RAMAIAH INSTITUTE OF TECHNOLOGY

MSR NAGAR, BENGALURU, 560054

Report on

CI / CD PIPELINE USING JENKINS

Submitted in partial fulfilment of the other component requirements as a part of the Devops Lab course with code ISL66 for the Semester 6 of degree of Bachelor of Engineering in Information Science And Engineering

Submitted By

Karthik B (1MS22IS060)

Kartik J H (1MS22IS062)

Under the Guidance of

Mrs. J R Shruti

Assistant Professor

Department of Information Science And Engineering

Ramaiah Institute of Technology

2024 - 2025

DevOps Pipeline Report

Introduction

This project implements a complete CI/CD pipeline for a weather forecasting web application using Jenkins. The pipeline is written in a declarative syntax within a Jenkinsfile and automates all critical stages including code checkout, vulnerability scanning, static code analysis, Docker image creation, and cloud deployment. This ensures that every change is delivered reliably and securely.

The weather application is a frontend-based project built with HTML, CSS, and JavaScript. It allows users to retrieve and view real-time weather data by interacting with a clean and responsive interface. By consuming weather data from external APIs, the app presents temperature, conditions, and other useful information in an intuitive way.

To streamline the software delivery lifecycle, this pipeline integrates tools like Git, Trivy, SonarQube, Docker, and Render, enabling automated build and deployment with minimal human intervention. Each stage of the pipeline enhances code quality, security, and deployment consistency, ensuring the weather app is always up-to-date and production-ready.

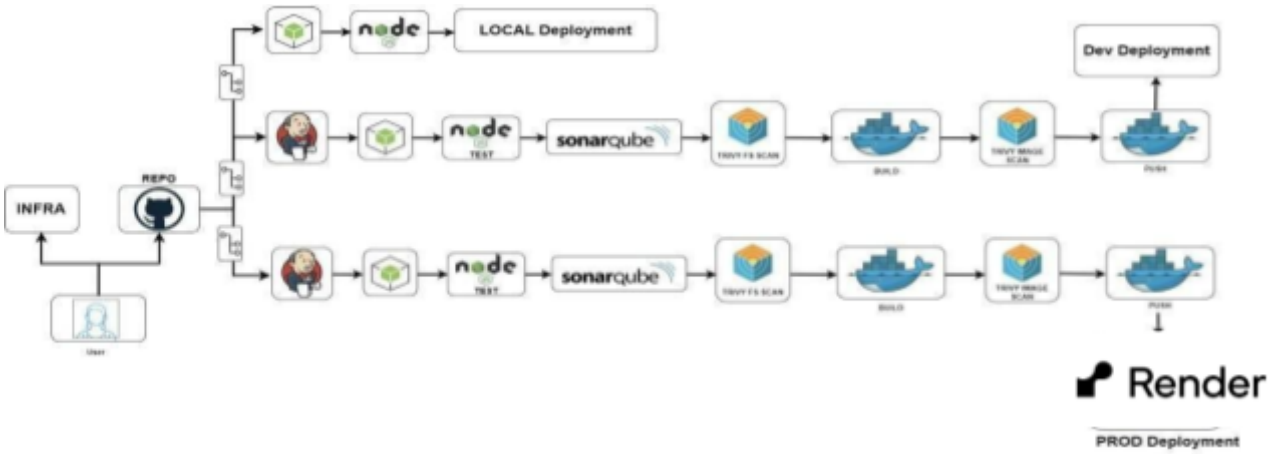
Pipeline Configuration Overview

This report outlines the CI/CD pipeline for the Weather Forecasting Web Application hosted on GitHub. The pipeline automates key stages such as source code checkout, file system vulnerability scanning, static code analysis, Docker image building, container image scanning, image publishing to Docker Hub, and optional deployment to the cloud using Render. The setup reflects a complete and efficient DevOps workflow that ensures secure and consistent application delivery.

The pipeline begins by pulling the latest source code from the GitHub repository. Once the code is checked out, Trivy is used to scan the local file system for sensitive files or security issues before the application is built. SonarQube is then triggered to perform static code analysis, highlighting code quality issues, bugs, or vulnerabilities in the JavaScript, CSS, and HTML codebase.

After quality checks, Docker builds a container image of the application using the Dockerfile. This image is further scanned by Trivy to detect vulnerabilities in the system layers and embedded libraries. If the scans pass, the image is pushed to Docker Hub using credentials managed in Jenkins.

Finally, the latest Docker image can be deployed to the Render cloud platform. The pipeline ensures each change is tested, verified, and deployed with minimal manual effort, maintaining both code quality and deployment consistency across environments.

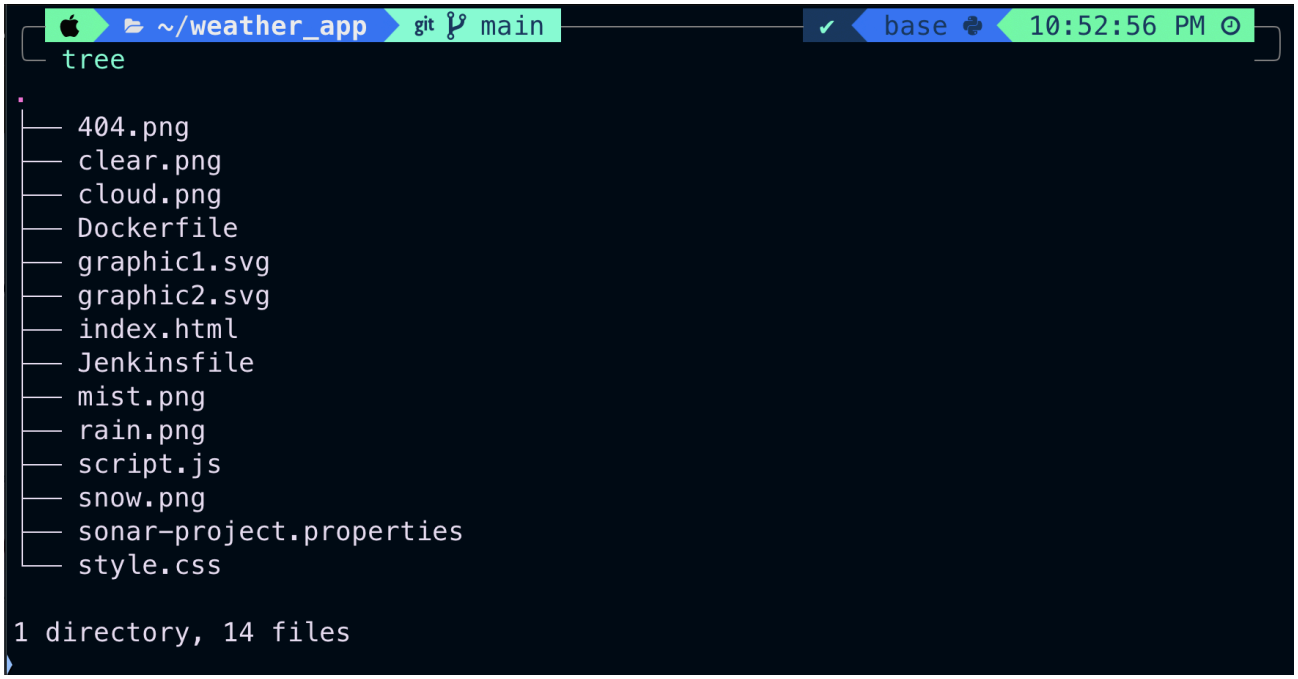


Tools and Technologies Used

- **Git**: A distributed version control system used to manage and track changes in the source code stored in a GitHub repository.
- **Jenkins**: An open-source automation server that orchestrates the continuous integration and delivery (CI/CD) pipeline, automating tasks like testing, scanning, building, and deploying.
- **Trivy**: A security scanner used to detect vulnerabilities in both the project's file system and Docker images to ensure secure software delivery.
- **SonarQube**: A static code analysis tool that inspects the HTML, CSS, and JavaScript codebase to identify bugs, code smells, and security issues.
- **Docker**: A containerization platform used to package the application and its dependencies into a standardized unit, ensuring consistent environments across development and production.
- **Docker Hub**: A cloud-based registry where the built Docker images are stored and shared for later deployment or collaboration.
- **Render**: A modern cloud platform used to deploy and host the final web application in a fully managed environment.

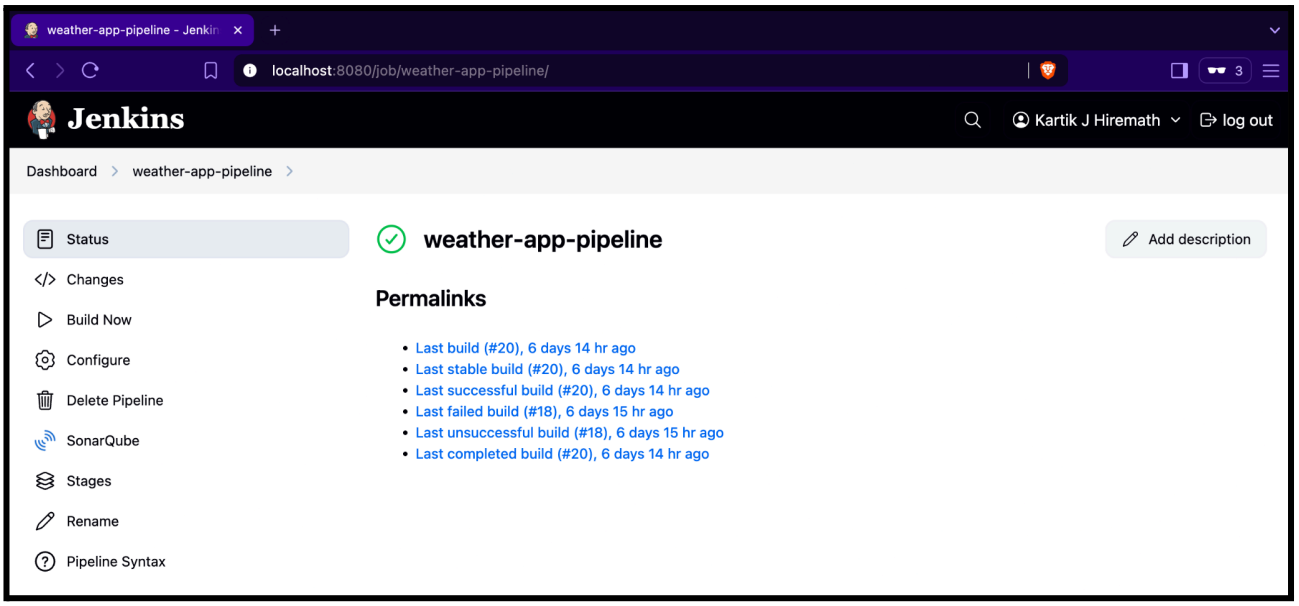
Project Setup and Directory Structure

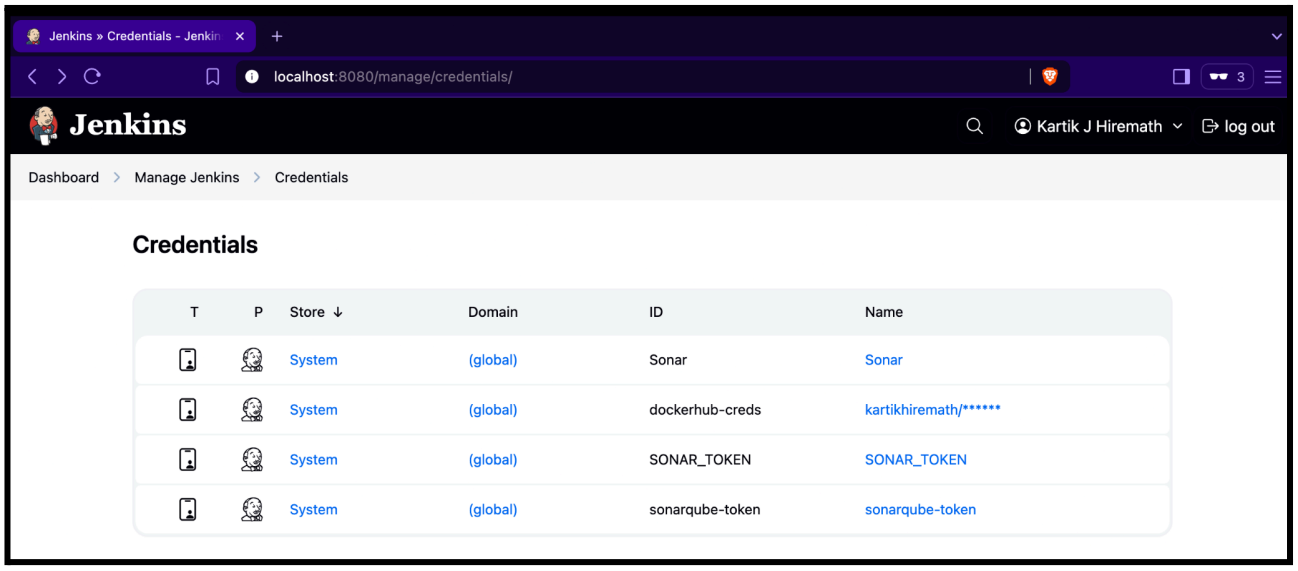
The weather application project is organized in a single root directory. Key files include the Dockerfile for building the container image, the Jenkinsfile for defining the CI/CD pipeline, and the frontend assets: index.html, style.css, and script.js. A sonar-project.properties file configures SonarQube analysis. Image and icon files such as clear.png, rain.png, snow.png, mist.png, cloud.png, and 404.png are stored alongside the code for use in the user interface. This layout enables Jenkins to clone the repository, run scans, build the image, and deploy without moving or restructuring files.



Jenkins Setup and Configuration

Jenkins was installed on a macOS host using Homebrew and started via the Jenkins WAR file on port 8080. An admin account was created and recommended plugins were installed. Global tool configuration was updated to include the SonarScanner installation. Credentials for GitHub access, Docker Hub login, and the Render deploy hook were added to the Jenkins credentials store. A pipeline job was created that points to the Jenkinsfile in the weather_app GitHub repository.
During pipeline execution, the Checkout stage uses stored credentials to clone the main branch. SonarQube environment variables and Docker commands are executed on the Jenkins agent where Docker and Trivy are installed.





Pipeline Stages and Descriptions

1. **Checkout**

   This stage retrieves the latest source code from the GitHub repository. It uses configured credentials to clone the main branch, ensuring the pipeline always works with the most recent code.

```
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential dockerhub-creds
 > /opt/homebrew/bin/git rev-parse --resolve-git-dir /Users/kartikhiremath/.jenkins/workspace/weather-app-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
 > /opt/homebrew/bin/git config remote.origin.url https://github.com/Kartik-Hiremath/weather_app.git # timeout=10
Fetching upstream changes from https://github.com/Kartik-Hiremath/weather_app.git
 > /opt/homebrew/bin/git --version # timeout=10
 > git --version # 'git version 2.46.0'
using GIT_ASKPASS to set credentials
 > /opt/homebrew/bin/git fetch --tags --force --progress -- https://github.com/Kartik-Hiremath/weather_app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > /opt/homebrew/bin/git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision c93b7ae3a04a28cc81603167b510a020fc8091ba (refs/remotes/origin/main)
 > /opt/homebrew/bin/git config core.sparsecheckout # timeout=10
 > /opt/homebrew/bin/git checkout -f c93b7ae3a04a28cc81603167b510a020fc8091ba # timeout=10
 > /opt/homebrew/bin/git branch -a -v --no-abbrev # timeout=10
 > /opt/homebrew/bin/git checkout -b main c93b7ae3a04a28cc81603167b510a020fc8091ba # timeout=10
Commit message: "Update Jenkinsfile"
```

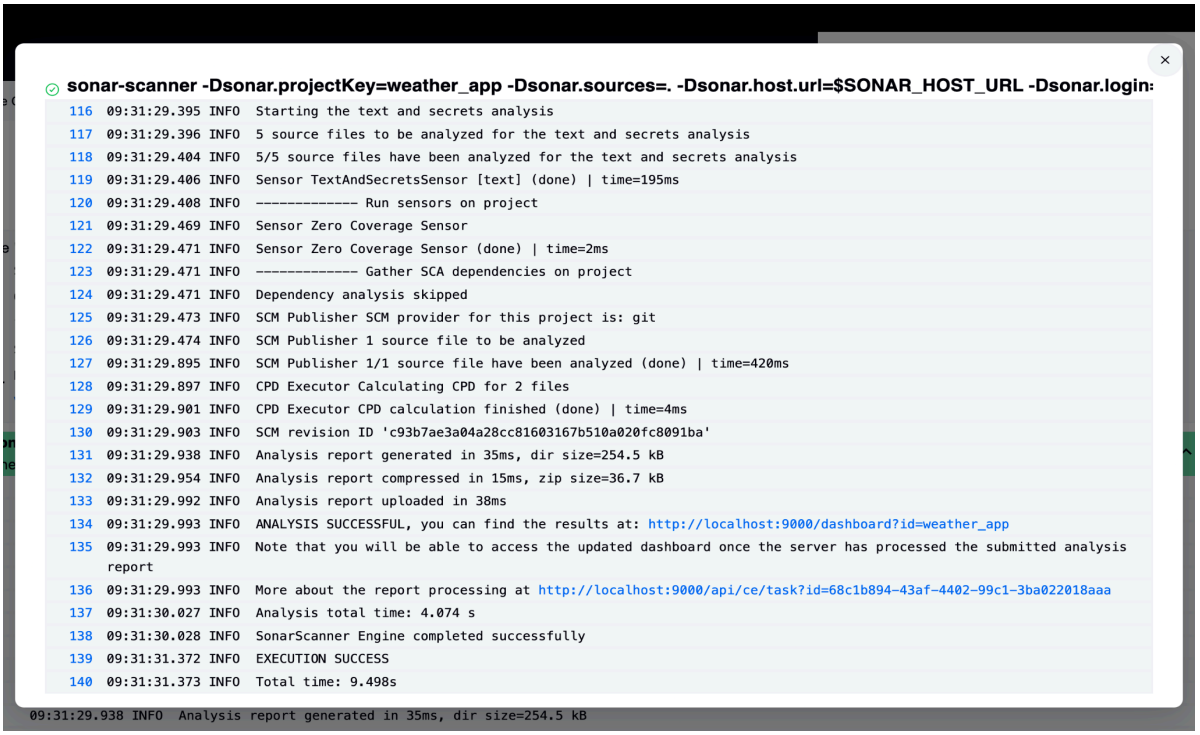
2. Trivy Filesystem Scan

Trivy scans the project directory for hardcoded secrets, vulnerable files, and misconfigurations before building. This early detection helps prevent issues from being packaged into the container.

```
+ trivy fs --exit-code 0 --severity LOW,MEDIUM,HIGH .
2025-06-12T09:31:20+05:30       INFO    [vuln] Vulnerability scanning is enabled
2025-06-12T09:31:20+05:30       INFO    [secret] Secret scanning is enabled
2025-06-12T09:31:20+05:30       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to
disable secret scanning
2025-06-12T09:31:20+05:30       INFO    [secret] Please see also
https://trivy.dev/v0.63/docs/scanner/secret#recommendation for faster secret detection
2025-06-12T09:31:20+05:30       INFO    Number of language-specific files       num=0
2025-06-12T09:31:20+05:30       WARN    [report] Supported files for scanner(s) not found.      scanners=[vuln]
2025-06-12T09:31:20+05:30       INFO    [report] No issues detected with scanner(s).    scanners=[secret]

Report Summary

┌────────┬──────┬─────────────────┬─────────┐
│ Target │ Type │ Vulnerabilities │ Secrets │
├────────┼──────┼─────────────────┼─────────┤
│   -    │  -   │        -        │    -    │
└────────┴──────┴─────────────────┴─────────┘
Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)
```

3. SonarQube Analysis

Static code analysis is performed on HTML, CSS, JavaScript, and the Dockerfile. SonarQube detects bugs, code smells, and security hotspots. Results are published to the SonarQube dashboard.



4. Build Docker Image

A Docker image is built from the source code using the Dockerfile. The image is tagged with the Jenkins build number for traceability.

```
+ docker build -t kartikhiremath/weather_app:latest .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 329B done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/nginx:alpine
#2 ...

#3 [auth] library/nginx:pull token for registry-1.docker.io
#3 DONE 0.0s

#2 [internal] load metadata for docker.io/library/nginx:alpine
#2 DONE 2.2s

#4 [internal] load .dockerignore
#4 transferring context: 2B done
#4 DONE 0.0s

#5 [1/3] FROM docker.io/library/nginx:alpine@sha256:65645c7bb6a0661892a8b03b89d0743208a18dd2f3f17a54ef4b76fb8e2f2a10
#5 resolve docker.io/library/nginx:alpine@sha256:65645c7bb6a0661892a8b03b89d0743208a18dd2f3f17a54ef4b76fb8e2f2a10 0.0s done
#5 DONE 0.0s

#6 [internal] load build context
#6 transferring context: 401.17kB 0.0s done
#6 DONE 0.0s

#7 [2/3] RUN rm -rf /usr/share/nginx/html/*
#7 CACHED

#8 [3/3] COPY . /usr/share/nginx/html/
#8 DONE 0.0s

#9 exporting to image
#9 exporting layers 0.0s done
#9 exporting manifest sha256:0633cd2df7abb202230d27354285fe9c76ee9903b8135925aba1f474296f9504
#9 exporting manifest sha256:0633cd2df7abb202230d27354285fe9c76ee9903b8135925aba1f474296f9504 done
#9 exporting config sha256:7848ddbc5aa453d8b5154b1d5fbf0f89b6d209922f154be24472d40add5f9af5 done
#9 exporting attestation manifest sha256:48ed2a415a57fbc3c505b0df493af66bf15ee0c34c3f6ae1b2662f1c79bc1cfd done
#9 exporting manifest list sha256:aec92fe25c7dd43ea1892e57a21459b7373702dc1bbffb2b03bd7a8c68350372 done
#9 naming to docker.io/kartikhiremath/weather_app:latest done
#9 unpacking to docker.io/kartikhiremath/weather_app:latest 0.0s done
#9 DONE 0.1s
```

5. Trivy Docker Image Scan

The newly built Docker image is scanned for known CVEs at the OS and library level. This ensures no critical vulnerabilities exist before pushing.

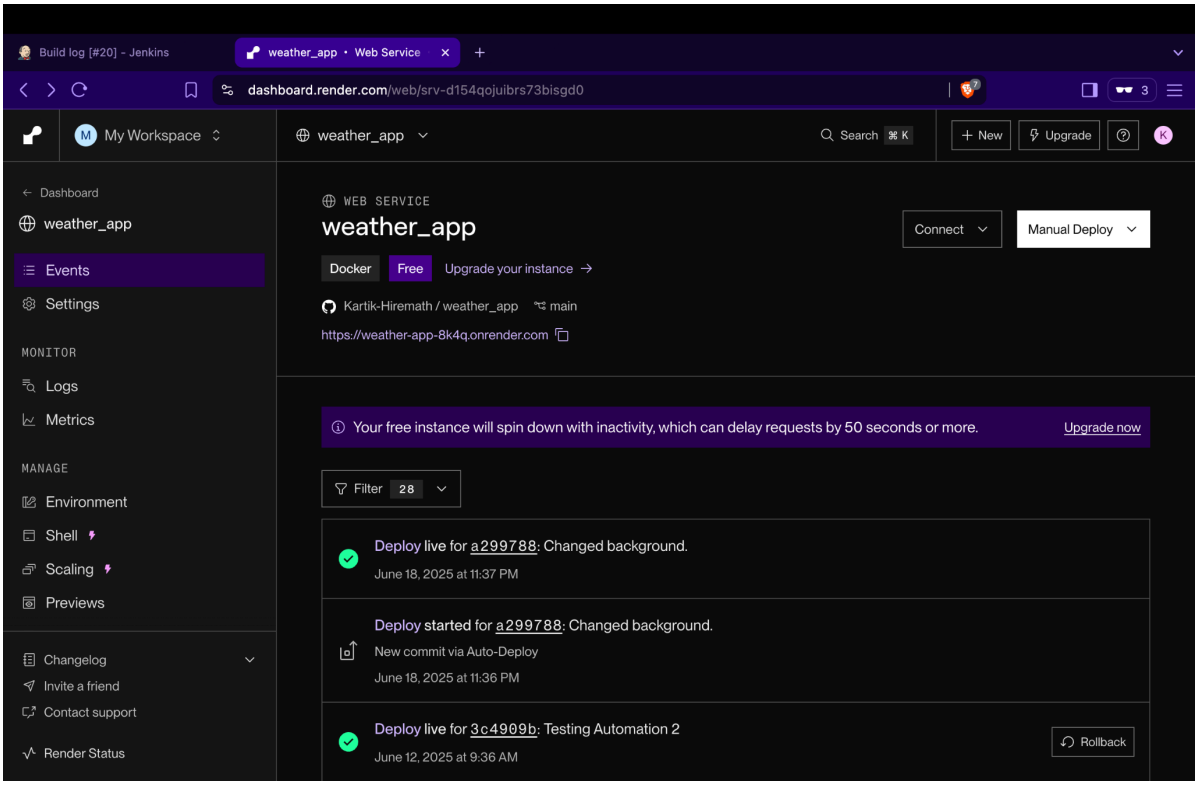
```
+ trivy image kartikhiremath/weather_app:latest
2025-06-12T09:31:37+05:30	INFO	[vuln] Vulnerability scanning is enabled
2025-06-12T09:31:37+05:30	INFO	[secret] Secret scanning is enabled
2025-06-12T09:31:37+05:30	INFO	[secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-06-12T09:31:37+05:30	INFO	[secret] Please see also https://trivy.dev/v0.63/docs/scanner/secret#recommendation for faster secret detection
2025-06-12T09:31:38+05:30	INFO	Detected OS	family="alpine" version="3.21.3"
2025-06-12T09:31:38+05:30	INFO	[alpine] Detecting vulnerabilities...	os_version="3.21" repository="3.21" pkg_num=68
2025-06-12T09:31:38+05:30	INFO	Number of language-specific files	num=0

Report Summary

┌──────────────────────────────────────────────┬────────┬─────────────────┬─────────┐
│                    Target                    │  Type  │ Vulnerabilities │ Secrets │
├──────────────────────────────────────────────┼────────┼─────────────────┼─────────┤
│ kartikhiremath/weather_app:latest (alpine 3.21.3) │ alpine │        2        │    -    │
└──────────────────────────────────────────────┴────────┴─────────────────┴─────────┘
Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)


For OSS Maintainers: VEX Notice
--------------------------------
If you're an OSS maintainer and Trivy has detected vulnerabilities in your project that you believe are not actually exploitable, consider issuing a VEX
(Vulnerability Exploitability eXchange) statement.
VEX allows you to communicate the actual status of vulnerabilities in your project, improving security transparency and reducing false positives for your users.
Learn more and start using VEX: https://trivy.dev/v0.63/docs/supply-chain/vex/repo#publishing-vex-documents

To disable this notice, set the TRIVY_DISABLE_VEX_NOTICE environment variable.


kartikhiremath/weather_app:latest (alpine 3.21.3)
==================================================
Total: 2 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 2, CRITICAL: 0)

┌─────────┬────────────────┬──────────┬────────┬───────────────────┬───────────────┬──────────────────────────────────────────────────────────┐
│ Library │ Vulnerability  │ Severity │ Status │ Installed Version │ Fixed Version │                          Title                           │
├─────────┼────────────────┼──────────┼────────┼───────────────────┼───────────────┼──────────────────────────────────────────────────────────┤
│ libxml2 │ CVE-2025-32414 │ HIGH     │ fixed  │ 2.13.4-r5         │ 2.13.4-r6     │ libxml2: Out-of-Bounds Read in libxml2                   │
│         │                │          │        │                   │               │ https://avd.aquasec.com/nvd/cve-2025-32414               │
│         ├────────────────┤          │        │                   │               ├──────────────────────────────────────────────────────────┤
│         │ CVE-2025-32415 │          │        │                   │               │ libxml2: Out-of-bounds Read in xmlSchemaIDCFillNodeTables│
│         │                │          │        │                   │               │ https://avd.aquasec.com/nvd/cve-2025-32415               │
└─────────┴────────────────┴──────────┴────────┴───────────────────┴───────────────┴──────────────────────────────────────────────────────────┘
```

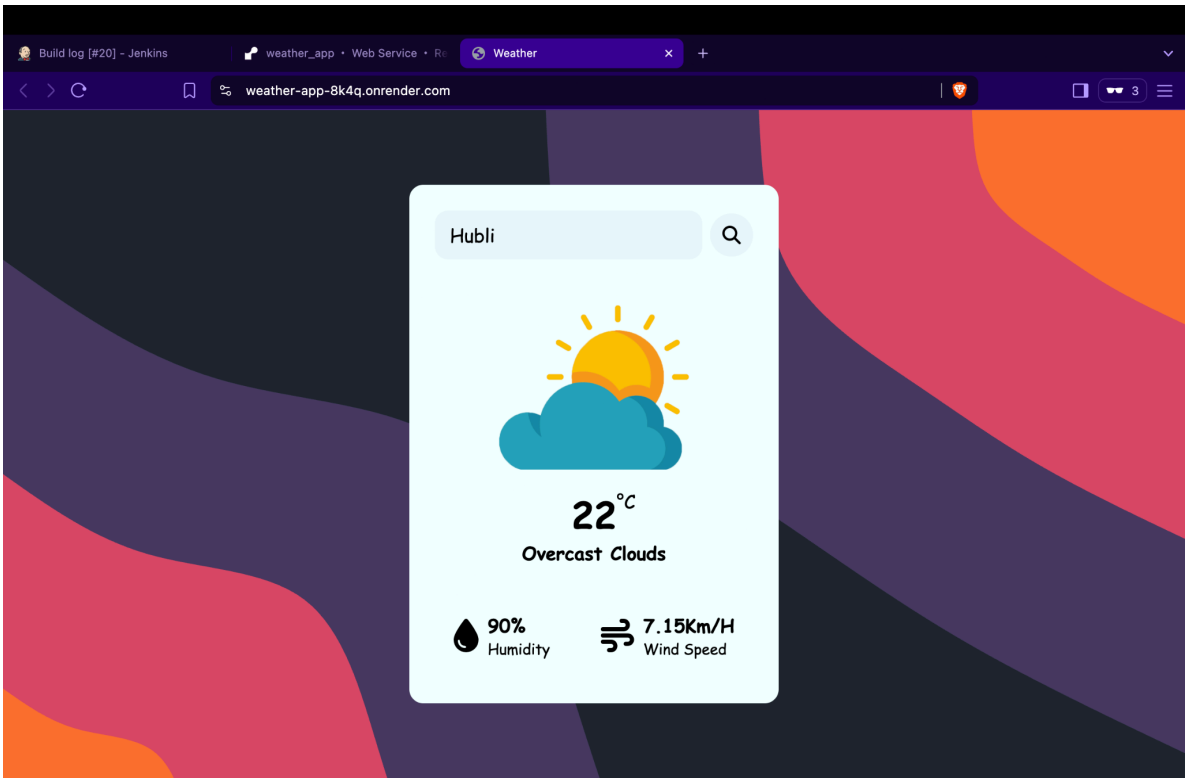
6. Push Docker Image

If all checks pass, Jenkins logs into Docker Hub using stored credentials and pushes the image under the repository kartikhiremath/weather_app:latest.

7. Deploy to Cloud

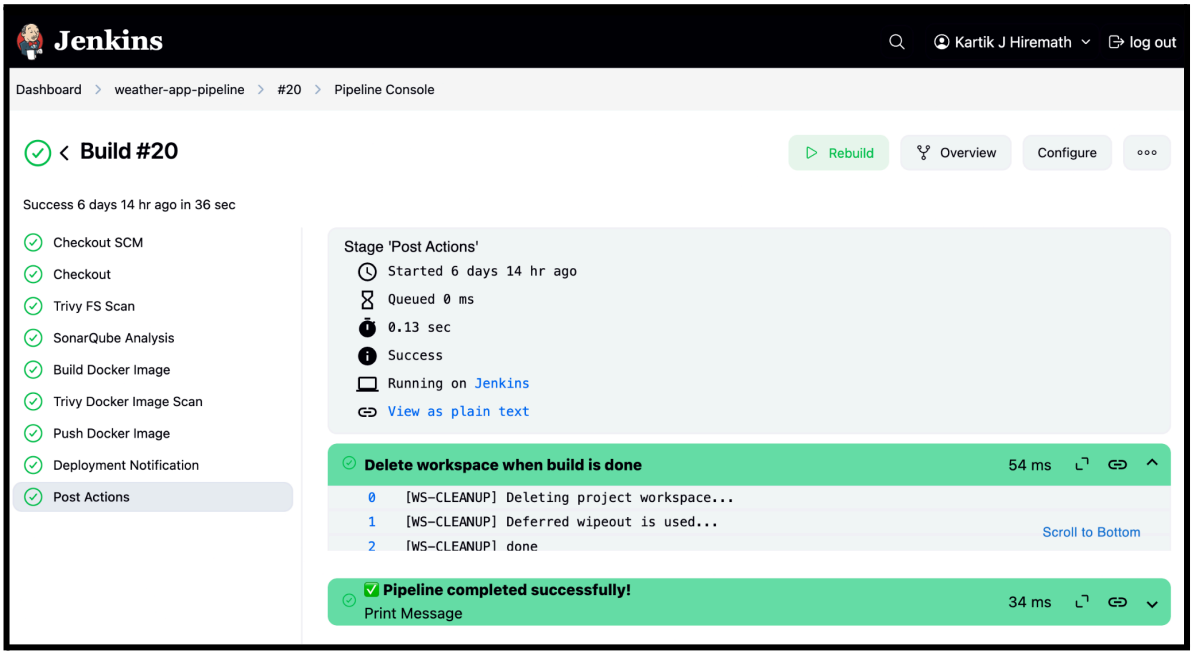
Render's GitHub integration automatically builds and deploys the container whenever the main branch is updated. Manual deployment stages are skipped to avoid conflicts with the local Jenkins host.





8. Cleanup

After deployment, the workspace is cleaned using the cleanWs() step to free disk space and prevent conflicts with future builds.



Summary

This CI/CD pipeline provides a fully automated workflow for a frontend weather application. Every change is securely scanned with Trivy, quality checked with SonarQube, packaged in Docker, and deployed to Render without manual intervention. The use of Docker ensures consistent runtime environments, while Trivy and SonarQube improve security and code quality. Deploying on Render simplifies hosting and scaling. The result is a reliable and repeatable process that accelerates feature delivery and reduces operational overhead.

Achievements

- Automated building, scanning, and deployment of a weather web application.
- Integrated static code analysis and vulnerability scanning into the pipeline.
- Packaged the application in Docker and published images to Docker Hub.
- Deployed the container to Render with zero-downtime updates.
- Maintained a clean workspace and secure credential management throughout the pipeline.

Appendix: Configuration Files

Jenkinsfile

```
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'kartikhiremath/weather_app:latest'
        SONARQUBE_ENV = 'MySonarQube' // must match your
Jenkins SonarQube config
    }

    stages {

        stage('Checkout') {
            steps {
                git branch: 'main',url:
'https://github.com/Kartik-Hiremath/weather_app.git',
credentialsId: 'dockerhub-creds'
            }
        }
```

```
        stage('Trivy FS Scan') {
            steps {
                sh 'trivy fs --exit-code 0 --severity
LOW,MEDIUM,HIGH .'
            }
        }

        stage('SonarQube Analysis') {
            environment {
                SONAR_TOKEN = credentials('sonarqube-token')
// must be a Jenkins credential
            }
            steps {
                withSonarQubeEnv("${SONARQUBE_ENV}") {
                    sh '''
                        sonar-scanner \
                        -Dsonar.projectKey=weather_app \
                        -Dsonar.sources=. \
                        -Dsonar.host.url=$SONAR_HOST_URL \
                        -Dsonar.login=$SONAR_TOKEN
                    '''
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                sh 'docker build -t $DOCKER_IMAGE .'
            }
        }

        stage('Trivy Docker Image Scan') {
            steps {
                sh 'trivy image $DOCKER_IMAGE'
            }
        }

        stage('Push Docker Image') {
            steps {

withCredentials([usernamePassword(credentialsId:
'dockerhub-creds', usernameVariable: 'DOCKER_USER',
```

```
                passwordVariable: 'DOCKER_PASS')]) {
                                sh '''
                                    echo $DOCKER_PASS | docker login -u
$DOCKER_USER --password-stdin
                                    docker push $DOCKER_IMAGE
                                '''
                            }
                        }
                    }

                    stage('Deployment Notification') {
                        steps {
                            echo '✅ App successfully deployed on Render
at: https://weather-app-8k4q.onrender.com'
                        }
                    }

                post {
                    always {
                        cleanWs()
                    }
                    failure {
                        echo '❌ Pipeline failed!'
                    }
                    success {
                        echo '✅ Pipeline completed successfully!'
                    }
                }
            }
```

Dockerfile

```
# Use NGINX to serve static files
FROM nginx:alpine

# Remove default nginx website
RUN rm -rf /usr/share/nginx/html/*

# Copy your frontend files
COPY . /usr/share/nginx/html/

# Expose port
```

```
EXPOSE 80

# Start NGINX
CMD ["nginx", "-g", "daemon off;"]
```

sonar-project.properties

```
sonar.projectKey=weather_app
sonar.projectName=Weather App
sonar.projectVersion=1.0
sonar.sources=.
sonar.exclusions=**/*.png,**/*.svg,**/*.jpg,**/*.jpeg

sonar.sourceEncoding=UTF-8
```

Links

- GitHub repository: https://github.com/Kartik-Hiremath/weather_app
- Live deployment: https://weather-app-8k4q.onrender.com