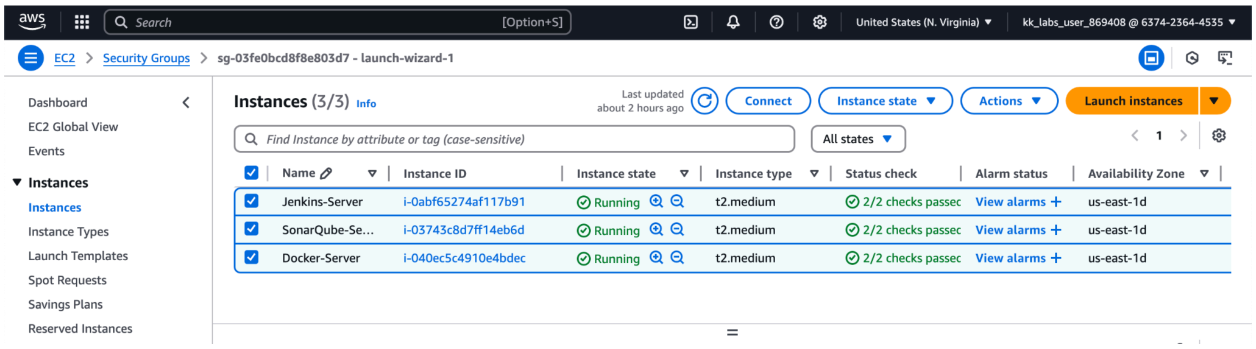
CI/CD Pipeline with GitHub, Jenkins, SonarQube, and Docker on AWS

Overview:

This project demonstrates the implementation of a robust CI/CD (Continuous Integration and Continuous Deployment) pipeline leveraging Jenkins, SonarQube, Docker, and AWS infrastructure. The setup is designed to streamline code integration, automate testing, and deploy containerized applications efficiently.

Architecture:

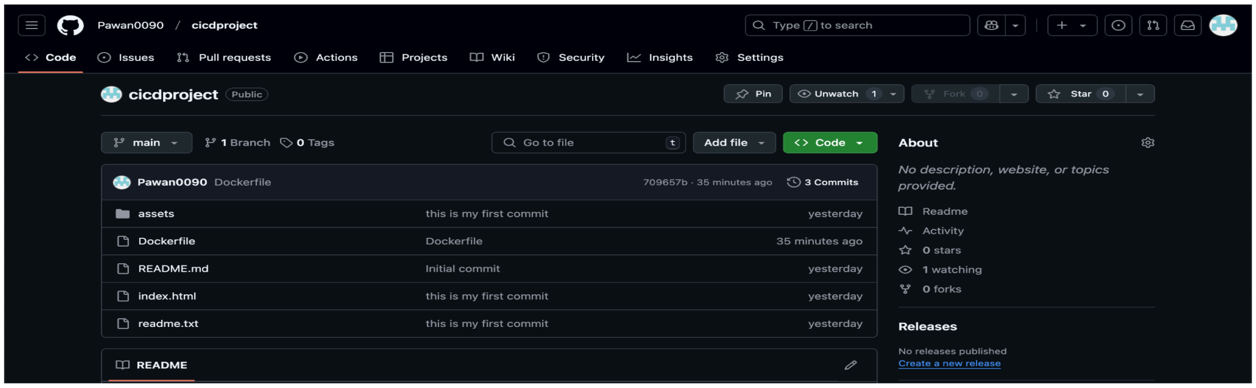
1. **AWS Infrastructure:**
   - **Three servers are deployed on AWS:**
     - Jenkins Server: Acts as the CI/CD orchestrator.
     - SonarQube Server: For static code analysis and quality checks.
     - Application Server: Hosts the deployed Docker container.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Jenkins-Server	i-0abf65274af117b91	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1d
SonarQube-Se...	i-03743c8d7ff14eb6d	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1d
Docker-Server	i-040ec5c4910e4bdec	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1d

GitHub Integration:

- The source code is hosted on a GitHub repository.
- A webhook is configured to trigger Jenkins jobs upon code changes (e.g., pushes or pull requests).

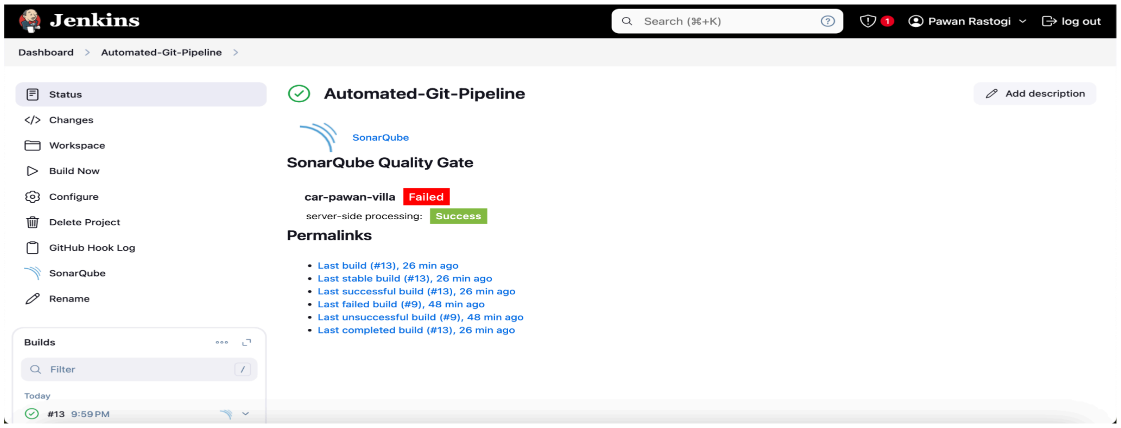


Pawan0090 Dockerfile 709657b · 35 minutes ago 3 Commits

assets	this is my first commit	yesterday
Dockerfile	Dockerfile	35 minutes ago
README.md	Initial commit	yesterday
index.html	this is my first commit	yesterday
readme.txt	this is my first commit	yesterday

Jenkins Pipeline:

- **Code Fetching**: Jenkins pulls the latest code from the GitHub repository.
- **Code Analysis**: Jenkins integrates with SonarQube to perform static code analysis, ensuring code quality and compliance with standards.
- **Dockerization**:
    - A Docker image of the website is created from the source code using a Dockerfile.
    - The Docker image is tagged and pushed to a container registry (e.g., Docker Hub).
- **Deployment**:
    - The Docker container is deployed on the application server, ensuring the latest version of the website is live.



Automated-Git-Pipeline

SonarQube Quality Gate

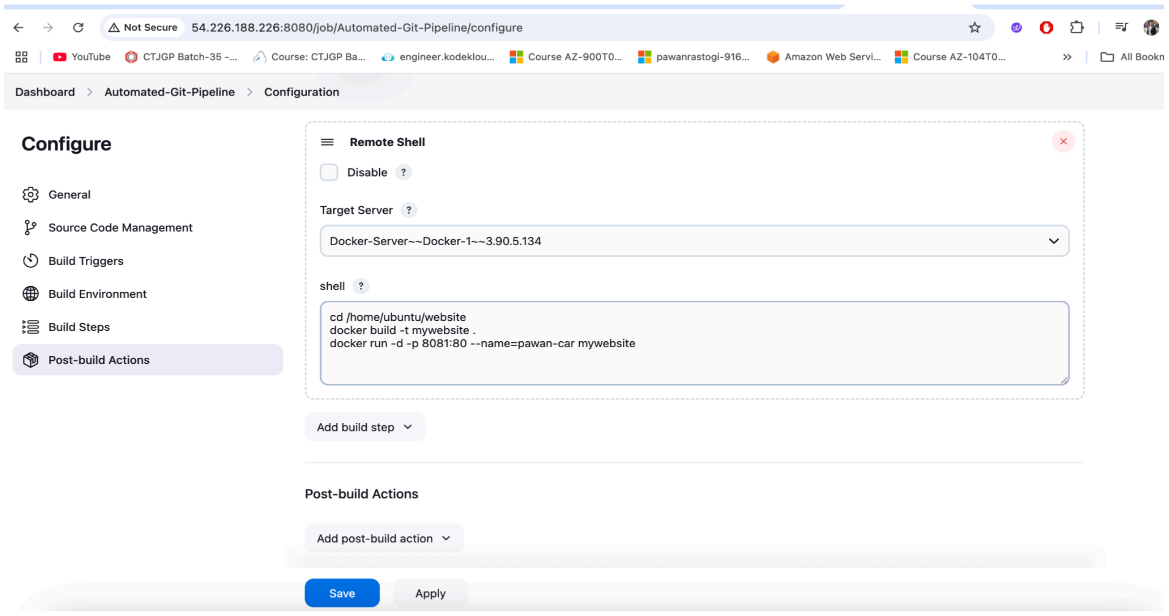
car-pawan-villa Failed
server-side processing: Success

Permalinks

- Last build (#13), 26 min ago
- Last stable build (#13), 26 min ago
- Last successful build (#13), 26 min ago
- Last failed build (#9), 48 min ago
- Last unsuccessful build (#9), 48 min ago
- Last completed build (#13), 26 min ago

#13 9:59 PM

In the image below I used a remote shell to execute the command on my docker container. The command below is basically building the image from the docker file -(mywebsite).



Remote Shell

Disable

Target Server: Docker-Server~~Docker-1~~3.90.5.134

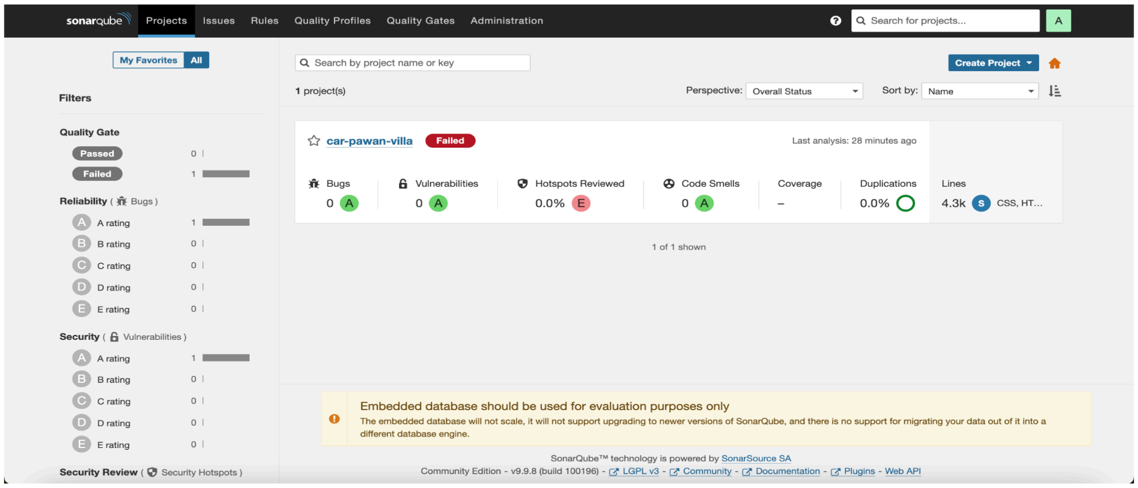
shell

```
cd /home/ubuntu/website
docker build -t mywebsite .
docker run -d -p 8081:80 --name=pawan-car mywebsite
```

Post-build Actions

SonarQube Integration:

- SonarQube is configured with Jenkins to perform detailed code analysis, providing insights into code smells, bugs, and vulnerabilities.



1 project(s)

Quality Gate	
Passed	0
Failed	1

Reliability (Bugs)	
A rating	1
B rating	0
C rating	0
D rating	0
E rating	0

Security (Vulnerabilities)	
A rating	1
B rating	0
C rating	0
D rating	0
E rating	0

Security Review (Security Hotspots)

car-pawan-villa Failed — Last analysis: 28 minutes ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	0.0% E	0 A	–	0.0%	4.3k S CSS, HT...

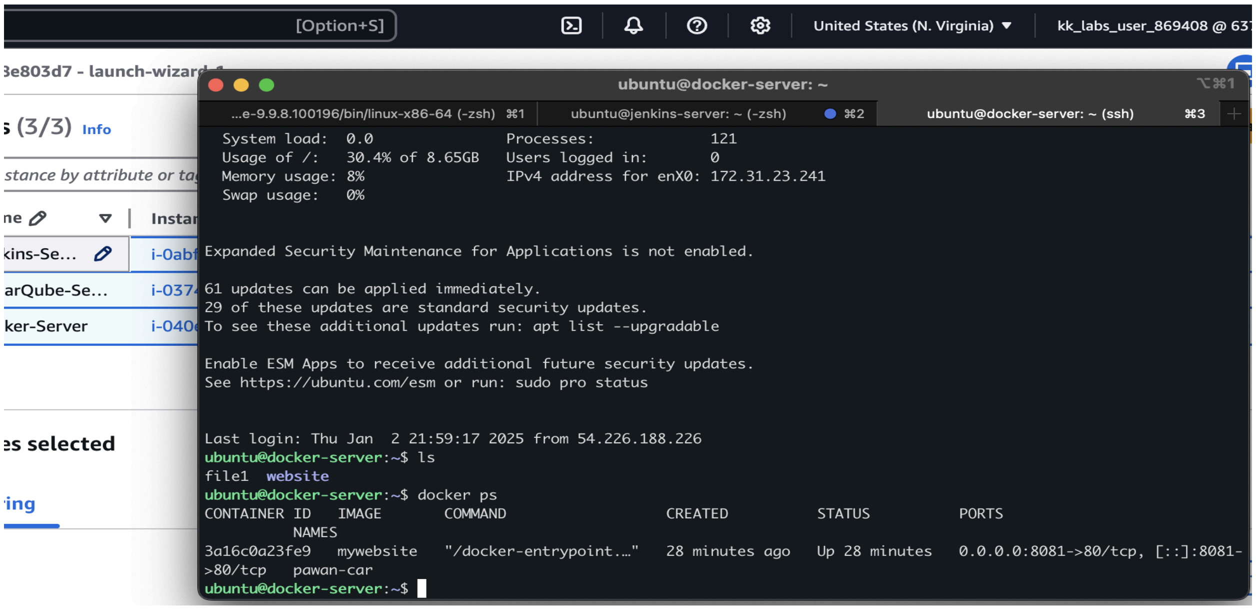
1 of 1 shown

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 different database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - v9.9.8 (build 100196) - LGPL v3 - Community - Documentation - Plugins - Web API

Containerization and Deployment:

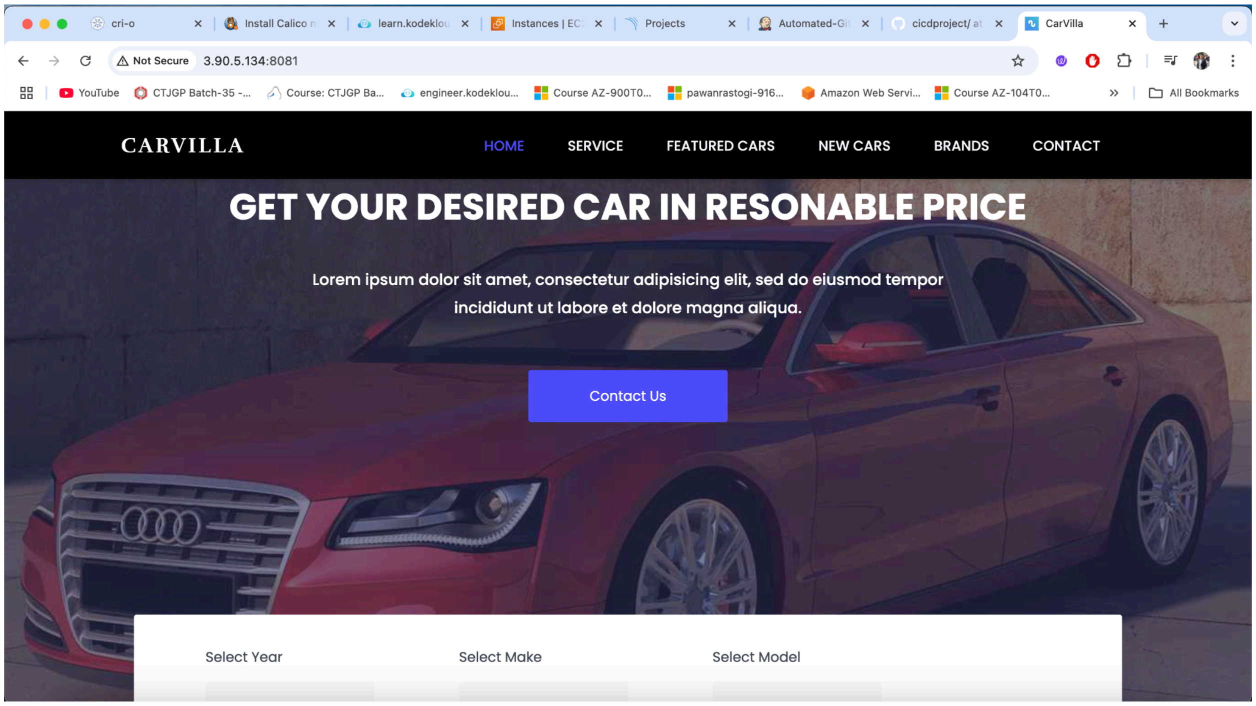
- Docker ensures portability and consistency across environments.
- The website is packaged as a container and deployed seamlessly, reducing manual intervention and deployment errors.
- The image below shows that docker file mywebsite got built and the container started running.



```
 System load:  0.0               Processes:             121
 Usage of /:   30.4% of 8.65GB   Users logged in:       0
 Memory usage: 8%                IPv4 address for enX0: 172.31.23.241
 Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

61 updates can be applied immediately.
29 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Thu Jan  2 21:59:17 2025 from 54.226.188.226
ubuntu@docker-server:~$ ls
file1  website
ubuntu@docker-server:~$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS
          NAMES
3a16c0a23fe9   mywebsite   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   0.0.0.0:8081->80/tcp, [::]:8081-
>80/tcp   pawan-car
ubuntu@docker-server:~$
```

However, I have done port forwarding to make my website accessible to external users, which runs on port 80 inside the container. For accessing my website, I must specify the IP address of my base machine: port number. In my case it is **3.90.5.134:8081**

Final Look of My-Web-Site.



Not Secure 3.90.5.134:8081
CARVILLA
HOME
SERVICE
FEATURED CARS
NEW CARS
BRANDS
CONTACT
GET YOUR DESIRED CAR IN RESONABLE PRICE
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Contact Us
Select Year
Select Make
Select Model

Benefits:

- Automated CI/CD Pipeline: Speeds up development and deployment cycles.
- Code Quality Assurance: Ensures the codebase meets quality and security standards via SonarQube.
- Scalable and Portable: Docker containers make it easy to scale and replicate the application across environments.
- Infrastructure as a Service (IaaS): Leveraging AWS for hosting provides scalability, reliability, and cost efficiency.

Conclusion:

This project integrates modern DevOps practices to build an end-to-end CI/CD pipeline. By combining AWS's scalability, Jenkins's automation capabilities, SonarQube's code quality assurance, and Docker's portability, the system ensures rapid, reliable, and consistent deployments of the website application.