CI/CD PIPELINE USING JENKINS

SCOPE:

To set up a Jenkins CI/CD pipeline using GitHub, SonarQube, and Docker on an AWS EC2 instance.

PURPOSE:

The main goal of a CI/CD pipeline using Jenkins is to streamline the software development and delivery process, making it more efficient, reliable, and responsive to changes in the codebase.

TOOLS AND TECHNOLOGIES USED:

1. DOCKER:

Docker is an open-source platform for building, shipping, and running containerized applications.

Docker has become a popular tool for application development and deployment, as it enables developers to package their applications into containers and deploy them to any platform that supports Docker, from local development environments to public clouds. It also provides a wide range of integrations and extensions, allowing users to customize and extend the platform as needed.

2. GITHUB:

GitHub is a code repository that provides version control and collaboration tools, and SonarQube is a code quality management tool that helps analyse and manage code quality in software projects.

GitHub is a web-based platform used for version control and collaboration on software development projects. It provides a central repository for code and allows developers to work together on projects, track changes to code over time, and collaborate on new features or bug fixes.

3. JENKINS:

Jenkins is a popular open-source automation server that helps automate various tasks in the software development process, including building, testing, and deploying software.

Jenkins is an open-source automation tool written in Java with plugins built for continuous integration. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project and making it easier for users to obtain a fresh build.

4.SONARQUBE:

SonarQube is a Code Quality Assurance tool that collects and analyses source code and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time.

IMPLEMENTATION:

we will create an AWS EC2 instance and install Jenkins on it. We will then set up a GitHub repository and integrate it with Jenkins. Next, we will configure SonarQube to analyse code quality in our GitHub repository. Finally, we will create a Docker image of our application and deploy it using Jenkins.

Implementing a CI/CD pipeline involves setting up a series of steps to automate the process of building, testing, and deploying software applications.
Setting up a CI/CD pipeline using Jenkins requires careful consideration of system requirements to ensure smooth operation and efficient performance. Below are the typical system requirements for a Jenkins-based CI/CD pipeline:

Hardware Requirements:

CPU: Multi-core processor (e.g., 4 cores or more)

RAM: At least 8 GB of RAM (More RAM may be required depending on the complexity of your build and test processes)
Disk Space: Sufficient disk space to store Jenkins itself, plugins, build artifacts, and logs. A minimum of 50 GB is recommended, but this may vary depending on the scale of your projects.
Operating System:

Jenkins can run on various operating systems, including Windows, macOS, and Linux. Linux is often preferred for its stability and performance.
Java Runtime Environment (JRE):

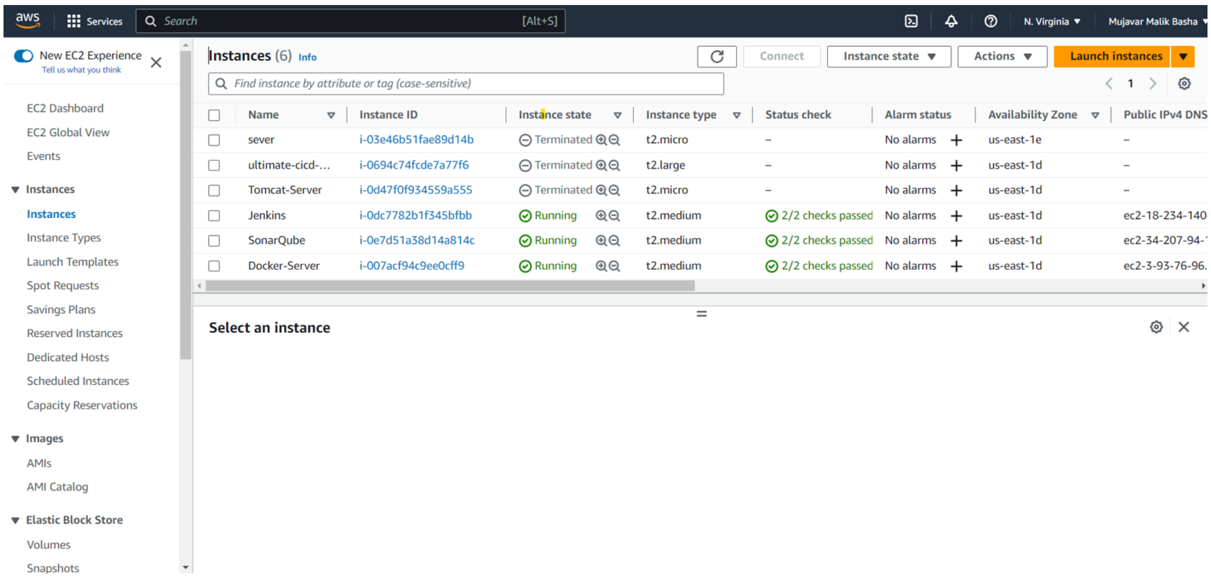
Jenkins is built on Java, so you will need a compatible JRE installed on your system. Jenkins typically requires Java 8 or higher.

Steps:

Version Control System (VCS):
Choose a version control system (e.g., Git) and set up a repository to host your project's codebase. Developers will commit their code changes to this repository.

Continuous Integration (CI):
Set up the CI part of the pipeline to automate the building and testing of your application whenever new code changes are pushed to the VCS.

Jenkins Installation:
Install Jenkins on a server or a cloud-based instance. You can follow the official Jenkins documentation for installation instructions.



Instances (6) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
sever	i-03e46b51fae89d14b	Terminated	t2.micro	–	No alarms	us-east-1e	–
ultimate-cicd-...	i-0694c74fcde7a77f6	Terminated	t2.large	–	No alarms	us-east-1d	–
Tomcat-Server	i-0d47f0f934559a555	Terminated	t2.micro	–	No alarms	us-east-1d	–
Jenkins	i-0dc7782b1f345bfbb	Running	t2.medium	2/2 checks passed	No alarms	us-east-1d	ec2-18-234-140
SonarQube	i-0e7d51a38d14a814c	Running	t2.medium	2/2 checks passed	No alarms	us-east-1d	ec2-34-207-94-
Docker-Server	i-007acf94c9ee0cff9	Running	t2.medium	2/2 checks passed	No alarms	us-east-1d	ec2-3-93-76-96.

Select an instance

Project Configuration:

Create a new Jenkins project (freestyle, pipeline, or multibranch pipeline) that corresponds to your application.

Source Code Management:

Configure Jenkins to connect to your version control system and specify the repository URL.

Build Triggers:

Set up triggers to automatically start the build process whenever changes are detected in the repository (e.g., polling for changes or using webhooks).

Build Step:

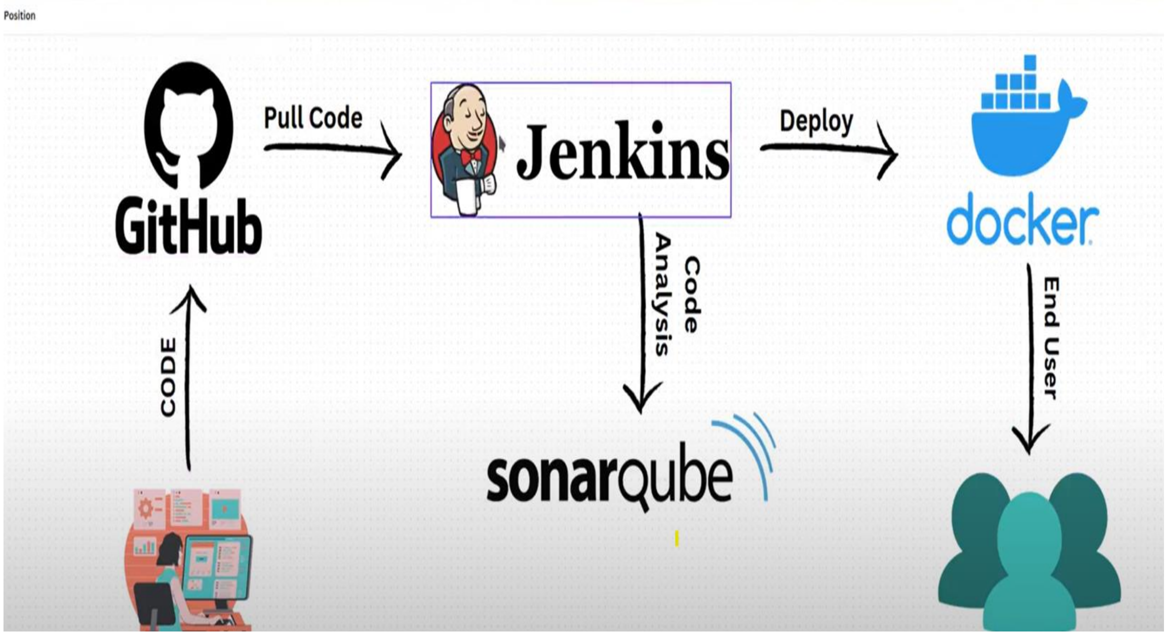
Define the build step that fetches the latest code, compiles the application, and generates the required artifacts.

Testing:

Integrate automated testing into the pipeline. You can use testing frameworks and tools suitable for your application's technology stack.

Artifacts:

Store the generated artifacts (e.g., binary files, executable packages) for later stages.



Continuous Deployment (CD):
Configure the CD part of the pipeline to automate the deployment of your application to various environments (e.g., staging, production) after successful testing.

Environment Configuration:
Set up the necessary environments where you want to deploy your application (e.g., staging, production servers).

Deployment Step:
Define the deployment step in your Jenkins pipeline that takes the tested artifacts and deploys them to the appropriate environment.

Rollback Mechanism:
Implement a rollback mechanism in case there are issues with the deployment.

Monitoring and Notifications:
Integrate monitoring tools to keep track of the application's performance and health in production. Set up notifications (e.g., email, Slack) to alert the team if something goes wrong.

Pipeline Customization:

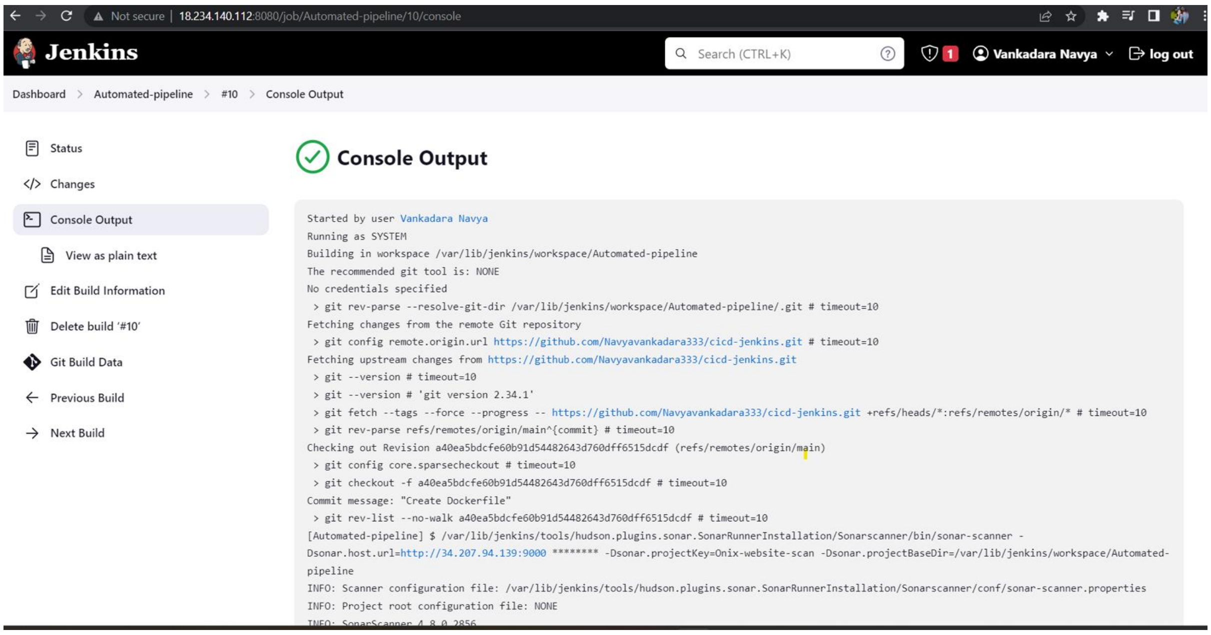
Customize your CI/CD pipeline to suit your specific needs. You may need to add additional stages, integrate with other tools, or implement security measures like access controls and secrets management.

Documentation and Training:
Ensure that the CI/CD pipeline is well-documented and provide training to team members on how to use and contribute to the pipeline effectively.

Continuous Improvement:
Regularly review and improve your CI/CD pipeline based on feedback, changing requirements, and modern technologies.

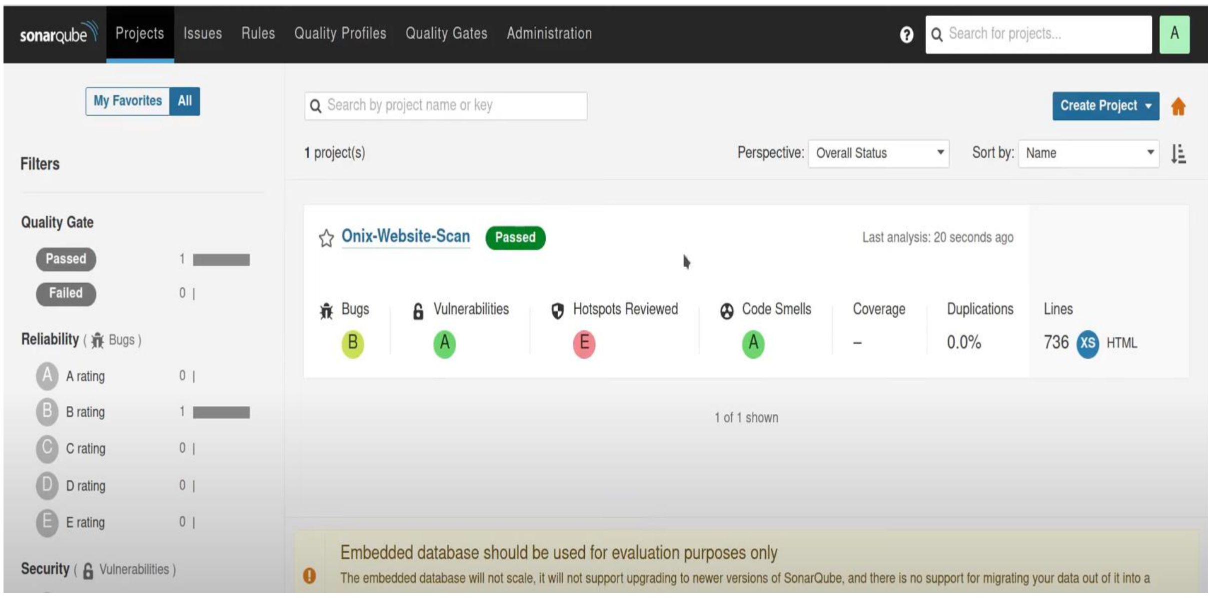
PROOF OF CONCEPT:

So, by using these Jenkins pipeline and with the help of the SonarQube which tested the vulnerability of the source code and finally we have deployed our source code and the Jenkins continuously tested the integrated changes and very responsive to the made changes.



```
Started by user Vankadara Navya
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Automated-pipeline
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Automated-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/Navyavankadara333/cicd-jenkins.git # timeout=10
Fetching upstream changes from https://github.com/Navyavankadara333/cicd-jenkins.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- https://github.com/Navyavankadara333/cicd-jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision a40ea5bdcfe60b91d54482643d760dff6515dcdf (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f a40ea5bdcfe60b91d54482643d760dff6515dcdf # timeout=10
Commit message: "Create Dockerfile"
 > git rev-list --no-walk a40ea5bdcfe60b91d54482643d760dff6515dcdf # timeout=10
[Automated-pipeline] $ /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/Sonarscanner/bin/sonar-scanner -
Dsonar.host.url=http://34.207.94.139:9000 ******** -Dsonar.projectKey=Onix-website-scan -Dsonar.projectBaseDir=/var/lib/jenkins/workspace/Automated-
pipeline
INFO: Scanner configuration file: /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/Sonarscanner/conf/sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.8.0.2856
```

This SonarQube is a tool it analysed our source code and provided a quality report.



Filters

Quality Gate

Passed	1	
Failed	0	

Reliability (Bugs)

A	A rating	0
B	B rating	1
C	C rating	0
D	D rating	0
E	E rating	0

Security (Vulnerabilities)

1 project(s)

Perspective: Overall Status

Sort by: Name

Onix-Website-Scan Passed

Last analysis: 20 seconds ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
B	A	E	A	–	0.0%	736 XS HTML

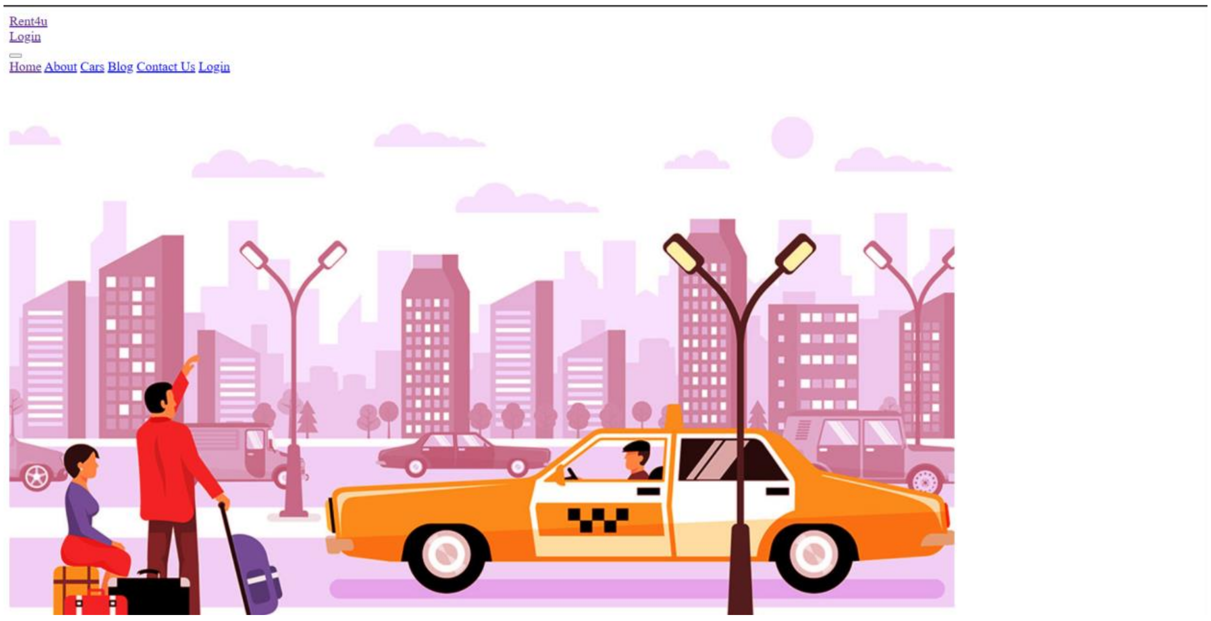
1 of 1 shown

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a

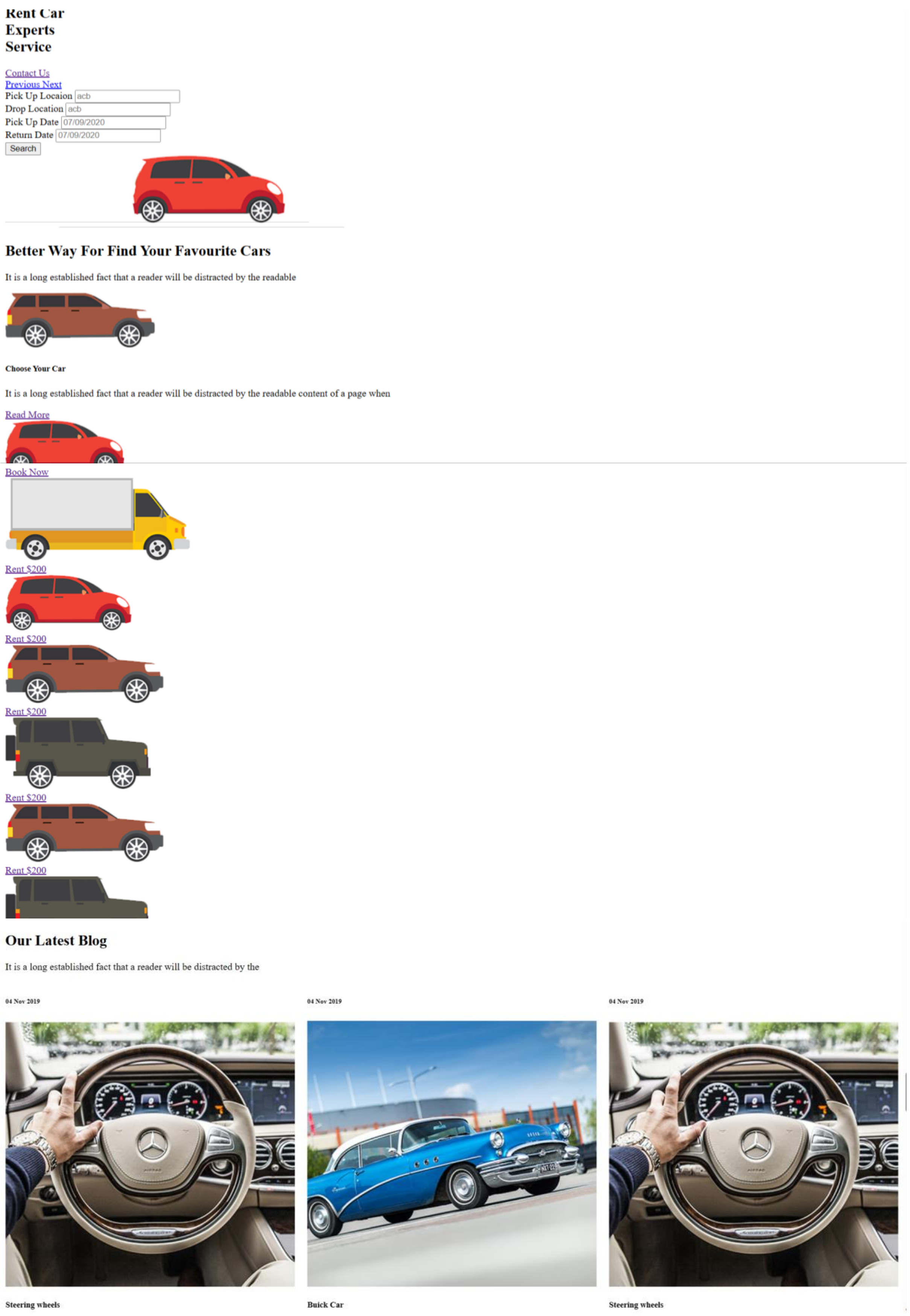
OUTCOME:

So, this is the application which we have deployed using the Jenkins pipeline.



Rent4u
Login

Home About Cars Blog Contact Us Login



Rent Car
Experts
Service

Contact Us
Previous Next
Pick Up Locaion acb
Drop Location acb
Pick Up Date 07/09/2020
Return Date 07/09/2020
Search

Better Way For Find Your Favourite Cars

It is a long established fact that a reader will be distracted by the readable

Choose Your Car

It is a long established fact that a reader will be distracted by the readable content of a page when

Read More

Book Now

Rent $200

Rent $200

Rent $200

Rent $200

Rent $200

Our Latest Blog

It is a long established fact that a reader will be distracted by the

04 Nov 2019

Steering wheels

04 Nov 2019

Buick Car

04 Nov 2019

Steering wheels

CONCLUSION:

In conclusion, implementing a CI/CD pipeline using Jenkins can provide many benefits, including improved scalability, reliability, and efficiency.

Jenkins has been successfully set up as a CI server, automating the build and testing process for the project. Developers can easily integrate their code changes into the main codebase, triggering automated builds and tests.

The CI/CD pipeline, coupled with SonarQube's feedback on code quality, has fostered a culture of continuous improvement. Developers receive immediate feedback on their changes, leading to better collaboration and knowledge sharing within the team.

With Docker containers, the application's deployment becomes scalable and reproducible across different environments. This reduces the risk of deployment-related issues and simplifies the process of scaling the application to handle increased loads.

In conclusion, the CI/CD pipeline project using Jenkins, SonarQube, and Docker has been a success. It has transformed the development and deployment processes, resulting in faster, more reliable, and higher-quality software delivery. The implementation of best practices for automation, testing, and code analysis has set the foundation for continuous improvement, enabling the team to respond quickly to changing requirements and deliver value to users efficiently.