Github Link:
https://github.com/aman-deep739/Hop-Petclinic

Tools used:

Continuous Integration :

- Git - Source Code Management
- AWS - For Jenkins, Nexus, Sonarqube and Dockerhost Servers
- Jenkins - Continuous Integration Tool
- Maven - Build Tool
- Nexus - Artifact Repository
- Sonarqube - Code Analysis

Continuous Deployment :

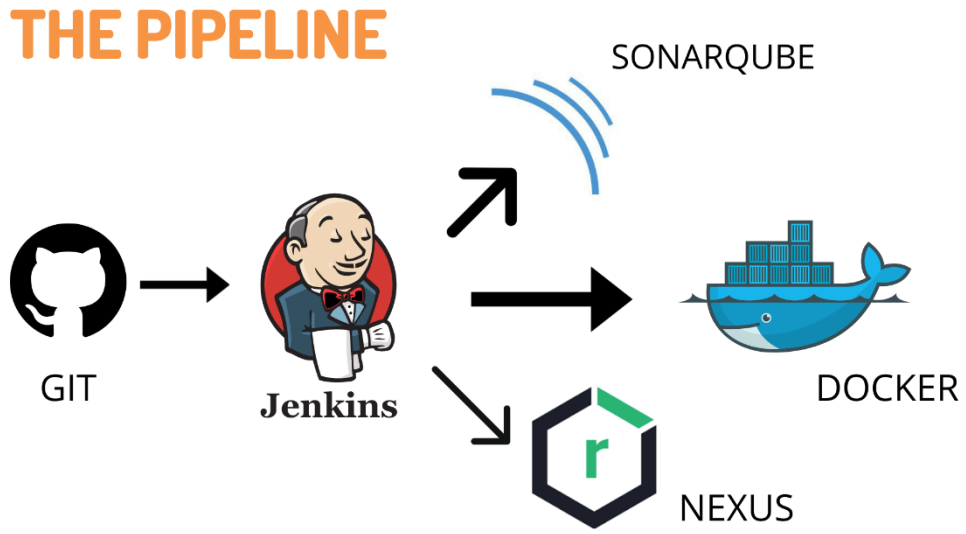
- Docker – Containerization

Why these tools:

- **AWS** – To host Jenkins, Sonarqube, Nexus and Docker servers on EC2 instances. Any public cloud service could have been used. I used AWS as its more widely used.
- **Maven** - Build Tool – To build and test the source code. The project on GitHub was a Maven based project.
- **Sonarqube** – I used Sonarqube for static code Analysis. Published the results of code analysis to Sonarqube server.
- **Nexus** - Artifact Repository – Stored the artifacts(jar files) on Nexus Artifact Repository. JFrog can also be used.
- **Jenkins** – Used as a continuous integration and deployment Automation tool. Integrated Sonarqube, Nexus and Dockerhost with Jenkins. Other tools like Github action, Travis CI etc can also be used.
- **Docker** - Containerization – Containerized the application using Docker. The end application was hosted on Dockerhost setup on AWS EC2. Docker is the industry standard when it comes to containerization.

Different tools can be used depending on the requirements.

The Pipeline: Flow



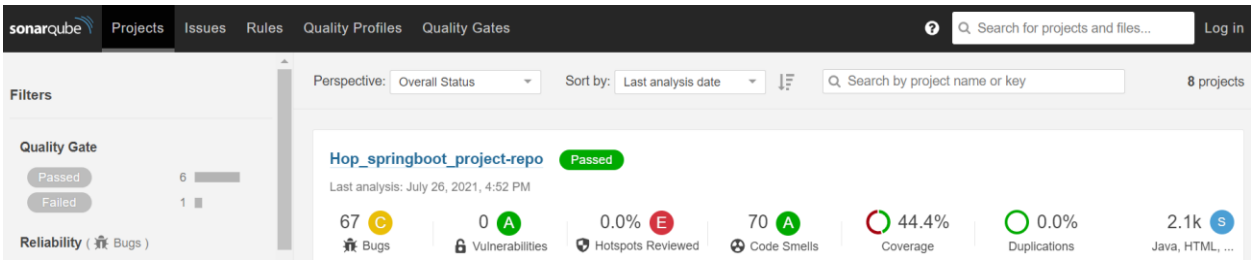
Setup and Execution:

CONTINUOUS INTEGRATION:

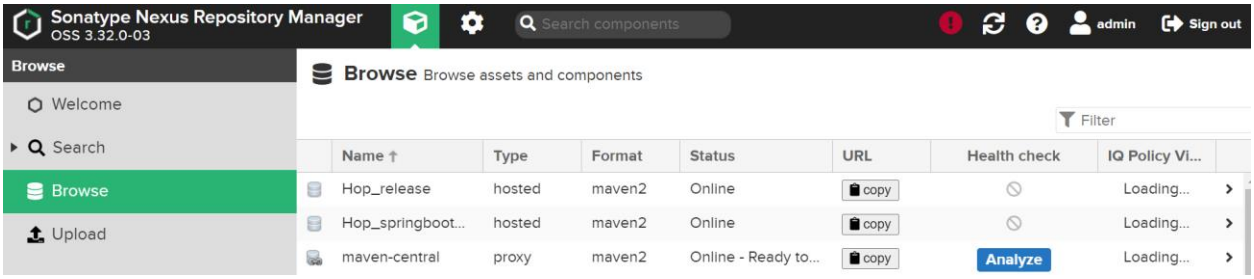
1. Setup Jenkins, Nexus, Sonarqube on AWS EC2 using shell scripts.
2. Did further configuration of Jenkins, Nexus, and Sonarqube servers. Installed necessary plugins in Jenkins. Created users in Nexus and Sonarqube. Created repositories in Nexus.
3. Created Jobs in Jenkins.
    i. Build Job - This Job will pull the source code from GIT and build it using Maven. It will generate a jar file.
    ii. Test Job - To do unit testing.
    iii. Integration Test Job - To do the integration testings.
    iv. Sonarqube-Code-Analysis - This job will perform the Code Analysis and publish Result on sonarqube server.
    v. Deploy artifact to Nexus - This job will publish the artifact on Nexus Server.

Each job will start only if the previous job was successful.

Sonarqube Server:

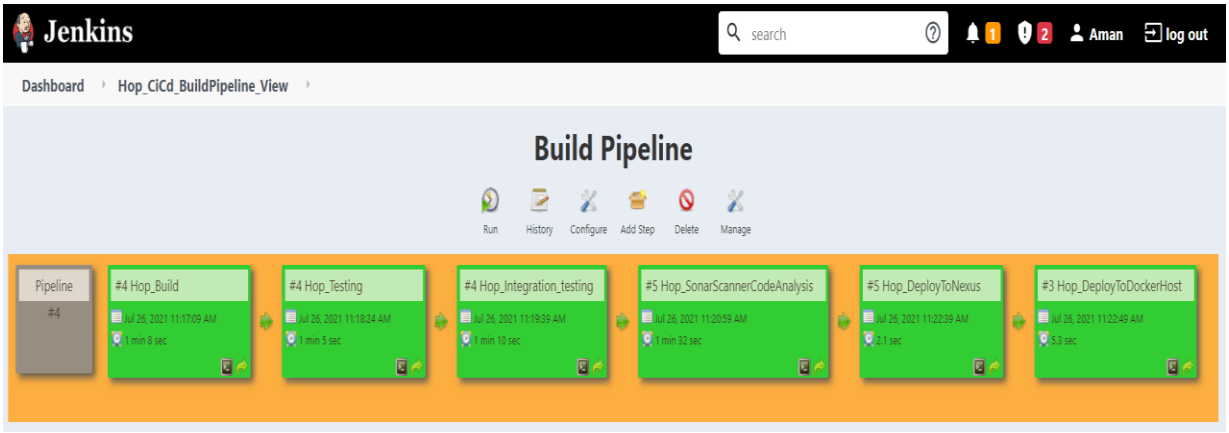


Nexus Server:



CONTINUOUS DELIVERY:

4. Setup a Docker host on EC2 server.
5. Wrote Dockerfile and docker-compose files in Docker host.
6. Created Job in Jenkins to build image and run the docker-compose file. This will bring up the application stack.



NOTE: In its default configuration, Petclinic uses an in-memory database (H2) which gets populated at startup with data. To have persistent data, I have used MySql database.

Challenges Faced:

- Previously I had made projects with much smaller source codes and had set up Jenkins on a t2.micro EC2 instance. This time around the EC2 instance was not able to handle the build process and it was crashing. I changed the instance type to t2.small and it then was able to work smoothly without any lags or crashes

- Initially, when I linked The frontend App and the backend MySql database using docker-compose, the DB was not able to link with the Application. Figured out a small change had to be made in the source code.

Handling Downtime:

- As of now the app is deployed on a single Ec2 Node as single containers (1 for App, 1 for DB).
- For production, we will be needing to deploy the application on a Kubernetes Cluster.
- We can setup multiple nodes as part of Kubernetes cluster and deploy Pods running the application. Such that, if a particular node/pod is down, the traffic is directed to a healthy node/pod.

Further Improvements:

- I had created the infrastructure on AWS manually as it was a small setup. For production level, Terraform has to be used to have the Infrastructure as Code for provisioning and better management of Resources.
- I had used freestlyle Jobs in jenkins. We can also use PaC and maintain it in Github.
- I have used Jenkins for building the docker image and running the docker-compose file. For production, Ansible has to be used for configuration management and deployment.
- Docker images can be pushed to an image repository like Dockerhub or ECR.
- Application should be deployed on Kubernetes cluster for High Availability.

~~~
~~~