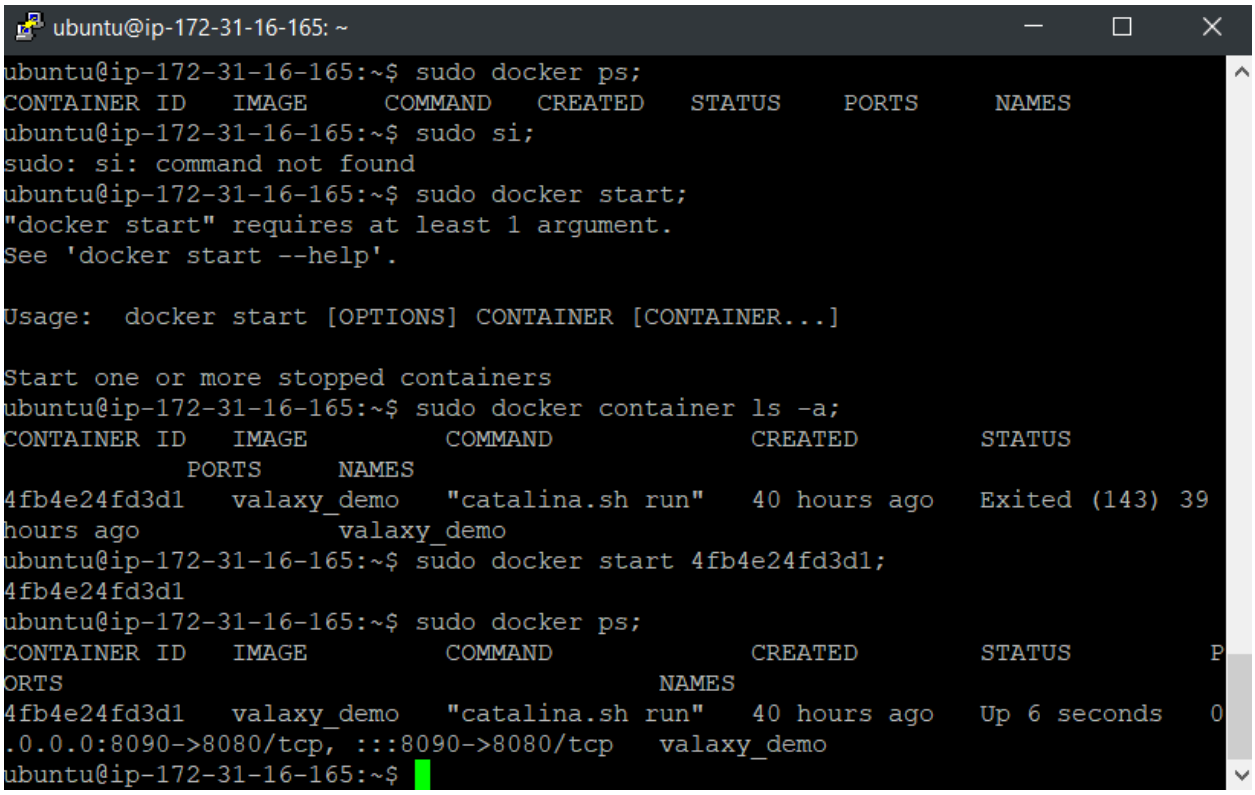
Our Java code being built

Successful build

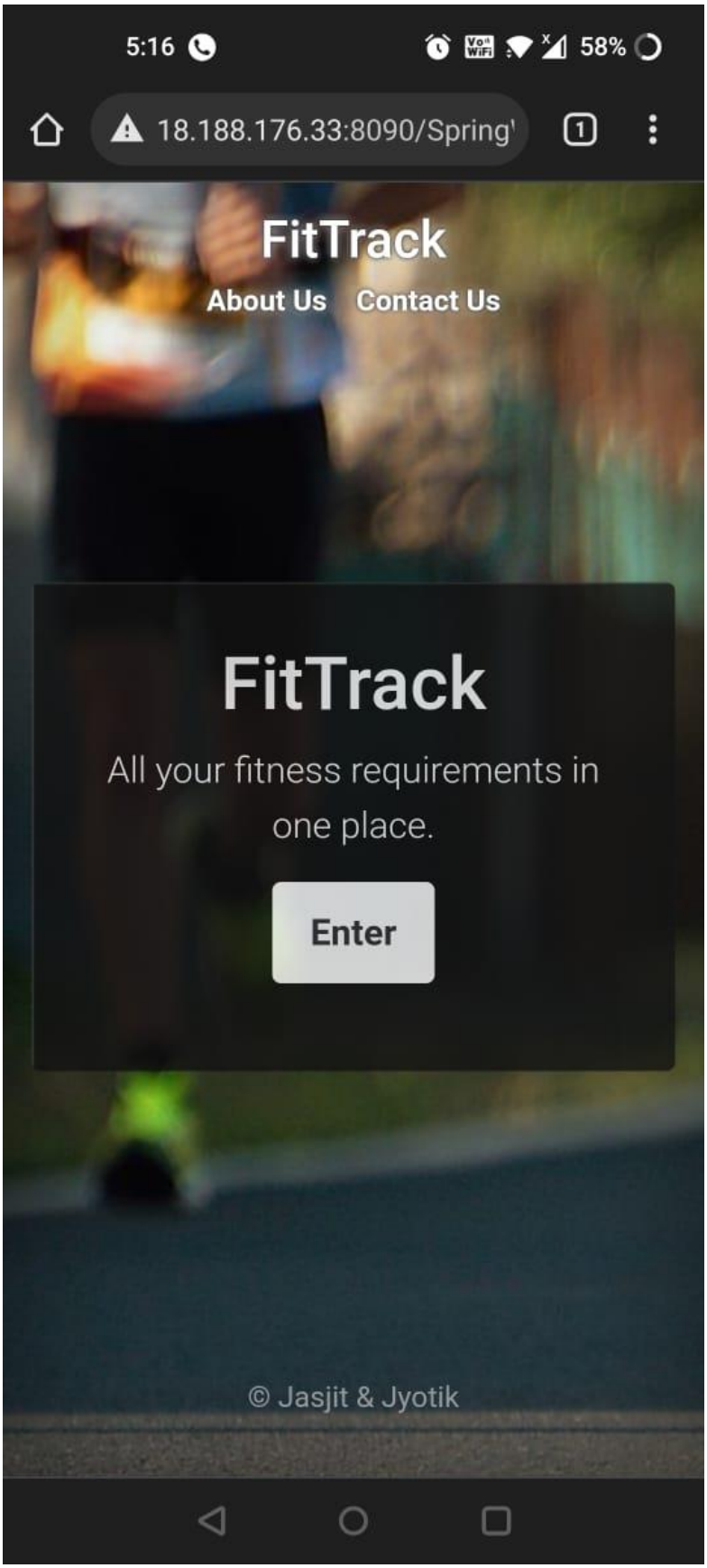
D) Docker Host

Docker has been used to containerize our Java war. Containers can be assigned to another machine that runs Docker and performed there without adaptability issues. Version control and component retain – you can pursue succeeding versions of a container, inspect irregularities, or go back to previous versions. Containers reuse segments from the preceding layers, which makes them remarkably light.

Docker runs on a docker host which is installed on an Amazon EC2 server. This the server which finally hosts our website for everyone on the internet to use.



Managing Docker Host configuration (using Putty)



Our website finally deployed (Notice IP of server)

Technologies Used

- **Java**

  Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA).

- **JSP**

  Jakarta Server Pages (JSP; formerly JavaServer Pages) is a collection of technologies that helps software developers create dynamically generated web pages based on HTML, XML, SOAP, or other document types.

- **Maven**

  Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

- **HTML**

  The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.

- **CSS**

  Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

- **BootStrap**

  Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- **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.

- **Docker**

  Docker is a set of the platform as a service (PaaS) product that use OS-level virtualization to deliver software in packages called containers. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine.

- **Git**

  Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

- **GitHub**

  GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features.

- **AWS EC2**

  Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired.

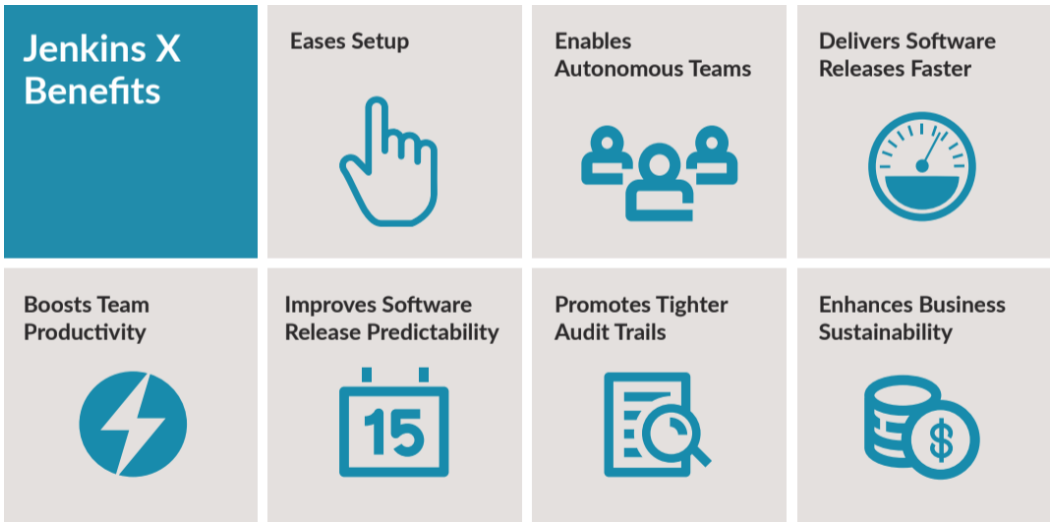
Feasibility Analysis

A. Preliminary Analysis

- Most of the companies build, develop, and test their software using manpower at each step and also hires people to look out for bugs and central maintenance of the code repository this leads to human blunders at each step and increase the time to deploy the product. We can solve this problem using the agile methodology of Jenkins and Docker which removes the need of people by at-least 50% for each process and makes the development to deployment time faster and easier and more centralized controlled.

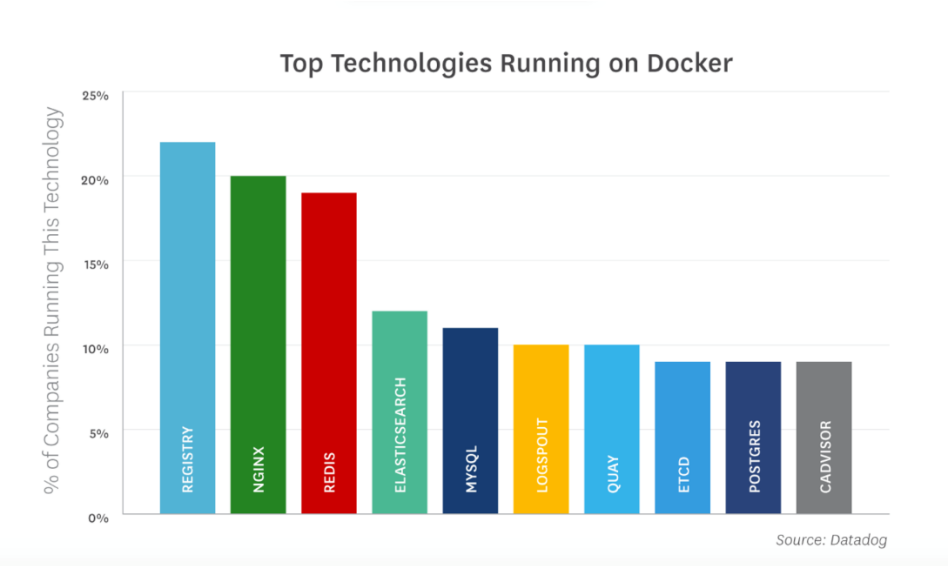
B. Projected costs and implications

- The software required to perform this level automations are on a pay as you go model, and you are not needed to spend more than you need plus the dependency on manual work is reduced and so it helps save precious man hour which can be diverted to other non-mundane tasks in the organization. Overall, the project costs are lowered but how much the actual figure is company and usage dependent. For a small size company, the impact of using it can be seen by the reduction of costs in almost 30-40% as manual testing of software costs labour and precious software cycle time.



C. Market Survey

- As per a survey by Testing Experts companies till rely on Manual testing and hire outsourced companies, this can be saved and company secrets can be protected using Jenkins framework for automated builds and tests, on prem DevOps engineers are required to monitor the pipeline and workflow. Source: (https://www.testingxperts.com/)
- Docker is being used by top companies and webservers all over the world today for a better workflow and error free deployments. for other uses simpler.



Tension free deployment of container images makes replication of images for other uses simpler.

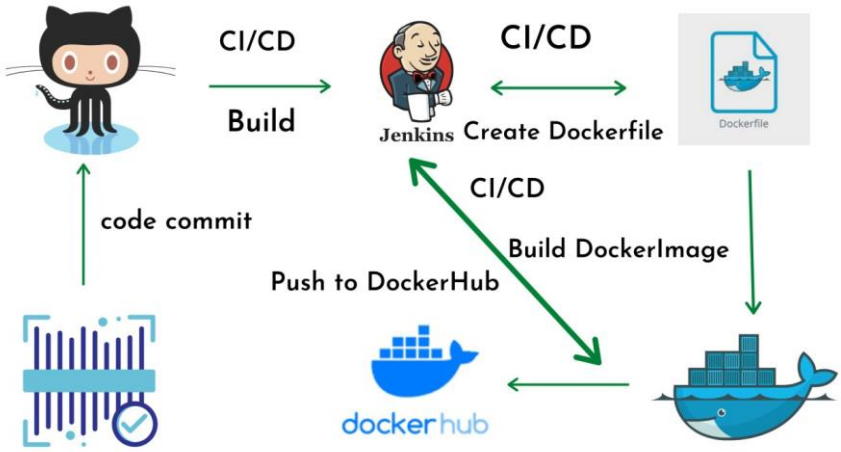
Conclusions

- Using Jenkins and Docker for Automatic builds and testing can help companies to reduce their production time and money and can also help to include customer feedback in products in Realtime through the agile method.

Software Requirement Specification

1. **Product Perspective**
   - This project is to demonstrate the use of Jenkins CLI and Docker, the website is made as test application which is deployed automatically by these tools thus completing the task.

2. **Backend-Process –**
   - Backend of the project is based on the Jenkins Docker Pipeline which automates the build and tests.



   Jenkins is hosted on our machine through PUTTY server and then code is being pushed to an Amazon EC2 instance.

3. **Frontend-Process**
   - Frontend of the Project is a website which takes your inputs and details and recommends a fitness plan to you this whole page works on JSP and automatically updates.

4. **Operation Environment**
   - Amazon EC2 instance is needed to be created
   - Docker Integration
   - Jenkins Server to be running in either on Personal computer through PUTTY server or through online Jenkins server provider.
   - Working Browser EDGE version 11.45.x or higher and Chrome version 44.2.x or higher

5. Performance Requirements

- (Application's performance not only depends on application design also on Customers System's Configuration (both Hardware and Software), Internet Access Speed, networks, and Others)
- Even though the performance is not only depending on application design, our application design and implementation also responsible for the Performance.

6. Compatibility Requirements

- (As it is a Internet Application, it has to support various Hardware configurations, Software and Network Communications)
- It should support all types of Hardware versions, Operating Systems and Browsers
- Operating systems - Windows 8.1.x and upper Versions (EX: Win10, windows 11, windows server 2012 and higher server OS) <Recommended, but optional>
- Ubuntu and other Browser supporting GUI based operating systems with Linux kernel 3.x and higher (Arch Linux, OPENSUS 14.x etc) <Recommended, but optional>

Introduction and detail of software used

a) JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets. It physically exists. It contains JRE + development tools.

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) etc. to complete the development of a Java Application.

Since the code for our project is written in java, JDK is a required software.

b) Apache Tomcat

Apache Tomcat is a free and open-source implementation of the Jakarta Servlet, Jakarta Expression Language, and WebSocket technologies. Tomcat provides a "pure Java" HTTP web server environment in which Java code can run.

Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license.

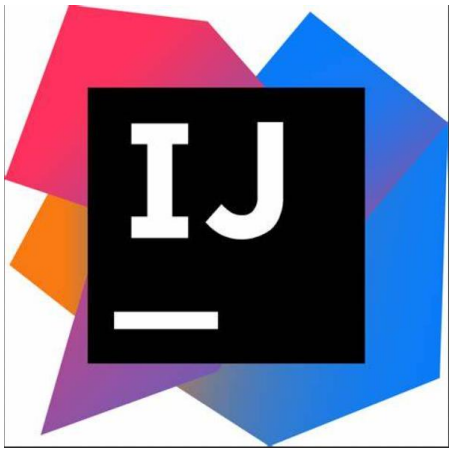
The initial idea of Apache tomcat software was to host and deploy the Java servlet that is the server-side Java code that manages HTTP results from client applications build using Java. It acts as a web server rather than a full-fledged application server that includes data persistence and load balancing capabilities. Apache Tomcat provides the basic feature of web server processing for the relevant servlets. It supports the java servlet lifecycle that are init(), service() and destroy() phases. It is the preferred web server software for Java implementations.

c) IntelliJ Idea

IntelliJ is one of the most powerful and popular Integrated Development Environments (IDE) for Java. It is developed and maintained by **JetBrains** and available as community and ultimate edition. This feature rich IDE enables rapid development and helps in improving code quality.

IntelliJ IDEA has some top productive Java code completion features. Its predictive algorithm can accurately assume what a coder is attempting to type, and completes it for him, even if he doesn't know the exact name of a particular class, member or any other resource.

IntelliJ IDEA is our primary IDE used for this project.



d) Git

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

Git is foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.

e) Microsoft Edge

Microsoft Edge is a cross-platform web browser created and developed by Microsoft. It was first bundled with Windows 10 and Xbox One in 2015, and later released for other platforms: Android and iOS in 2017, macOS and older Windows versions (Windows 7 and later) in 2019, and Linux in 2020.

The Chromium-based Edge replaced Internet Explorer (IE) in Windows 11, as the default web browser (for compatibility with Google Chrome).

Edge is the primary browser for testing our web app.

f) Jenkins

Jenkins is an open-source automation tool written in Java programming language that allows continuous integration.

Jenkins **builds** and **tests** our software projects which continuously making it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build.

It also allows us to continuously **deliver** our software by integrating with many testing and deployment technologies.

Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

With the help of Jenkins, organizations can speed up the software development process through automation. Jenkins adds development life-cycle processes of all kinds, including build, document, test, package, stage, deploy static analysis and much more.

Jenkins achieves CI (Continuous Integration) with the help of plugins. Plugins is used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you must install the plugins for that tool. For example: Maven 2 Project, Git, HTML Publisher, Amazon EC2, etc.

For example: If any organization is developing a project, then **Jenkins** will continuously test your project builds and show you the errors in early stages of your development.

Possible steps executed by Jenkins are for example:

- Perform a software build using a build system like Gradle or Maven Apache
- Execute a shell script
- Archive a build result
- Running software tests

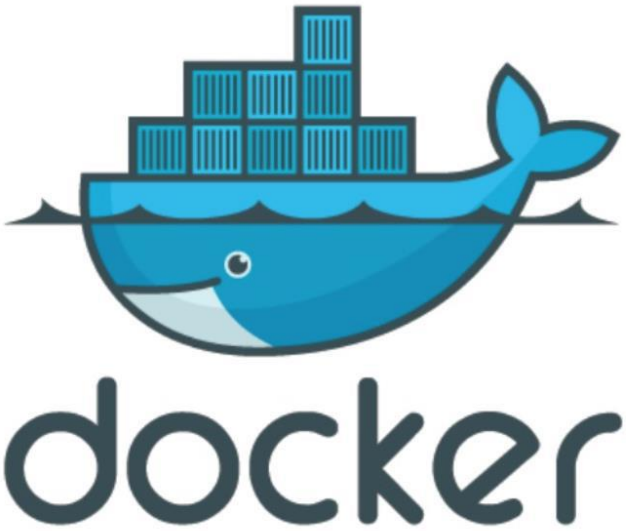
g) Docker

Docker is a container management service. The keywords of Docker are **developed, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

Docker can reduce the size of development by providing a smaller footprint of the operating system via containers.

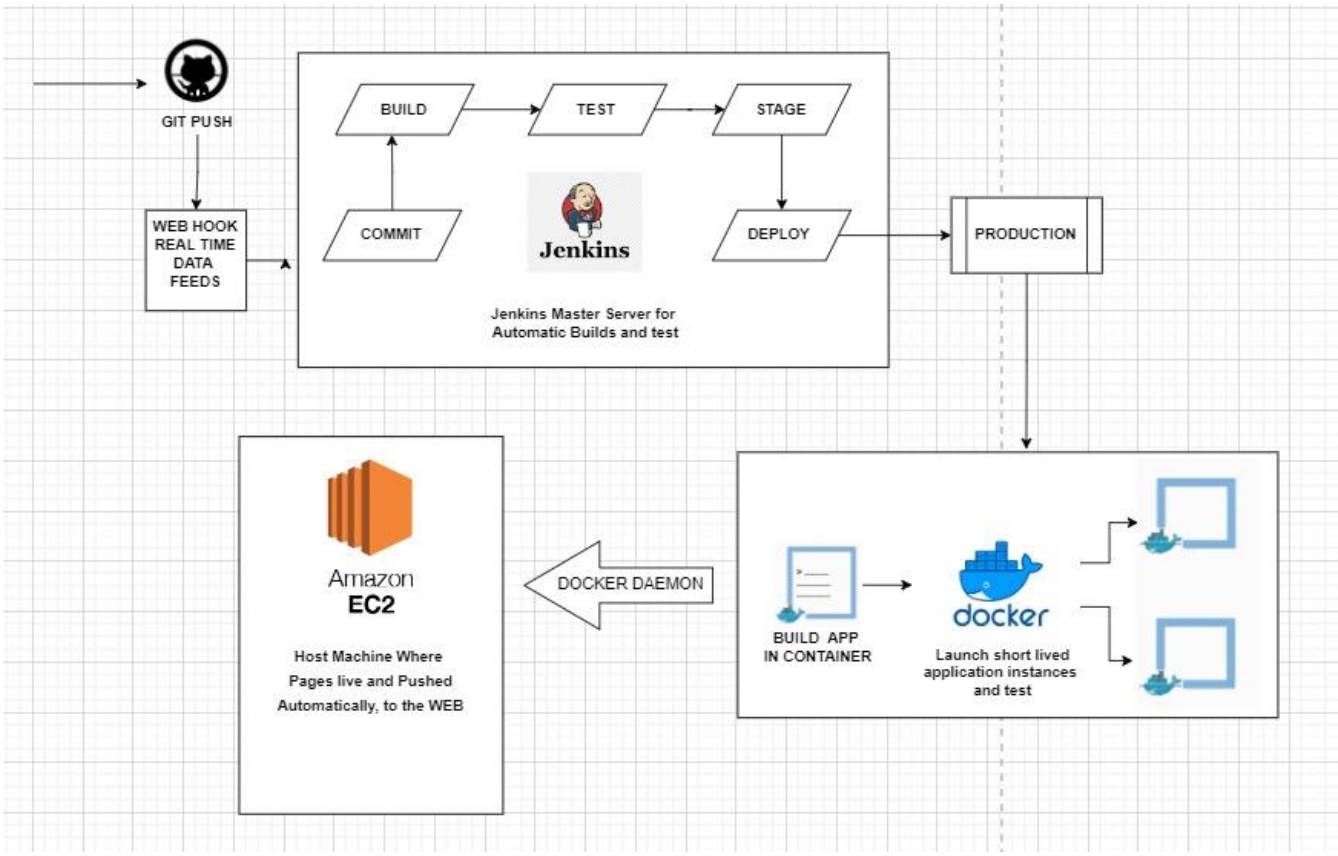
With containers, it becomes easier for teams across different units, such as development, QA, and Operations to work seamlessly across applications.
You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.

Since Docker containers are lightweight, they are very easily scalable.

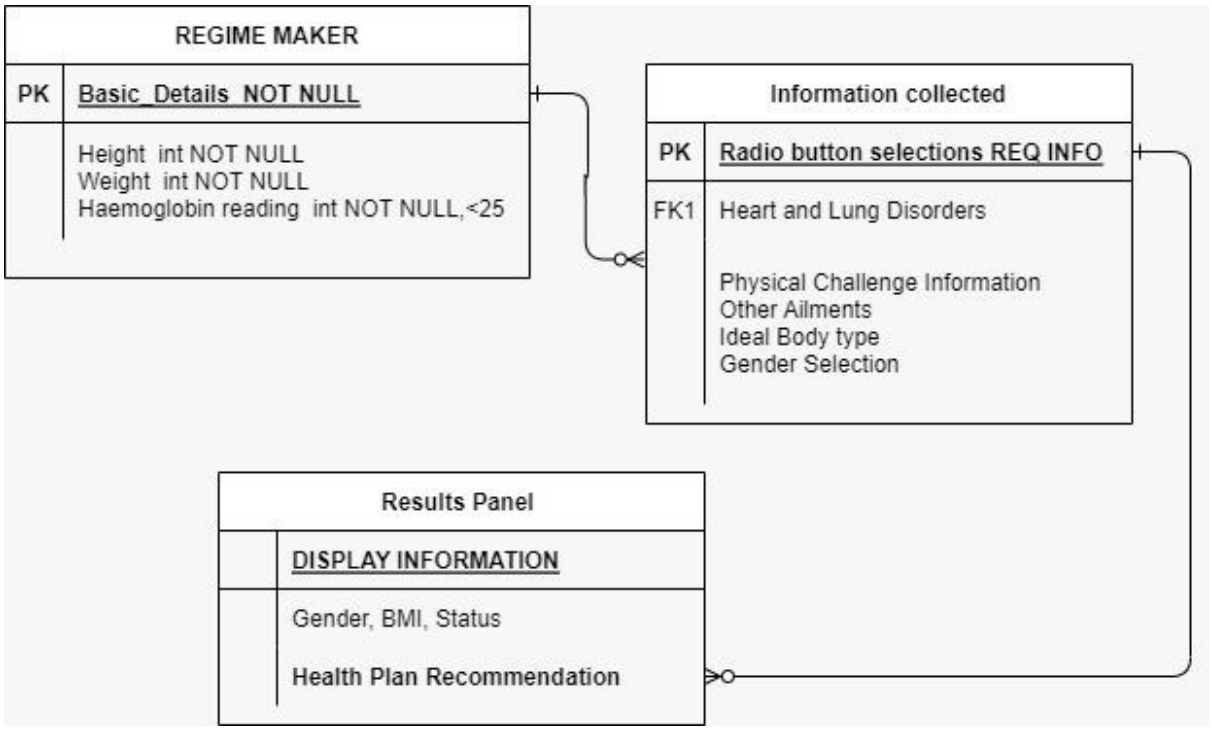

docker

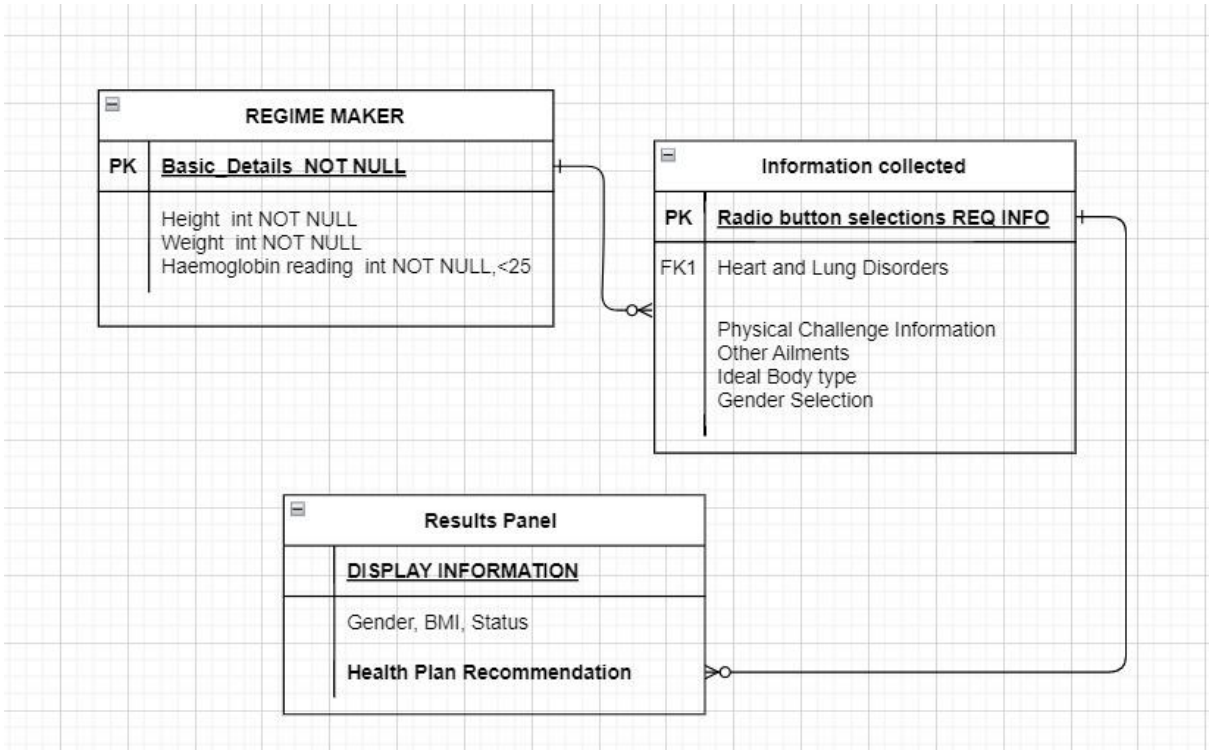
System Design

Data Flow Diagram (DFD)



Entity Relation Diagrams


REGIME MAKER
PK
Basic_Details NOT NULL
Height int NOT NULL
Weight int NOT NULL
Haemoglobin reading int NOT NULL,<25
Information collected
PK
Radio button selections REQ INFO
FK1
Heart and Lung Disorders
Physical Challenge Information
Other Ailments
Ideal Body type
Gender Selection
Results Panel
DISPLAY INFORMATION
Gender, BMI, Status
Health Plan Recommendation


REGIME MAKER
PK
Basic_Details NOT NULL
Height int NOT NULL
Weight int NOT NULL
Haemoglobin reading int NOT NULL,<25
Information collected
PK
Radio button selections REQ INFO
FK1
Heart and Lung Disorders
Physical Challenge Information
Other Ailments
Ideal Body type
Gender Selection
Results Panel
DISPLAY INFORMATION
Gender, BMI, Status
Health Plan Recommendation

Implementation

>Webapp

Our webapp's implementation is divided into two main parts:

a) Front-end

Front end has been implemented using HTML, CSS, and Bootstrap modules. Using tags, the integrity of page is maintained.

```
          None
        </label>
      </div>
    </div>
  </div>
</fieldset>

<fieldset class="form-group">
  <div class="row">
    <legend class="col-form-label col-sm-2 pt-0">Any physical challenges?</legend>
    <div class="col-sm-10">
      <div class="form-check">
        <input class="form-check-input" type="radio"  id="yes" name="physicalChallenges" value="yes" required>
        <label class="form-check-label" for="gridRadios1">
          Yes
        </label>
      </div>
      <div class="form-check">
        <input class="form-check-input" type="radio" id="no" name="physicalChallenges" value="no" checked>
        <label class="form-check-label" for="gridRadios2">
          No
        </label>
      </div>
```

Front End Code Snippet

b) Back-end

Back-end functionality has been implemented using jsp technology. Conditional statements have been put to deal with various scenarios to provide best recommendation.

```
<%
            String c = request.getParameter("weight");
            String j = request.getParameter("height");
            String disease = request.getParameter("disease");
            String ailments = request.getParameter("ailments");
            String physicalChallenges = request.getParameter("physicalChallenges");
            String bodyType = request.getParameter("bodyType");

            if(!(c == null || c.isEmpty()))
            {
                float weight = Float.parseFloat(c);
                float height = Float.parseFloat(j);

                float bmi = weight/(height*height);
                String grade ="";
                String recommendation = "";

                if(bmi < 19){
                    grade = "Underweight";
                    //request.setAttribute("BMI", grade);
                }
                else if(bmi > 19 && bmi < 25){
                     grade = "Healthy";
                     //request.setAttribute("BMI", grade);
                }
                else if(bmi > 19 && bmi < 25){
                     grade = "Overweight";
                     //request.setAttribute("BMI", grade);
                }
```

Back-end code snippet

> EC2 (global)

Inbound and Outbound rules for EC2 are as follows:

Inbound rules (8)

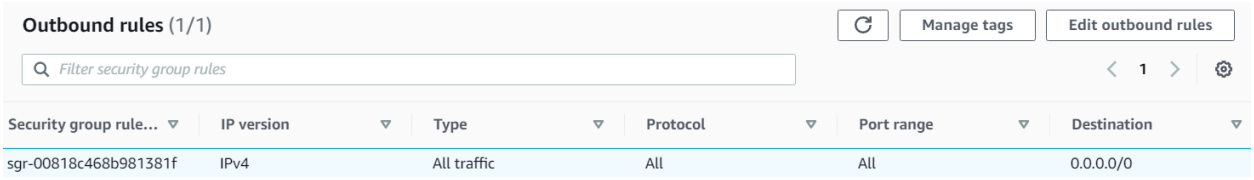
Manage tags | Edit inbound rules

Filter security group rules

1

Security group rule...	IP version	Type	Protocol	Port range	Source
sgr-00966033cabee3c41	IPv4	SSH	TCP	22	0.0.0.0/0
sgr-0564b6de08eb8f703	IPv4	HTTP	TCP	80	0.0.0.0/0
sgr-0c370ff47eea7bacc	IPv4	HTTPS	TCP	443	0.0.0.0/0
sgr-04923d413ebd5e...	IPv6	Custom TCP	TCP	8080	::/0
sgr-0a084eef181cec00c	IPv4	Custom TCP	TCP	8090	0.0.0.0/0
sgr-0a4c95501330046f0	IPv6	HTTP	TCP	80	::/0
sgr-0c7fdfcb9b3abf62f	IPv6	HTTPS	TCP	443	::/0
sgr-00e68b321f4fd89cc	IPv4	Custom TCP	TCP	8080	0.0.0.0/0

Inbound Rules



Outbound Rules

AMI: Ubuntu Inferred

EC2 Type: T2Micro

> Jenkins

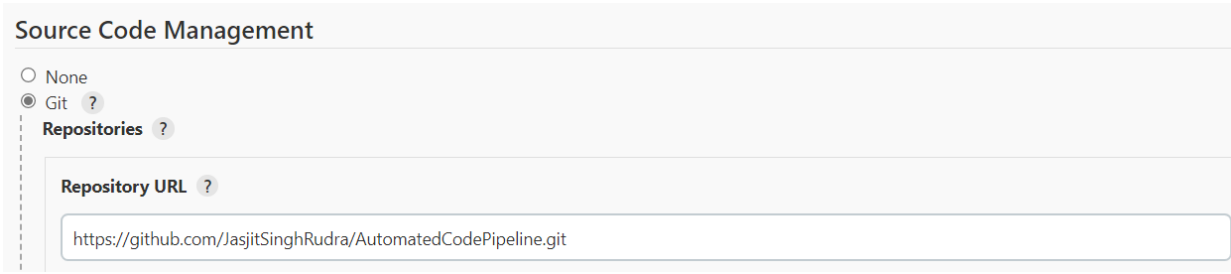
Jenkins EC2 Configuration

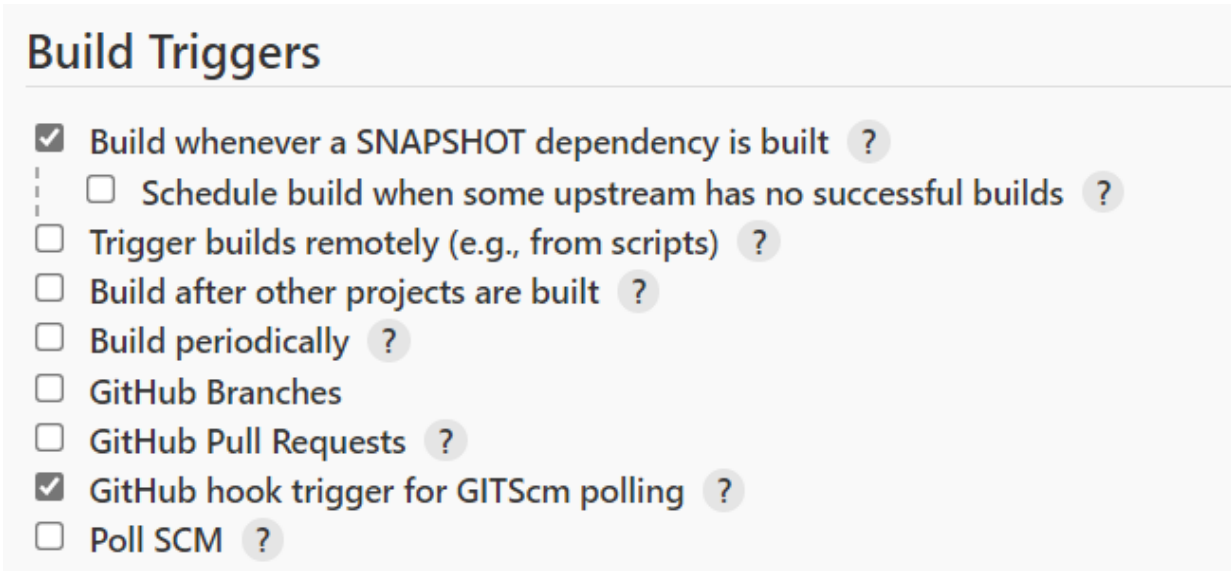
```
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-
stable/jenkins.io.key

amazon-linux-extras install epel
amazon-linux-extras install java-openjdk11
yum install Jenkins

# Start jenkins service
service jenkins start
```

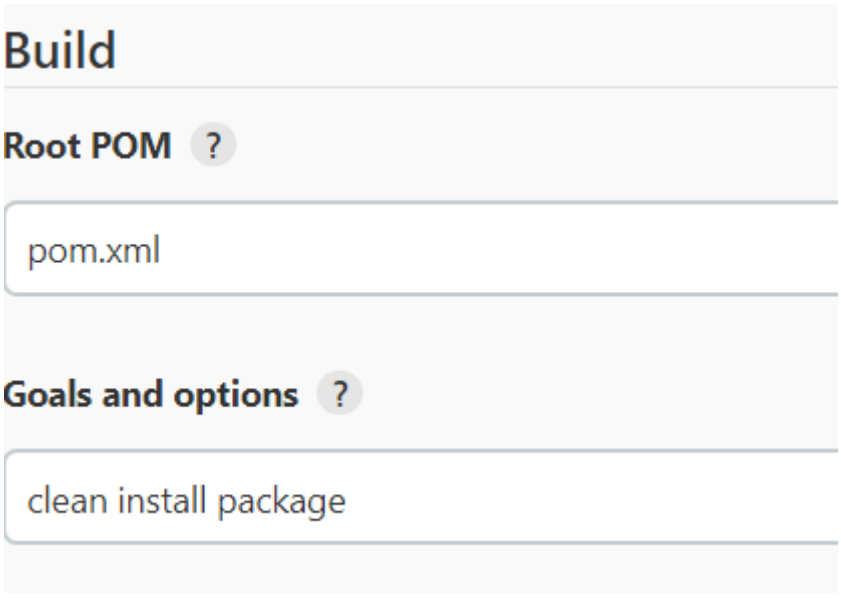
Accessing Jenkins

```
http://YOUR-SERVER-PUBLIC-IP:8080
```

Jenkins Configuration


Source Code Management
None
Git
Repositories
Repository URL
https://github.com/JasjitSinghRudra/AutomatedCodePipeline.git


Build Triggers
Build whenever a SNAPSHOT dependency is built
Schedule build when some upstream has no successful builds
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub Branches
GitHub Pull Requests
GitHub hook trigger for GITScm polling
Poll SCM


Build
Root POM
pom.xml
Goals and options
clean install package

Send files or execute commands over SSH

?

SSH Publishers

SSH Server

Name ?

docker host

Transfers

Transfer Set

Source files ?

target/*.war

Remove prefix ?

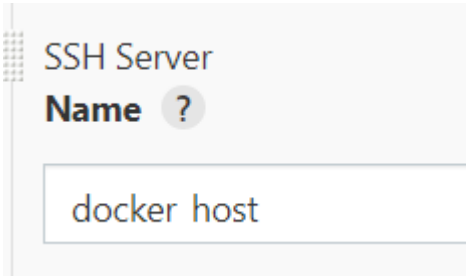
target

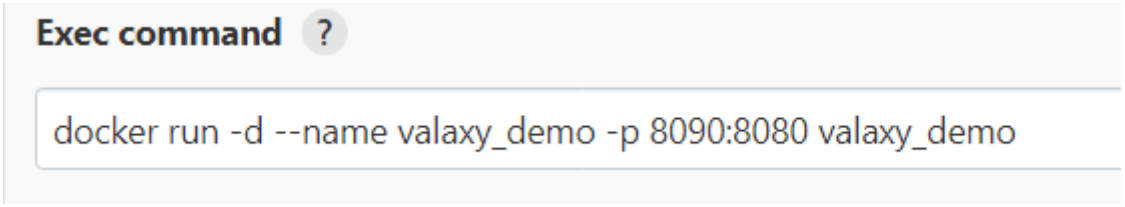
Remote directory ?

//opt//docker

Exec command ?

```
docker stop valaxy_demo;
docker rm -f valaxy_demo;
docker image rm -f valaxy_demo;
cd /opt/docker;
docker build -t valaxy_demo .
```





> **Docker**

Docker EC2 Configuration

Install Docker

```
sudo yum update -y
sudo yum -y install docker
```

Start Docker

```
sudo service docker start
```

Create a new user for Docker management and add him to Docker (default) group

```
useradd dockeradmin
passwd dockeradmin
usermod -aG docker dockeradmin
```

Write a Docker file under /opt/docker

```
mkdir /opt/docker

### vi Dockerfile
# Pull base image
From tomcat:8-jre8

# Maintainer
MAINTAINER "valaxytech"

# copy war file on to container
COPY ./webapp.war /usr/local/tomcat/webapps
```

Exec command[s] :

```
docker stop valaxy_demo;
docker rm -f valaxy_demo;
docker image rm -f valaxy_demo;
cd /opt/docker;
docker build -t valaxy_demo .
```

Login to Docker host and check images and containers. (no images and containers).

Execute Jenkins job.

Check images and containers again on Docker host. This time an image and container get creates through Jenkins job.

Access web application from browser which is running on container.

```
<docker_host_Public_IP>:8090
```

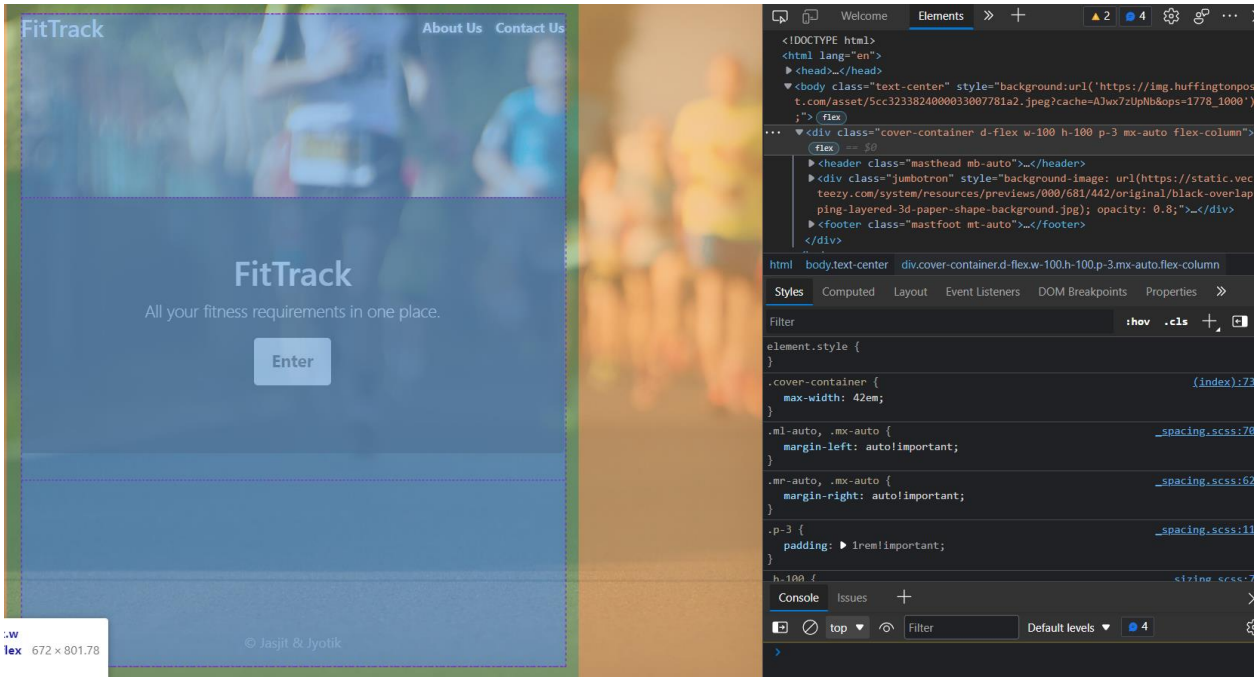
Testing

A) Testing Backend

Test Case for BMI Calculation

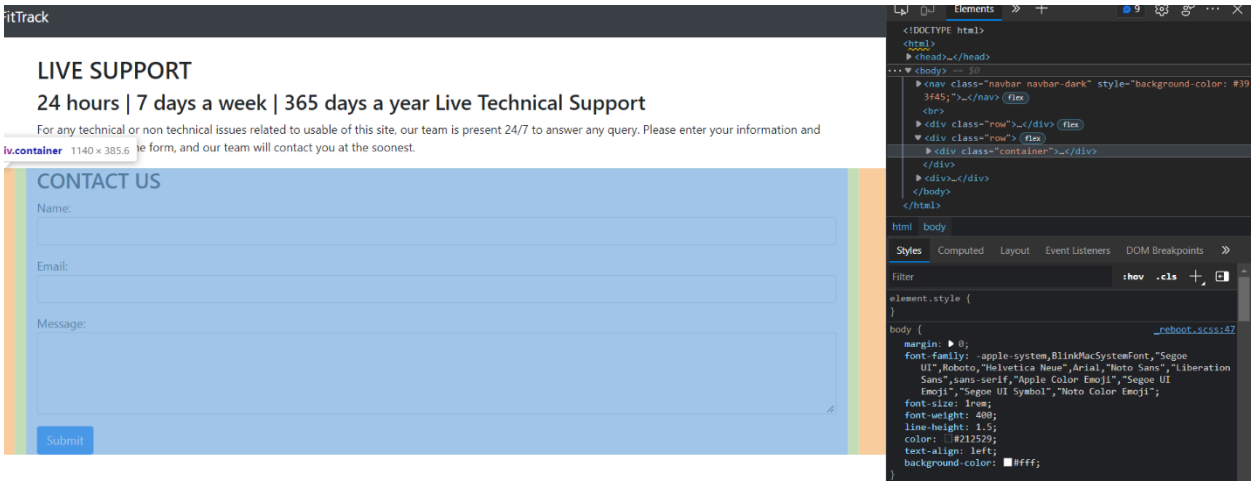
```java
package com.demo;

import org.junit.Assert;
import org.junit.Test;
import org.mockito.*;

public class BmiTest {

    @Mock
    Bmi bmi;

    @Test
    public void testBMI(){
        Mockito.when(bmi.setGender()).thenReturn( value: "Male");
        Mockito.when(bmi.setHeight()).thenReturn( value: 180);
        Mockito.when(bmi.setWeight()).thenReturn( value: 70);
        Assert.assertEquals( expected: "Healthy", bmi.getResult());
    }
}
```

B) Testing Frontend

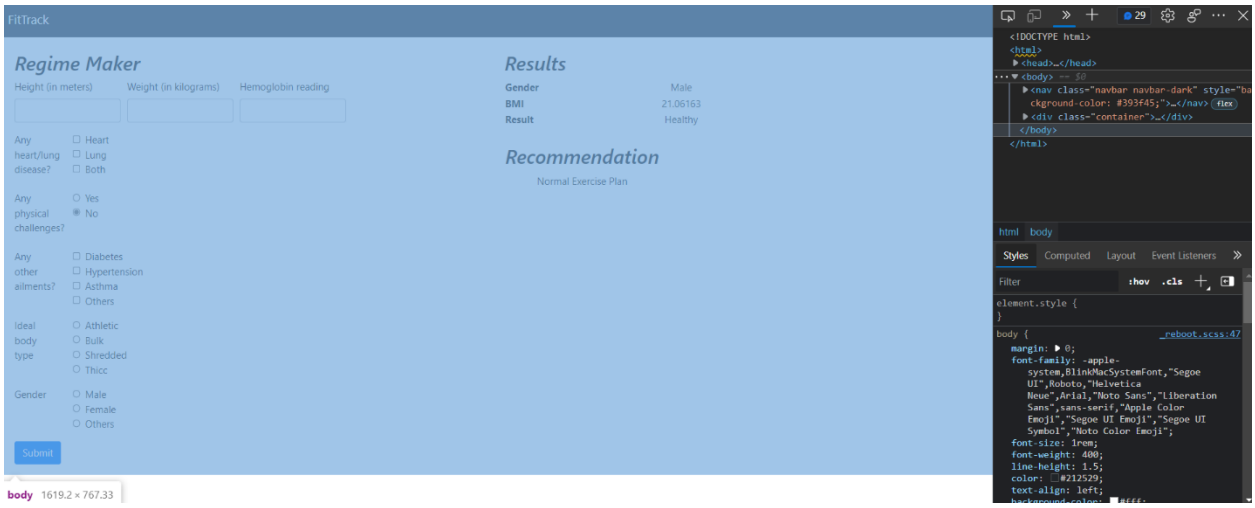


Home page was inspected and tested by examining it using inspect element to check every element is behaving as expected.

About us page was inspected and tested by examining it using inspect element to check every element is behaving as expected.



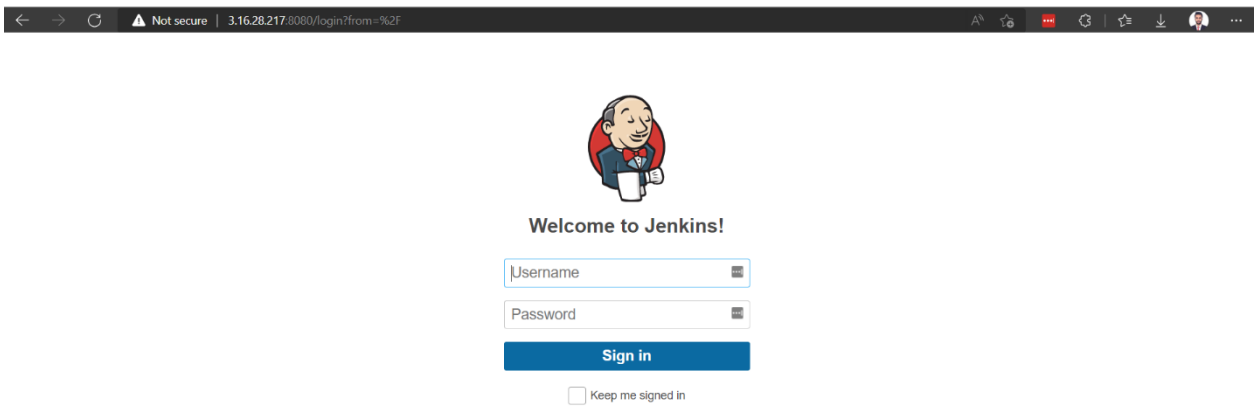
Contact us was inspected and tested by examining it using inspect element to check every element is behaving as expected.

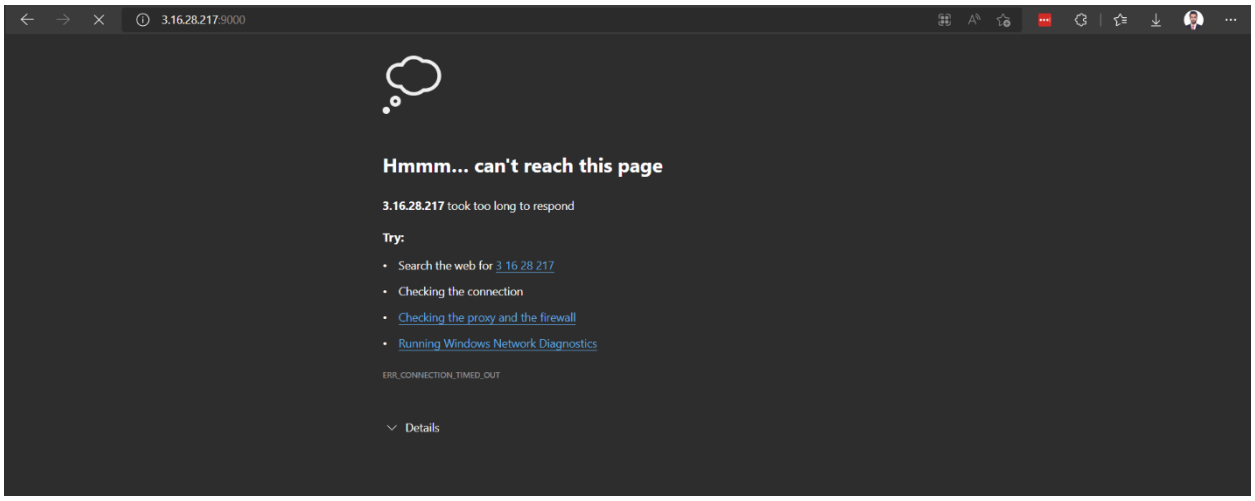


BMI calculation page was inspected and tested by examining it using inspect element to check every element is behaving as expected.

C) Testing EC2 Security

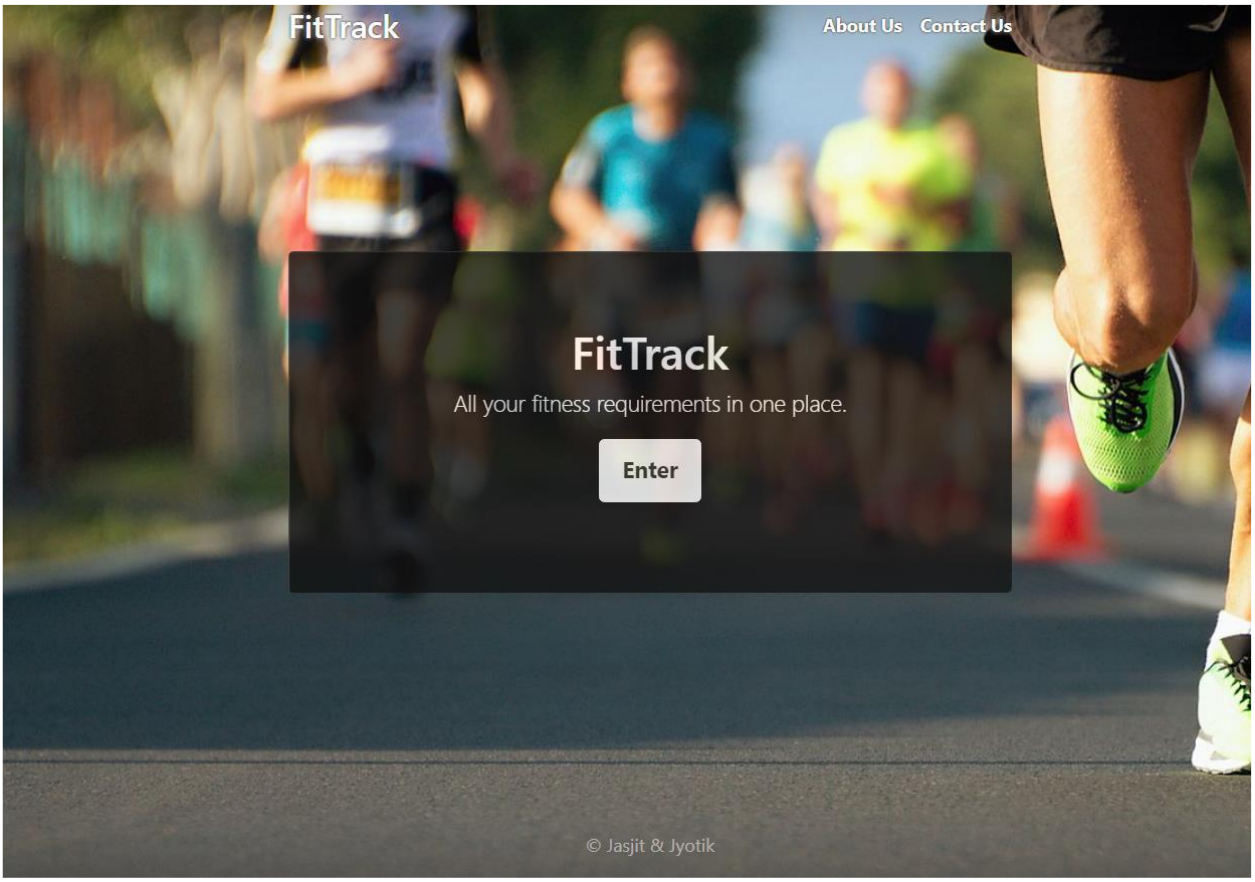
To make sure we are only able to access EC2 through allowed ports only, both wrong and right port addresses were provided



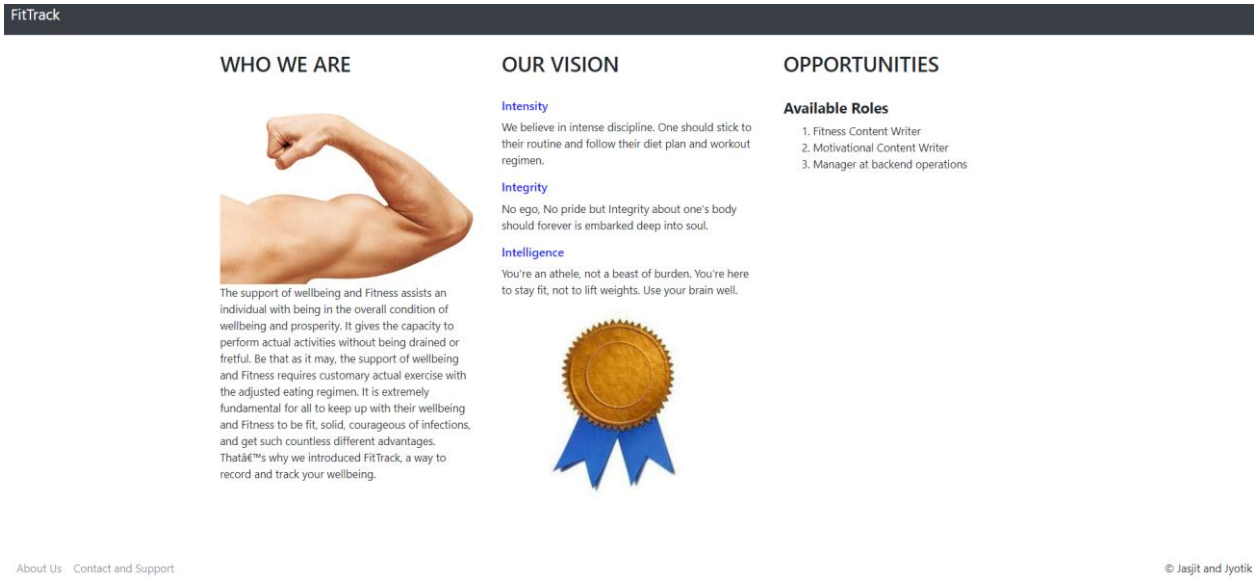

3.16.28.217:9000
Hmmm... can't reach this page
3.16.28.217 took too long to respond
Try:
Search the web for 3 16 28 217
Checking the connection
Checking the proxy and the firewall
Running Windows Network Diagnostics
ERR_CONNECTION_TIMED_OUT
Details

Screenshots

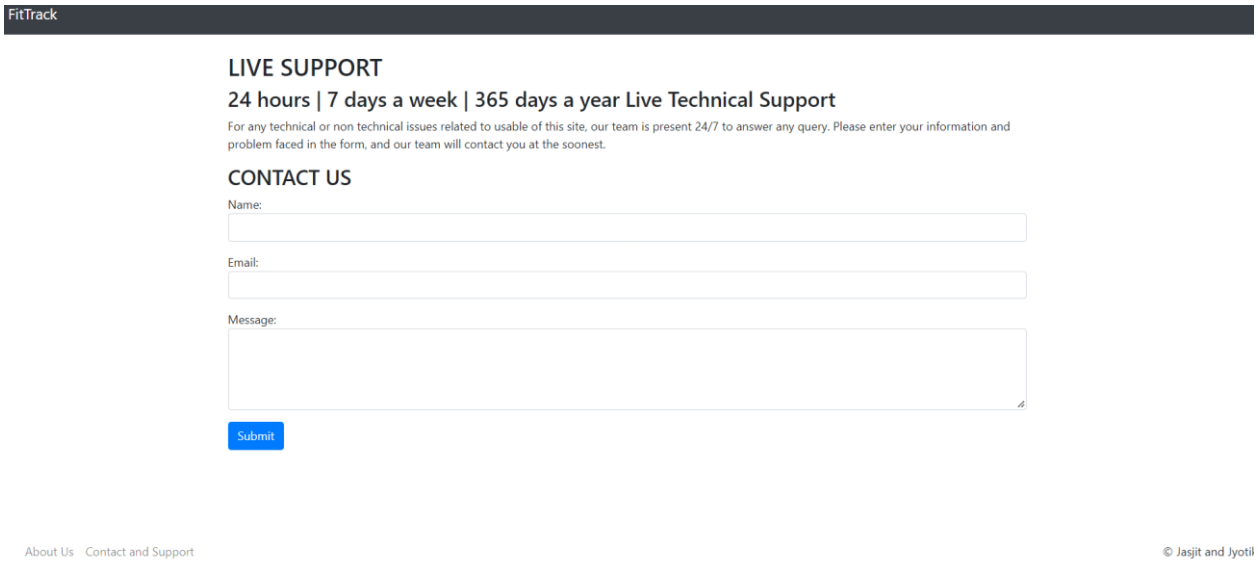
- **Webapp**



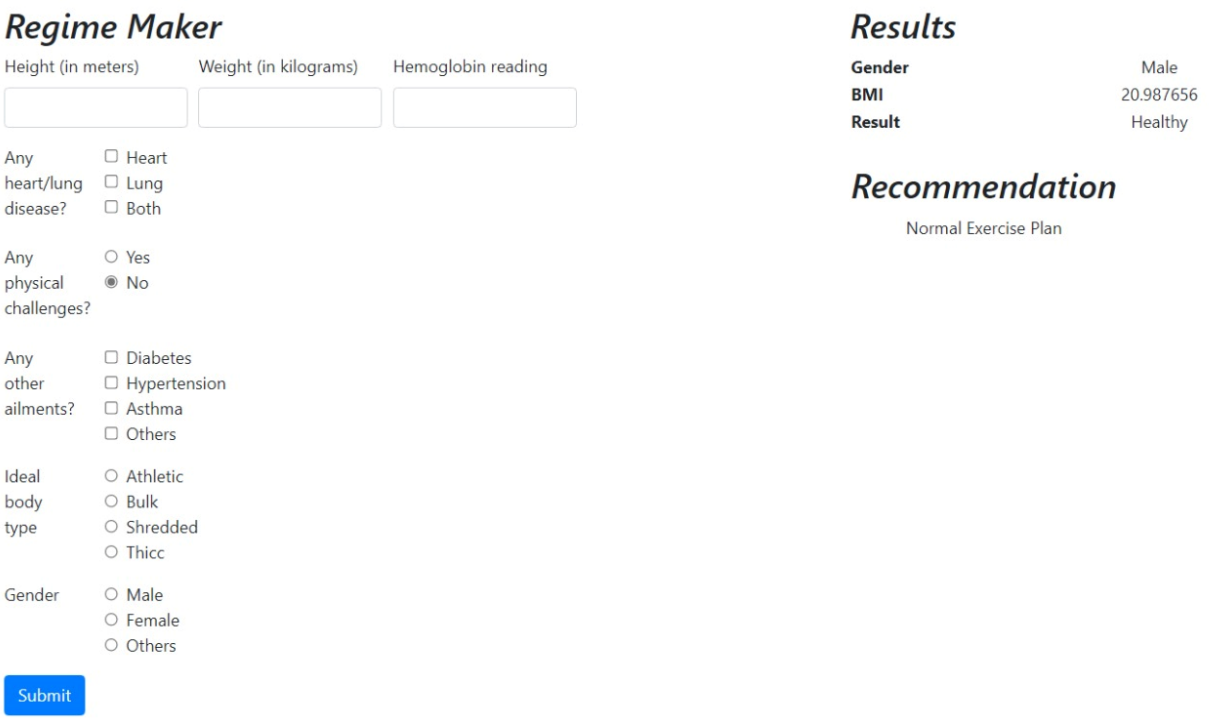
Homepage



About Us Page

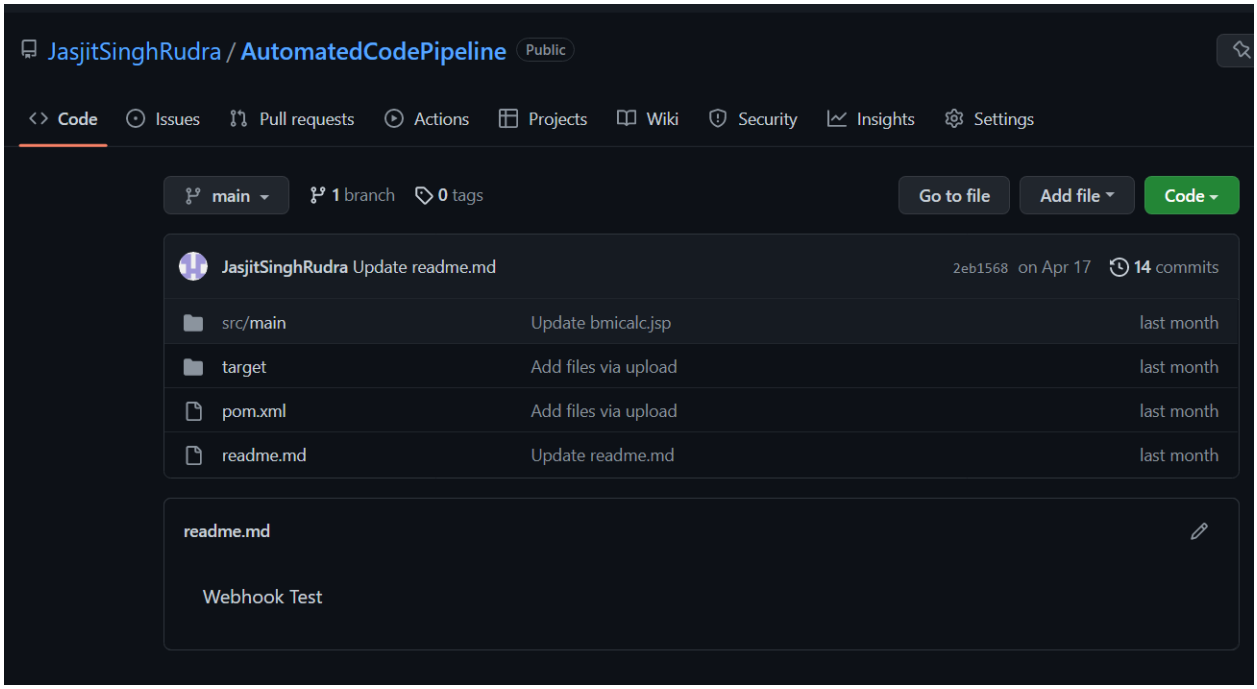


Contact Us Page

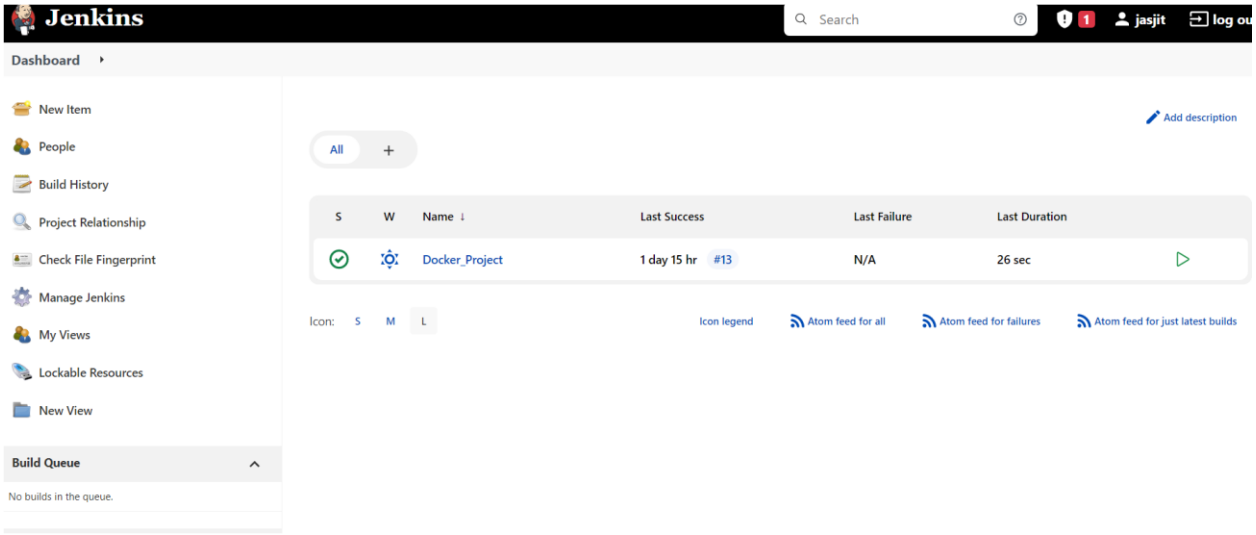


BMI Calculation Page

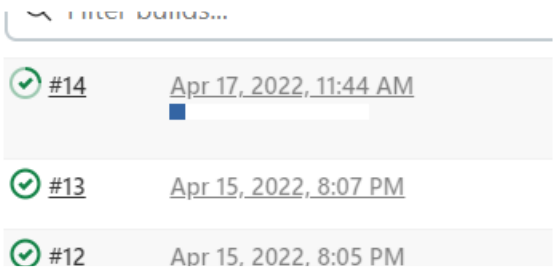
- **Version Control System (Github.com)**



- **Jenkins Server**



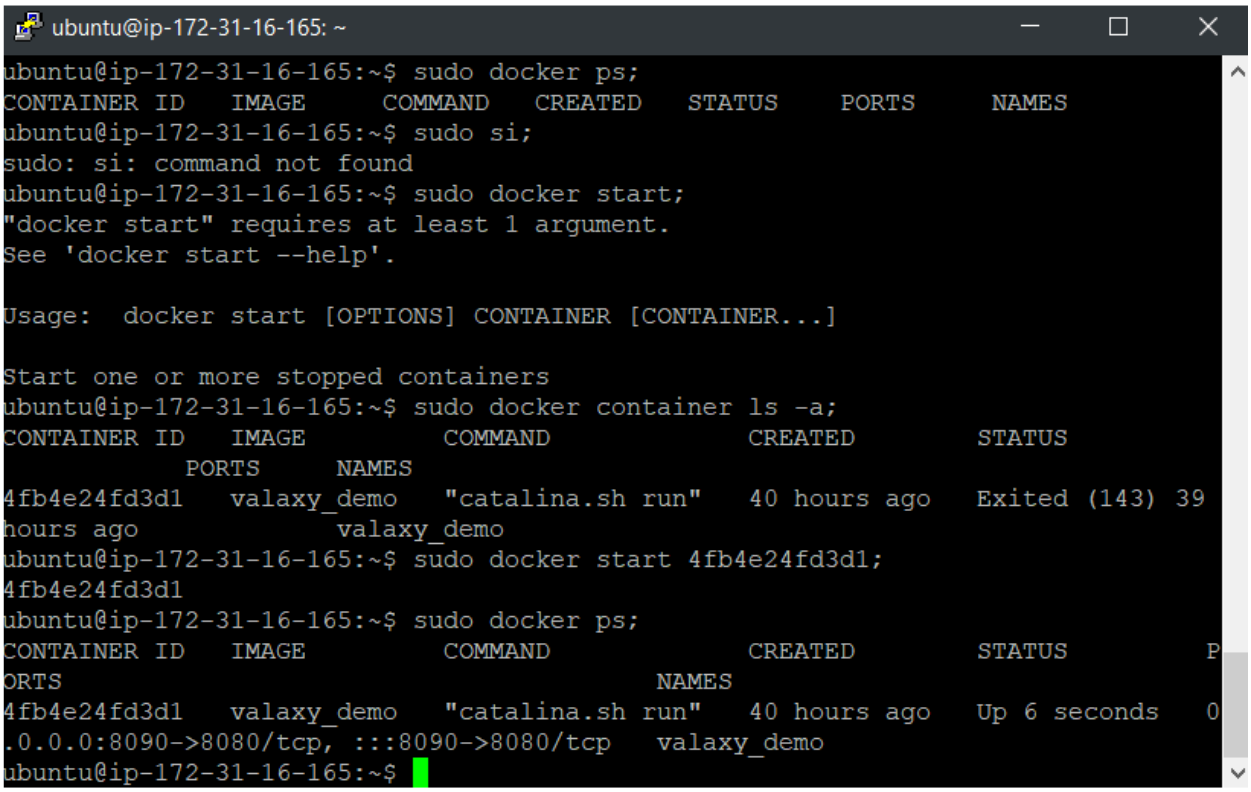
Jenkins Home



Our Java code being built

Successful build

- **Docker Server**



Conclusion and Future Scopes

In the ever evolving and competitive software industry, change is inevitable. Using older mindsets and methodologies for software development only produces more stress on developers, managers and reduces customer satisfaction.

Using Agile,

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers, and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between businesspeople and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.

DevOps extends the focus to delivery and maintenance. Release faster and work smarter. Teams that practice DevOps release deliverables more frequently, with higher quality and stability. The team with the fastest feedback loop is the team that thrives. Full transparency and seamless communication enable DevOps teams to minimize downtime and resolve issues faster.

Large projects in earlier times were almost impossible to change and costed over millions to change something, but now with smaller iterations, change is always welcome.

Demonstrating the use of modern DevOps CI/CD tools and technologies, we have successfully demonstrated the speed which can now be achieved to deploy our project from pushing to VCS to deployment.

DevOps, especially DevSecOps is a very wide scope. In the future of our pipeline, we can integrate security tools and code coverage tools such as Sonarqube, Checkmarx, Blackduck which can help us determine code smells, vulnerabilities in code, code coverage percentage and dependencies having security concerns.

Moreover, the whole process of cloud deployment could further be automated using Terraform or AWS CodeDeploy which offers Infrastructure as a Code. We can have similar/exact configurations for different availability zones or regions as demand for our application grows worldwide.

References

JDBC Servlets and JSP - Java Web Development Fundamentals
https://udemy.com/course/jdbcservletsandjsp/learn/lecture/5801198?start=0#overview

Ultimate AWS Certified Cloud Practitioner - 2022
https://udemy.com/course/aws-certified-cloud-practitioner-new/learn/lecture/20053288?start=15#overview

DevOps: CICD with Git GitLab Jenkins, Docker and Django
https://udemy.com/course/devops-project-cicd-with-git-gitlab-jenkins-and-django/learn/lecture/22779645?start=0#overview

Introduction to Continuous Integration & Continuous Delivery
https://udemy.com/course/introduction-to-continuous-integration-and-continuous-delivery/learn/lecture/22937046?start=0#overview

Jenkins
https://www.jenkins.io/doc/book/

Docker
https://docs.docker.com

AWS
https://aws.amazon.com/whitepapers