Project 1: Jenkins CICD pipeline

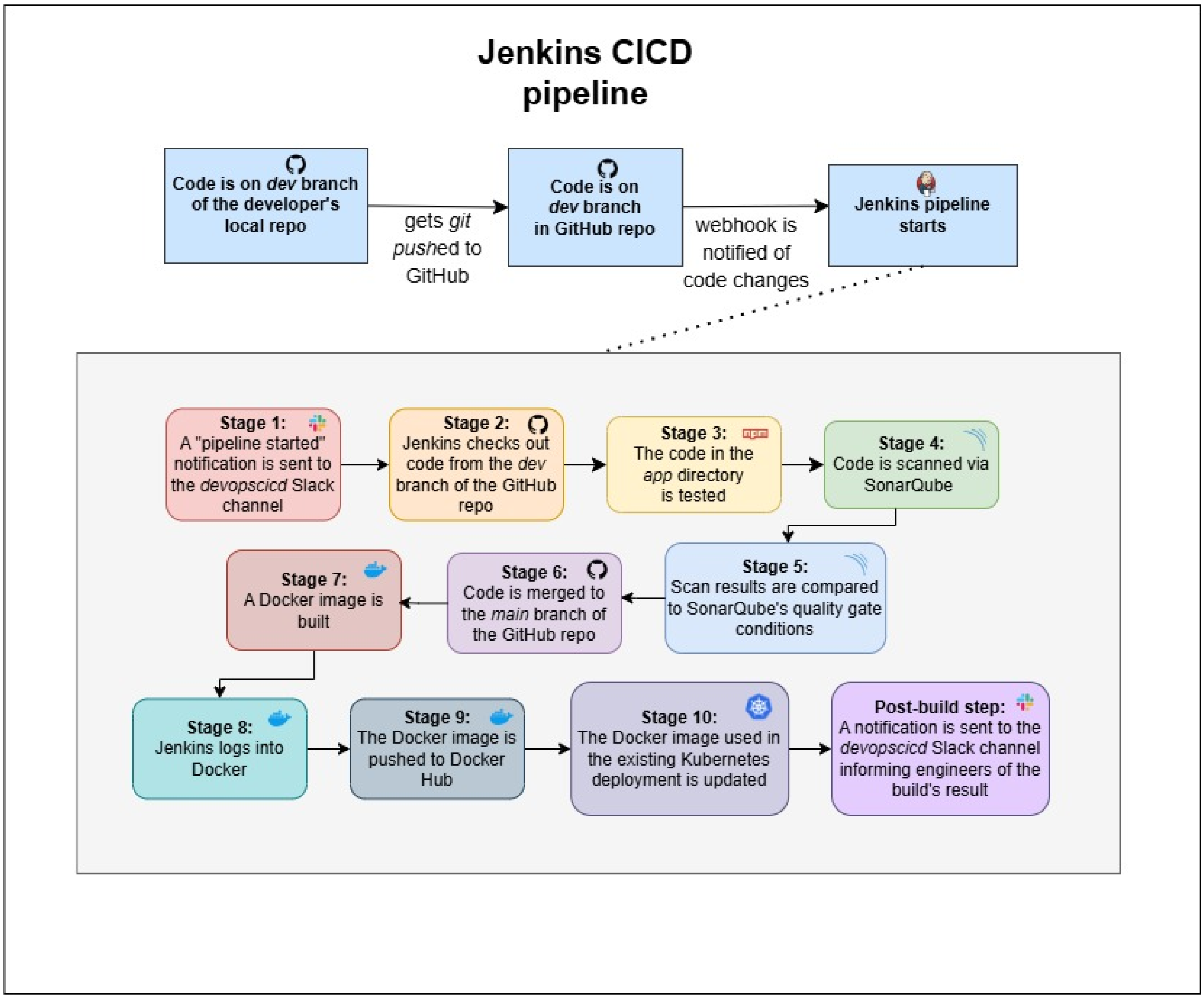
Goal of the project

The goal of this project is to implement a CI/CD pipeline for efficient software delivery. The pipeline will run on Jenkins and use SonarQube, Docker, and Kubernetes for the automated integration, code-scanning, deployment, and management of a containerised application already running via Minikube on an EC2. When the pipeline is started, a notification will be sent to a Slack channel to inform the DevOps team, and upon its completion, another notification will be sent to update them on the build's result.

Prerequisites

- A GitHub repo containing the app code with a *dev* and a *main* branch
- A Jenkins server
- A target VM with the app already running via Minikube, which references an image stored on Docker Hub
- A SonarQube server with a project for the app and a webhook set up to communicate with Jenkins
- A Slack account & channel where notifications will be sent, with a token generated for Jenkins via the Slack's Jenkins integration
- **Jenkins plugins**:
  - SSH Agent
  - SonarQube
  - NodeJS (with version 20 installed under *Manage Jenkins>Tools*)
  - Slack Notifications (with workspace and default channel set under *Manage Jenkins>Tools*)
- **Credentials loaded into Jenkins**:
  - Docker Hub username & password
  - SonarQube token
  - GitHub SSH key
  - SSH key for the target VMs
  - Slack

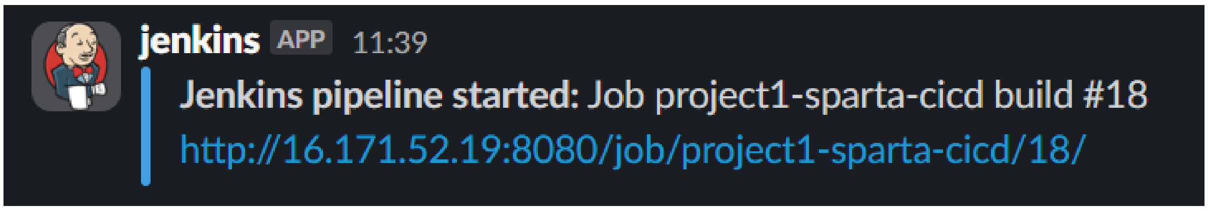
Diagram of pipeline


Jenkins CICD pipeline
Code is on *dev* branch of the developer's local repo
gets *git pushed* to GitHub
Code is on *dev* branch in GitHub repo
webhook is notified of code changes
Jenkins pipeline starts
Stage 1: A "pipeline started" notification is sent to the *devopscicd* Slack channel
Stage 2: Jenkins checks out code from the *dev* branch of the GitHub repo
Stage 3: The code in the *app* directory is tested
Stage 4: Code is scanned via SonarQube
Stage 5: Scan results are compared to SonarQube's quality gate conditions
Stage 6: Code is merged to the *main* branch of the GitHub repo
Stage 7: A Docker image is built
Stage 8: Jenkins logs into Docker
Stage 9: The Docker image is pushed to Docker Hub
Stage 10: The Docker image used in the existing Kubernetes deployment is updated
Post-build step: A notification is sent to the *devopscicd* Slack channel informing engineers of the build's result

Pipeline steps overview

Full pipeline available to view here

1. Outside of the pipeline, a colour map function is first defined that will be used in the post-build Slack notification step
2. The pipeline starts, with credentials for Docker Hub, GitHub, and the SSH key for the target VM stored as variables that can be referenced throughout the entire pipeline in the `environment` block
3. NodeJS version 20 is specified in the `tools` block
4. **Stages**:
    1. A notification is sent to the *#devopscicd* Slack channel informing the DevOps team that the pipeline has been started

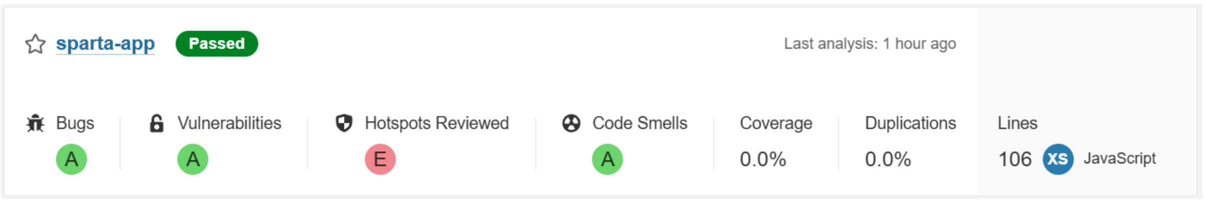


jenkins APP 11:39

Jenkins pipeline started: Job project1-sparta-cicd build #18
http://16.171.52.19:8080/job/project1-sparta-cicd/18/

    2. The code is checked out from the *dev* branch of the GitHub repo using the credentials stored as an environment variable
    3. The *app* folder is set as the working directory, and tests are run on the code

   4. The code is scanned by the SonarQube server, where it is stored as a project called *sparta-app*

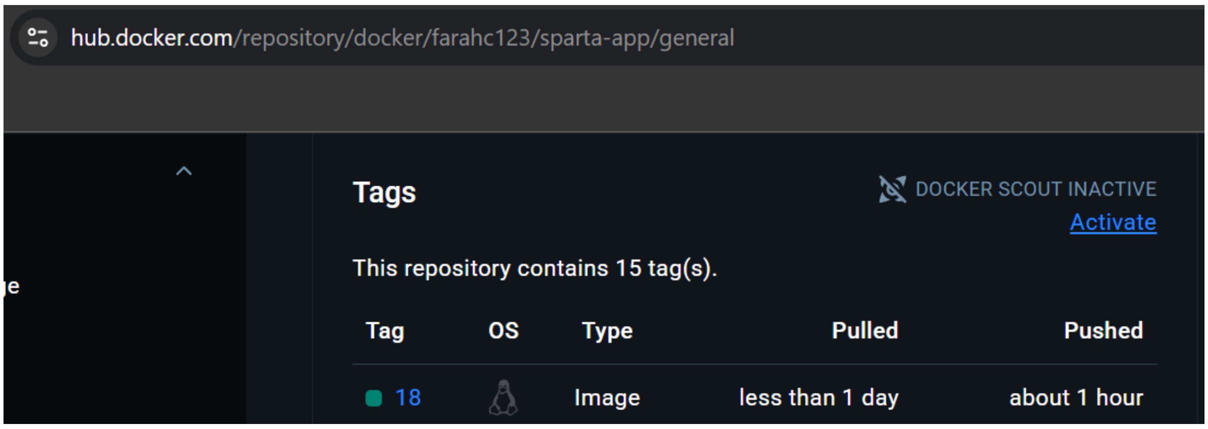


sparta-app Passed

Last analysis: 1 hour ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
A	A	E	A	0.0%	0.0%	106 XS JavaScript

   5. The results of the scan are compared to the quality gate conditions (in this case, SonarQube's default quality gate is used)
   6. Provided the above stages run successfully, the tested code is merged to the *main* branch of the GitHub repo
   7. The Docker image is built, with the build number of the Jenkins pipeline being passed as the image tag
   8. Jenkins logs into Docker using the provided credentials
   9. The Docker image is pushed to Docker Hub



hub.docker.com/repository/docker/farahc123/sparta-app/general

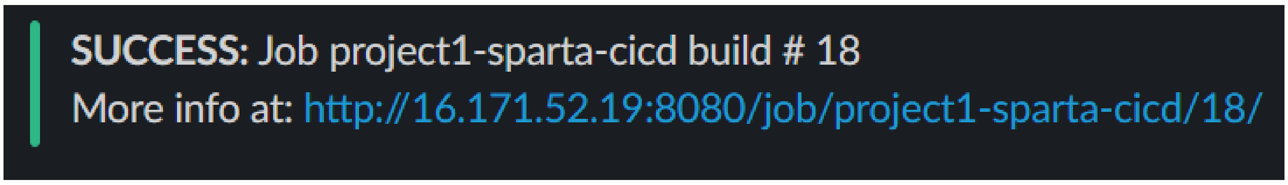
Tags

DOCKER SCOUT INACTIVE Activate

This repository contains 15 tag(s).

Tag	OS	Type	Pulled	Pushed
18		Image	less than 1 day	about 1 hour

   10. The Docker image used in the existing Kubernetes deployment (running on Minikube on the target VM) is updated
5. Post-build, a notification is sent to the *#devopscicd* Slack channel informing the DevOps team of the build's result



SUCCESS: Job project1-sparta-cicd build # 18
More info at: http://16.171.52.19:8080/job/project1-sparta-cicd/18/

Demonstration

Watch the pipeline in action here