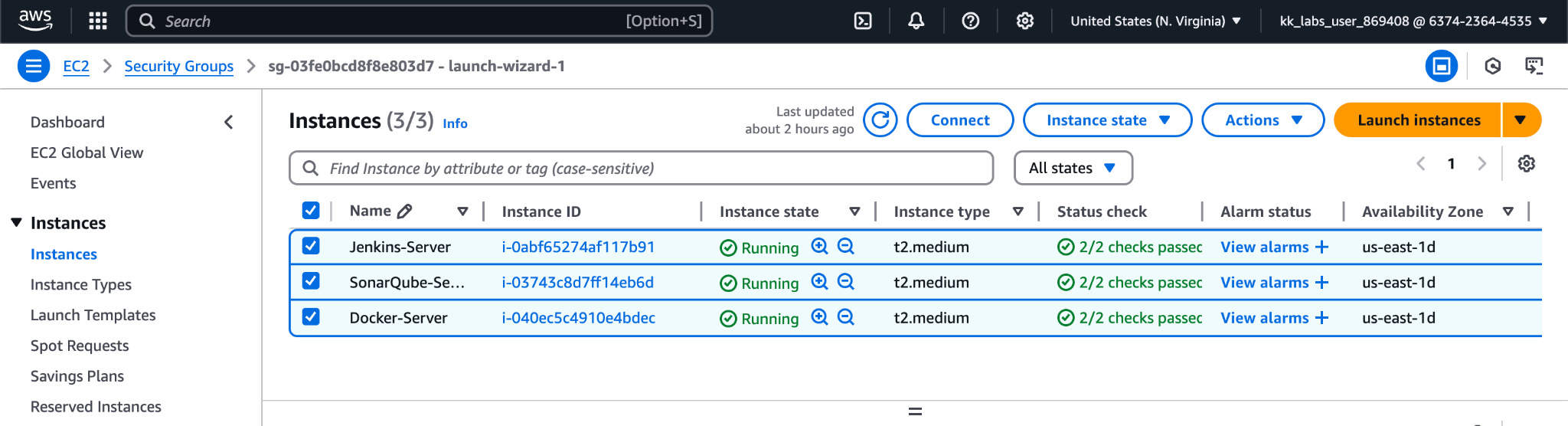
**CI/CD Pipeline with GitHub, Jenkins, SonarQube, and Docker on AWS**

**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.

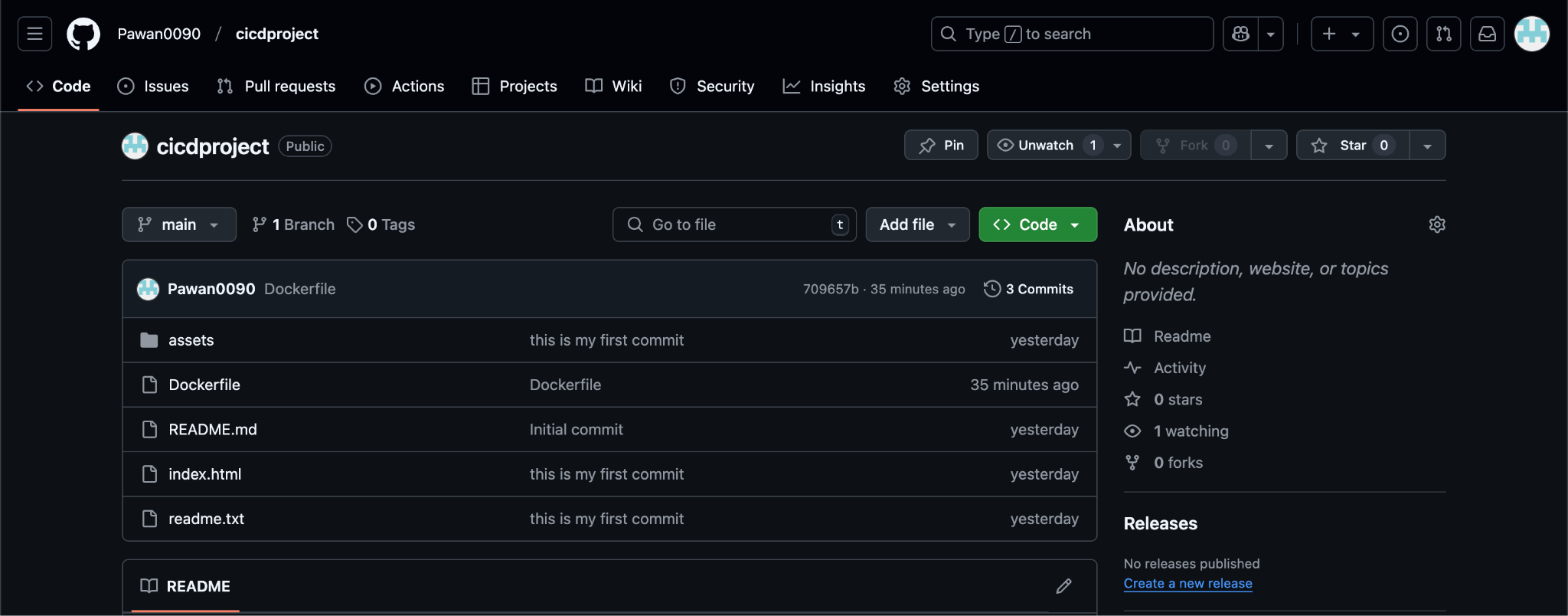
**Architecture:**

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.

****

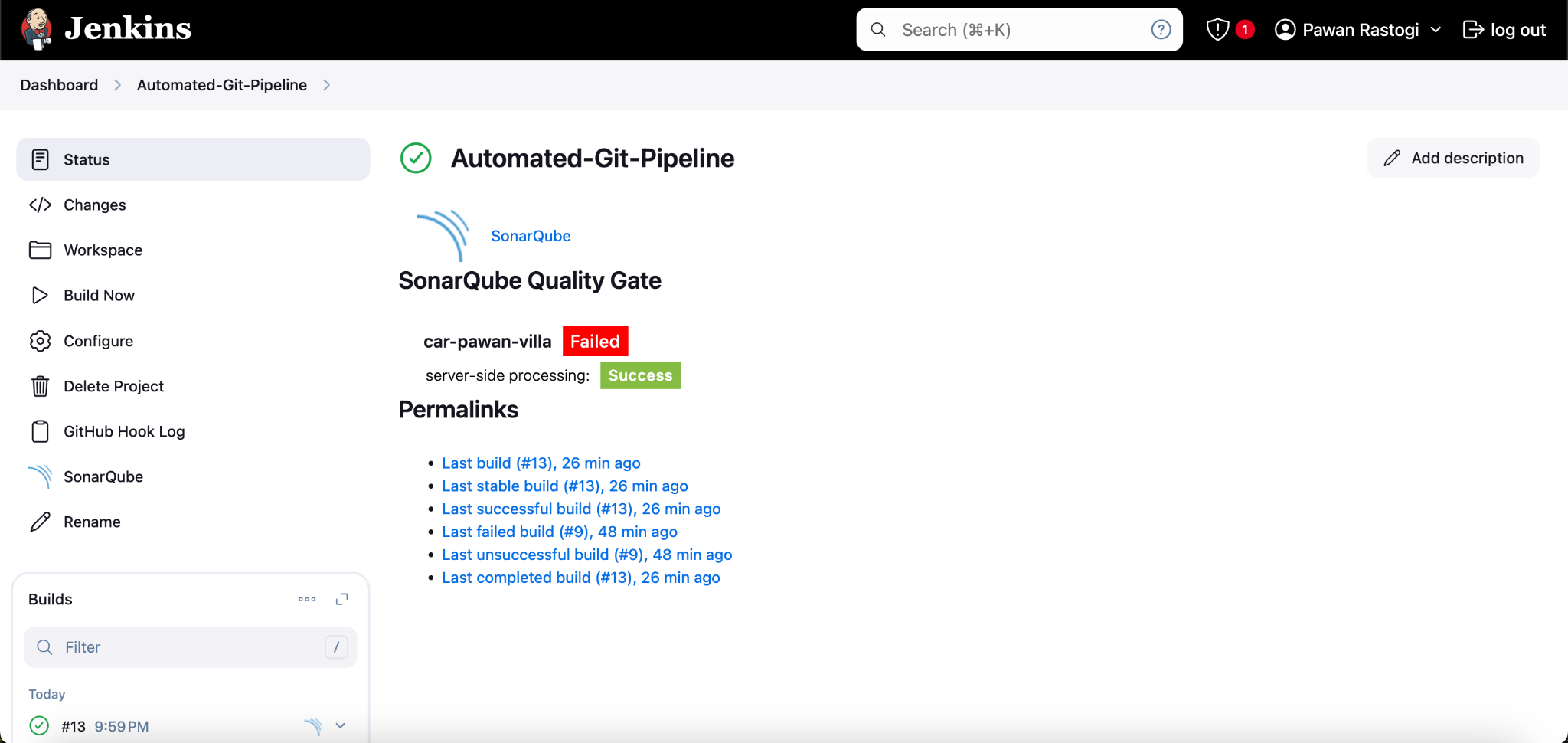
**GitHub Integration:**

* 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).

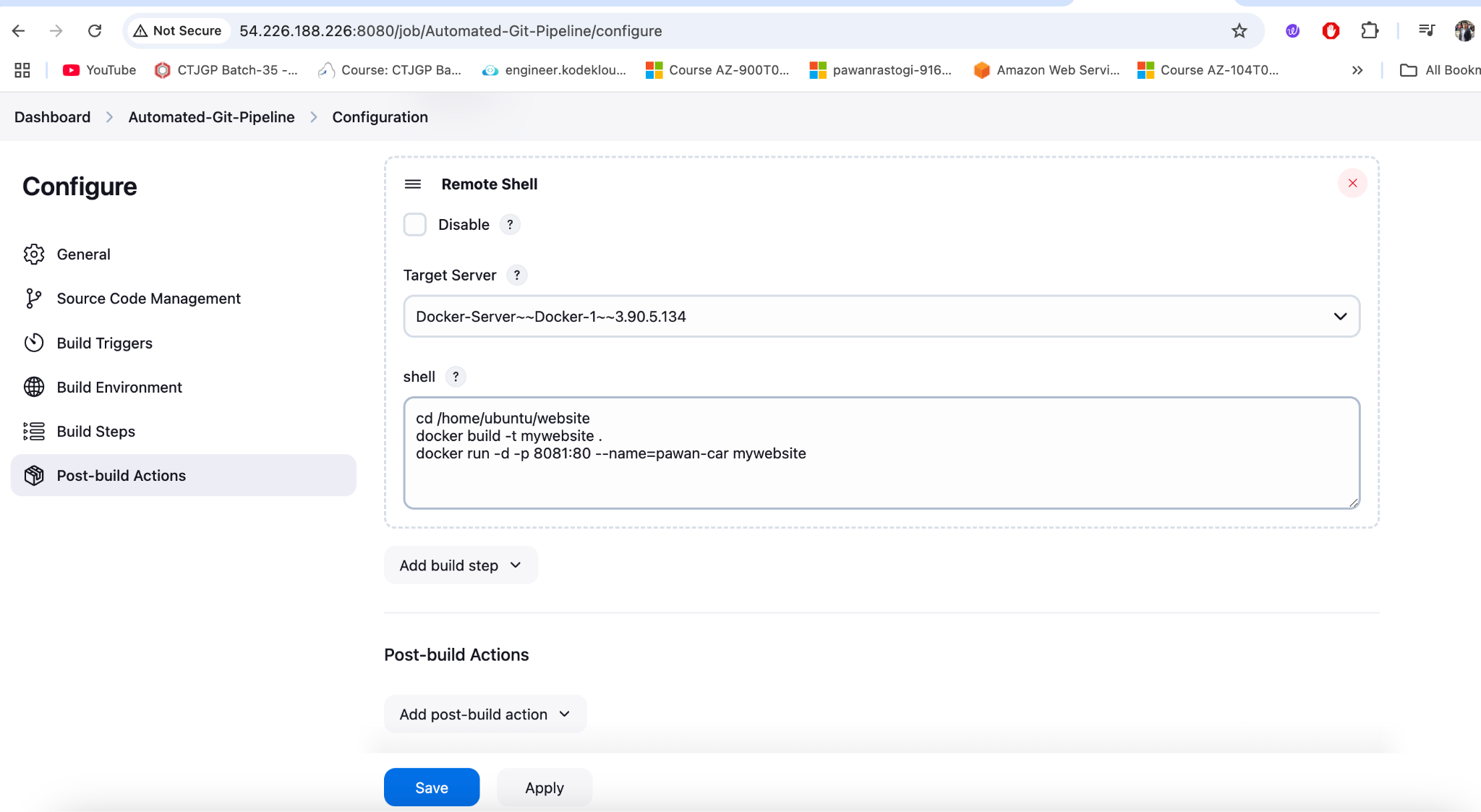


**Jenkins Pipeline**:

* **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.

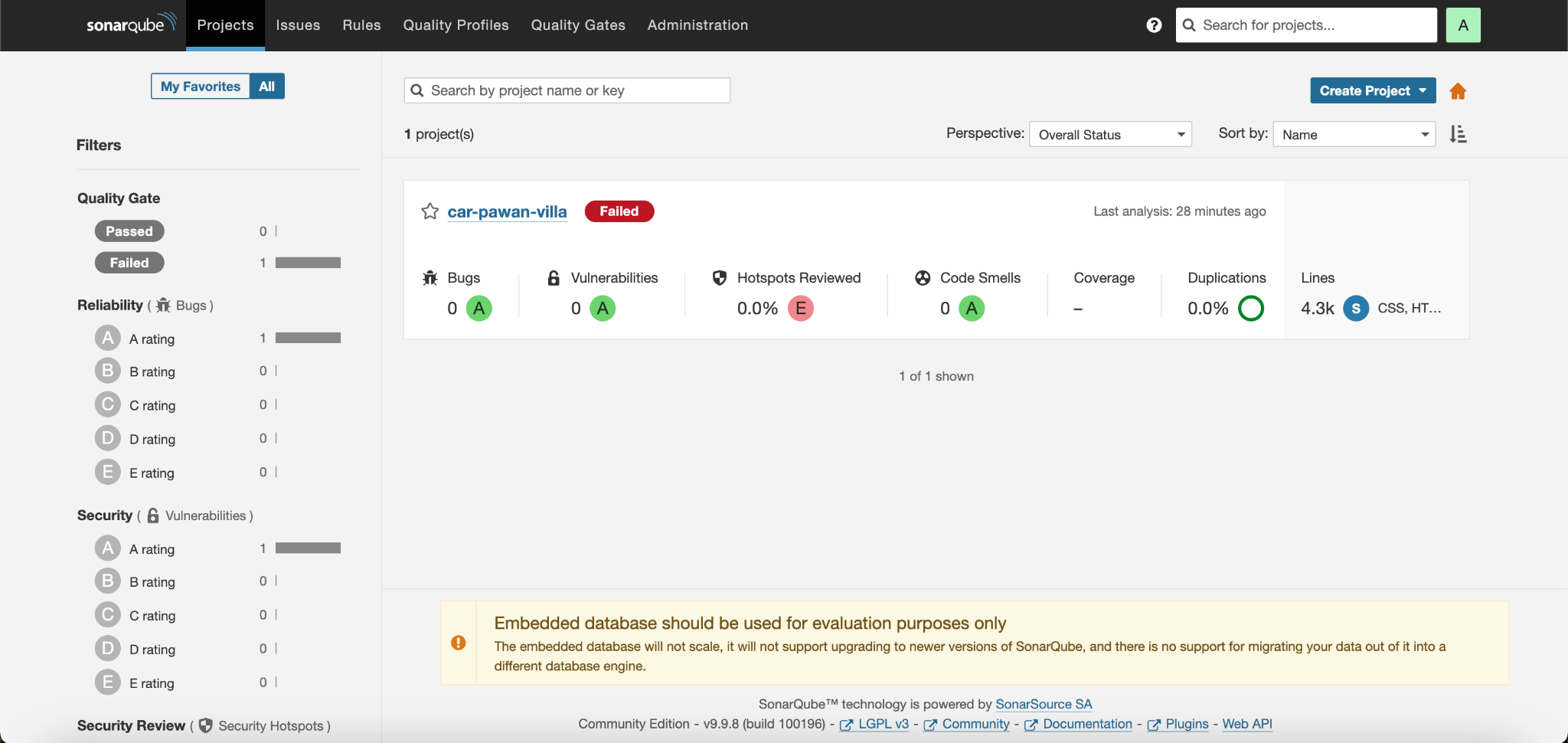


In the image below I used a remote shell to execute the command on my docker container. The command below is basically building the image from the docker file -(mywebsite).



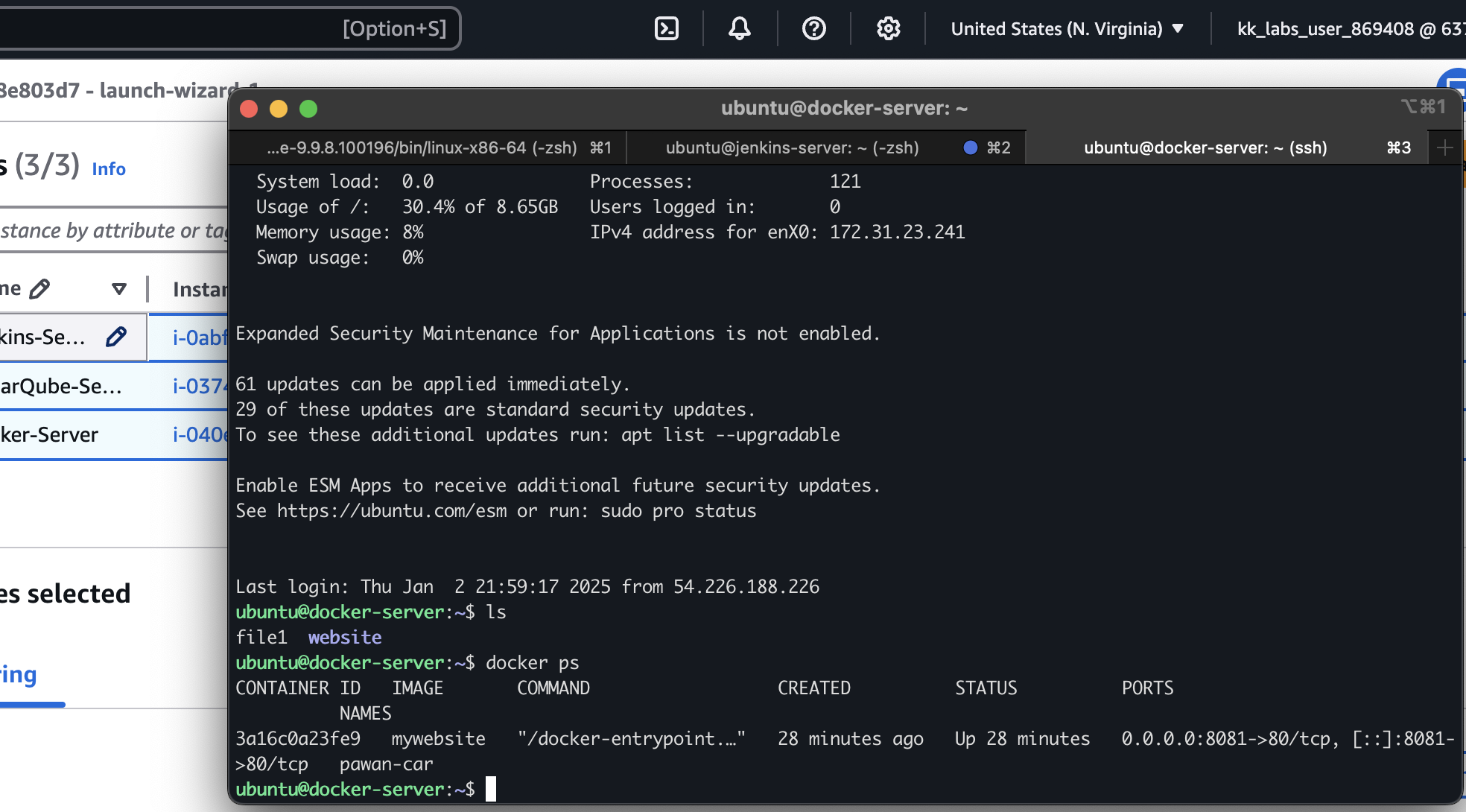
**SonarQube Integration**:

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



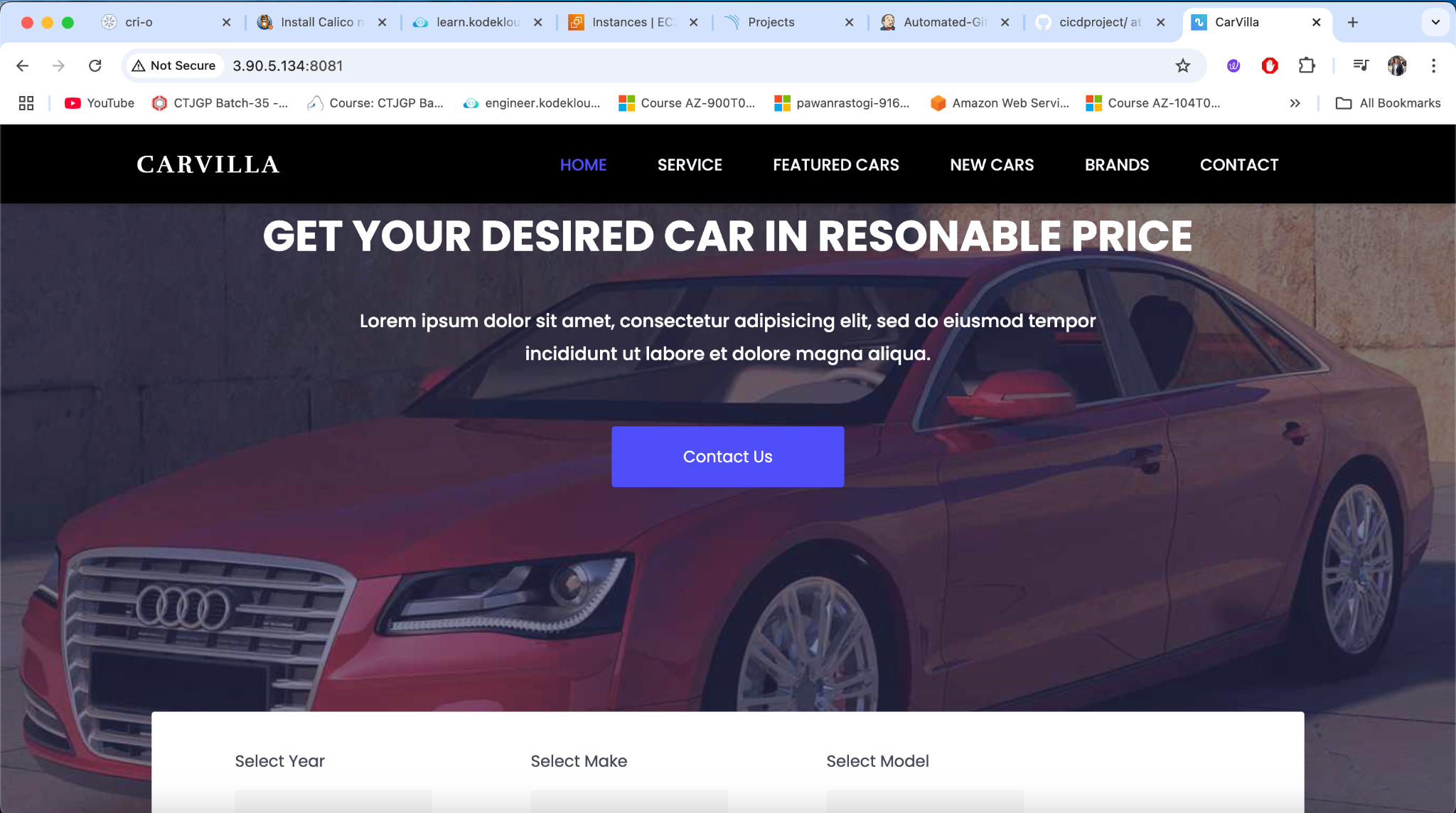
**Containerization and Deployment**:

* 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.



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.**

****

**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.