CI/CD Pipeline by using Jenkins for Maven Build, Test, Integration Test, Sonarqube (Quality Gates), Nexus (Artifacts Repo Manager), Software Testing and Slack for Notifications

(Hands on Practice from Udemy) Date: Feb 28, 2022

Scenario

Consider there is a product development is happening, with Agile SDLC. In this developers building a products and making a regular code changes. All these code changes which gets merged with remote repository, to be built, tested and deploy to server for further. Once the code is packaged into an artifacts, it will be deployed to server. Software testing/ integration testing after deployment. Test report gets evaluated and approval for prod deploy issued.

Problems

- In an Agile SDLC, there will be frequent code changes
- Manual deployment is time consuming
- Involves task assignment/ ticketing/ approvals
- Dependency on Ops, & Build & Release Team

Solution

- Build, Test, Deploy & Test for every commit
- Automated Process
- Notify for every build
- Fix code if bugs, or error found instantly rather than waiting

Tools to be used

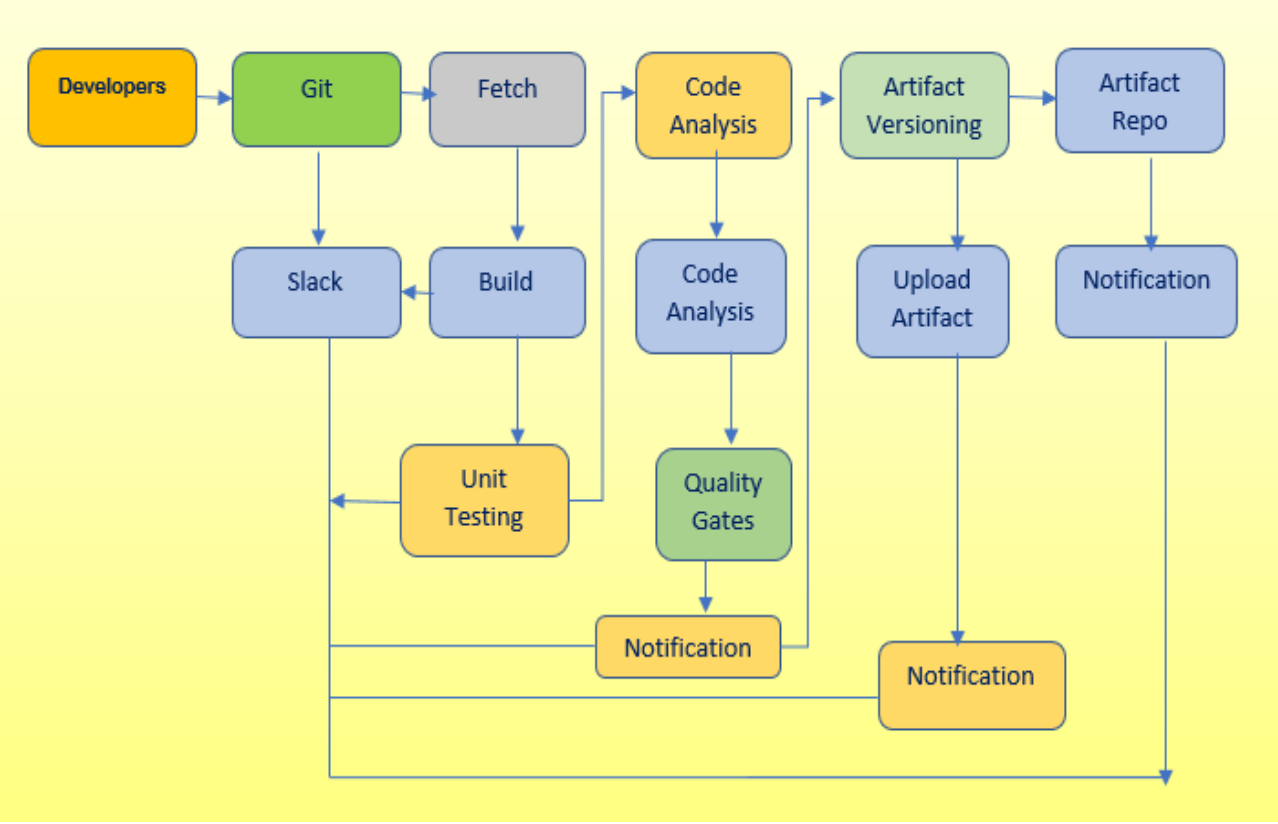
- Jenkins as Continuous Integration Server
- Maven as build tool
- Git as version control
- Check style, Slack
- Nexus as Artifact Repository
- SonarQube as Code Analysis
- Tomcat as Web application Server
- Selenium for Software testing
- Windows server and AWS EC2 for all servers

Steps to be followed:

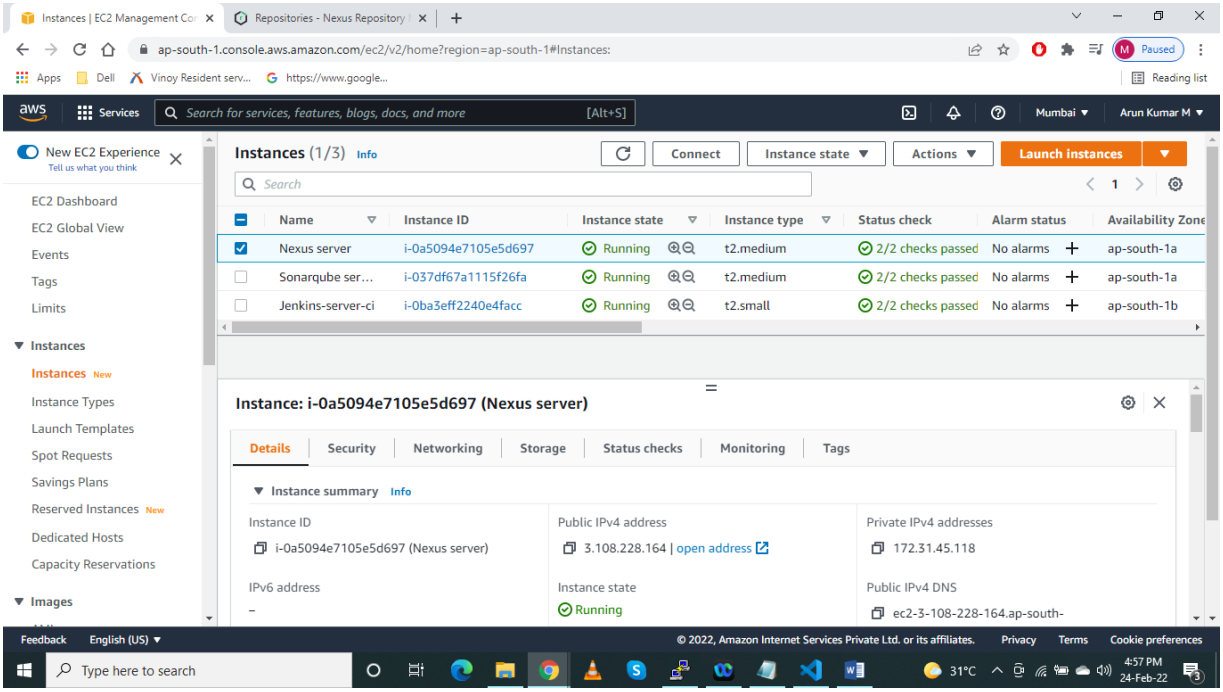
- Login to AWS account.
- Create login key
- Create Security group for Jenkins, Nexux, Sonar
- Create EC2 instances with user data for Jenkins, Nexus and Sonar Qube

- Post Jenkins installation, access from web page and complete initial login setup
- Create repository in Nexus post Nexus installation including Maven repo
- Sonarqube post installation, and create a token which is to mapped with Jenkins job
- Jenkins Steps: Build Job, Setup Slack Notification, Check style code analysis job, Set up Sonar integration, Sonar Code Analysis, Artifact upload job,
- Connect all jobs with build pipeline.
- Set automatic build trigger for each pipeline jobs.
- Test with git
- Create Security group for windows server, Tomcat and backend server
- Setup tomcat and backend server on EC2 with user data
- Add windows node as slave to Jenkins
- Create job to run software tests, from windows server
- Deploy artifact to production tomcat server
- Connect all the jobs with build pipeline
- Test it by committing the code to Git Hub

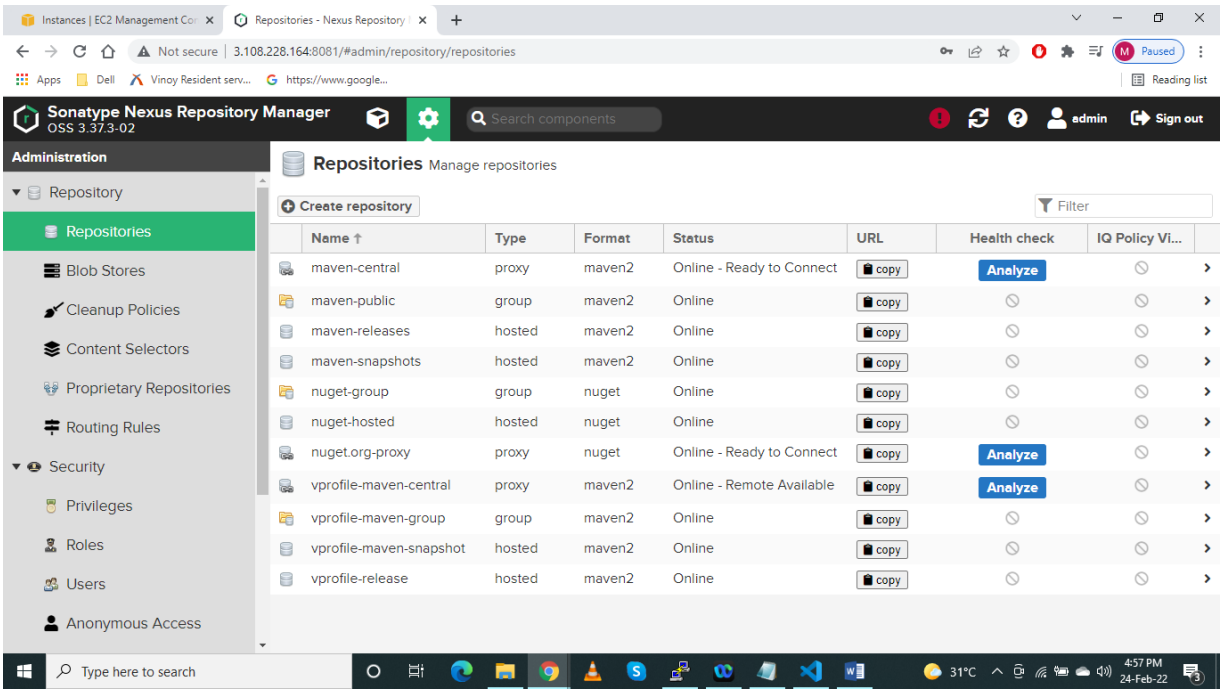
Workflow of Continuous Integration



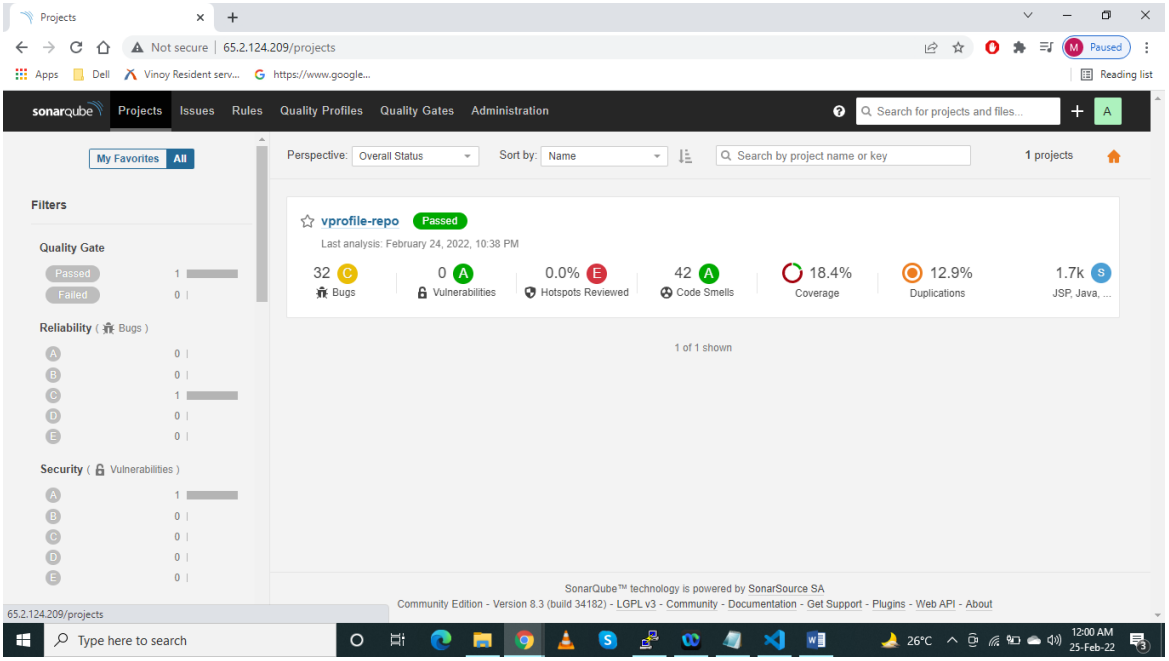
Screenshots of EC2 services of Jenkins, Nexus, and Sonarqube servers



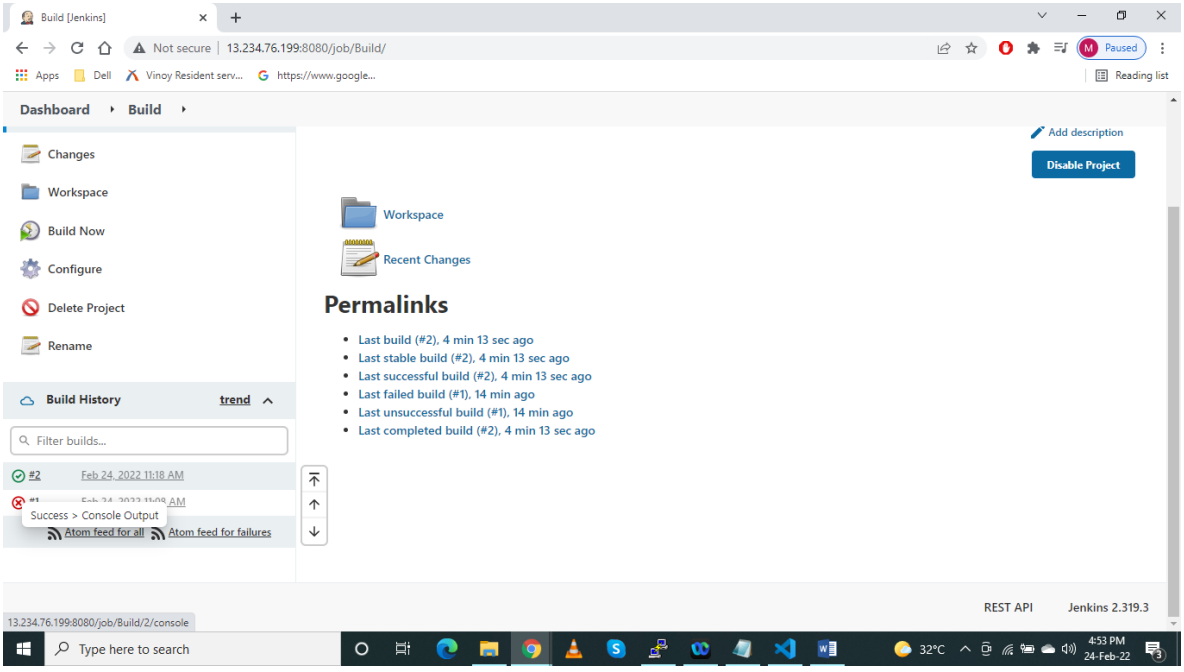
Screenshot of Nexus Repository Manager



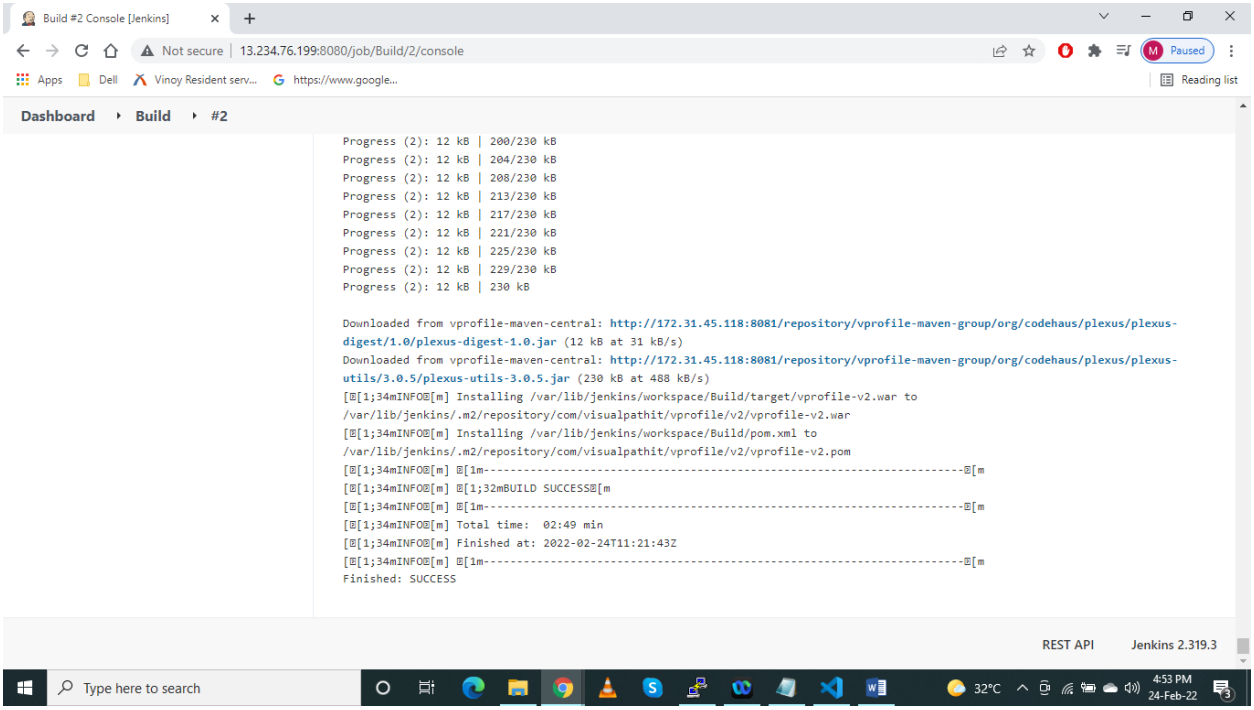
Screenshot of Sonarqube Quality gates



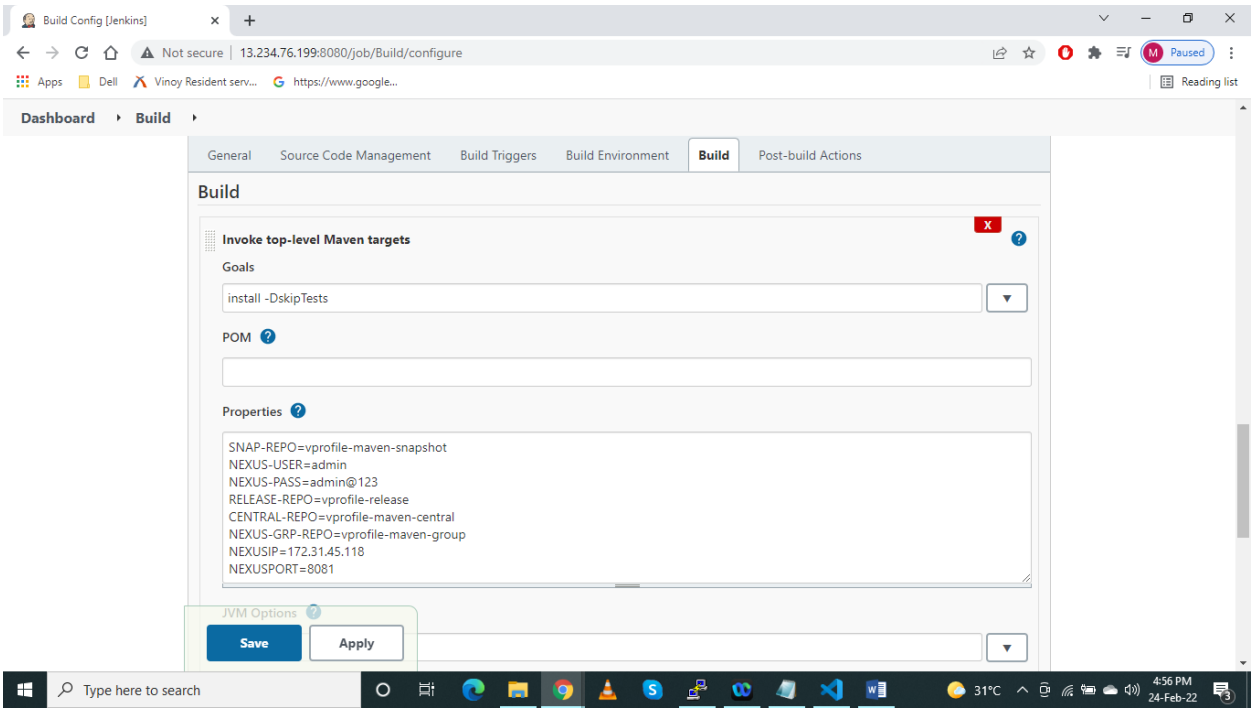
Post installation of Jenkins, ready to configure to build a pipeline



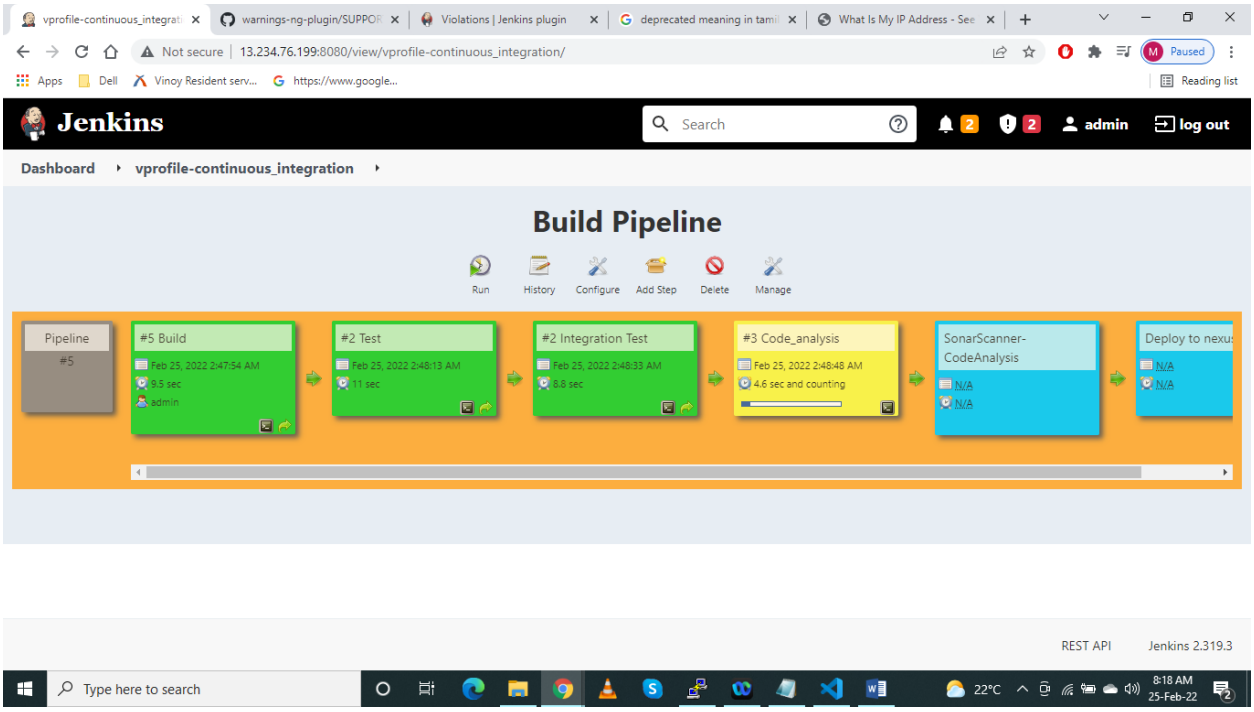
Console output of Maven build when build comes success

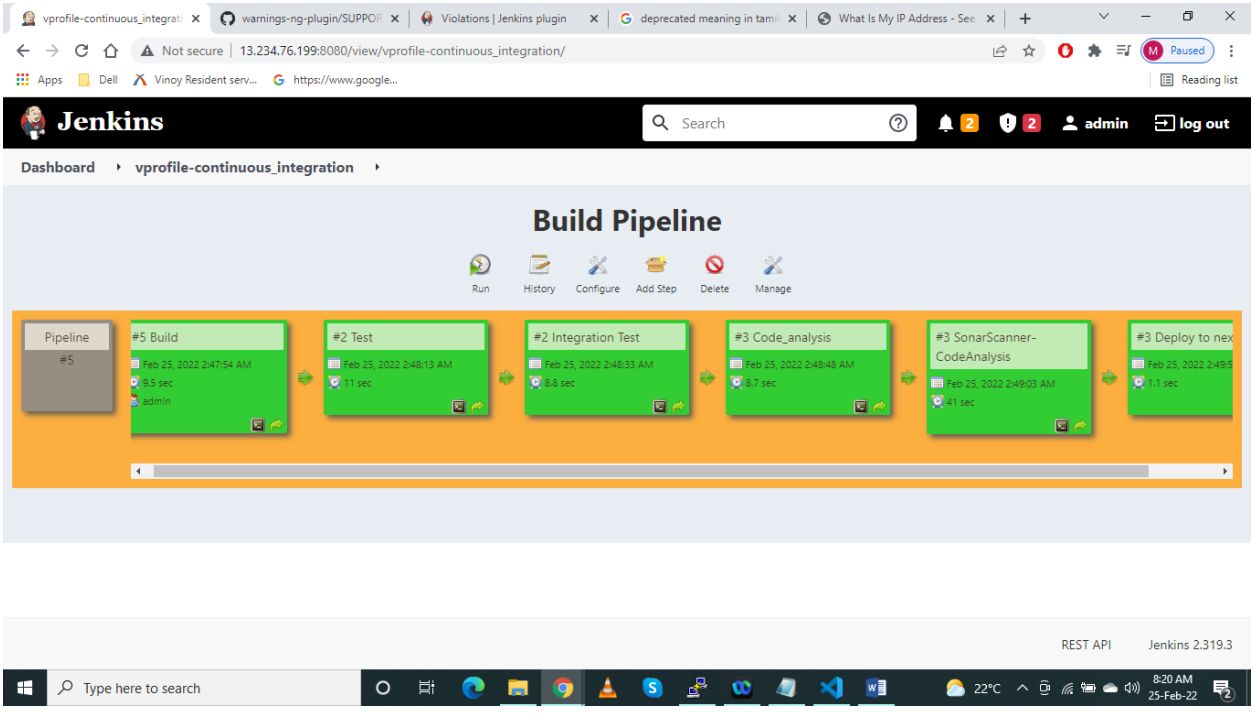


Giving application properties with variables in build job configuration.

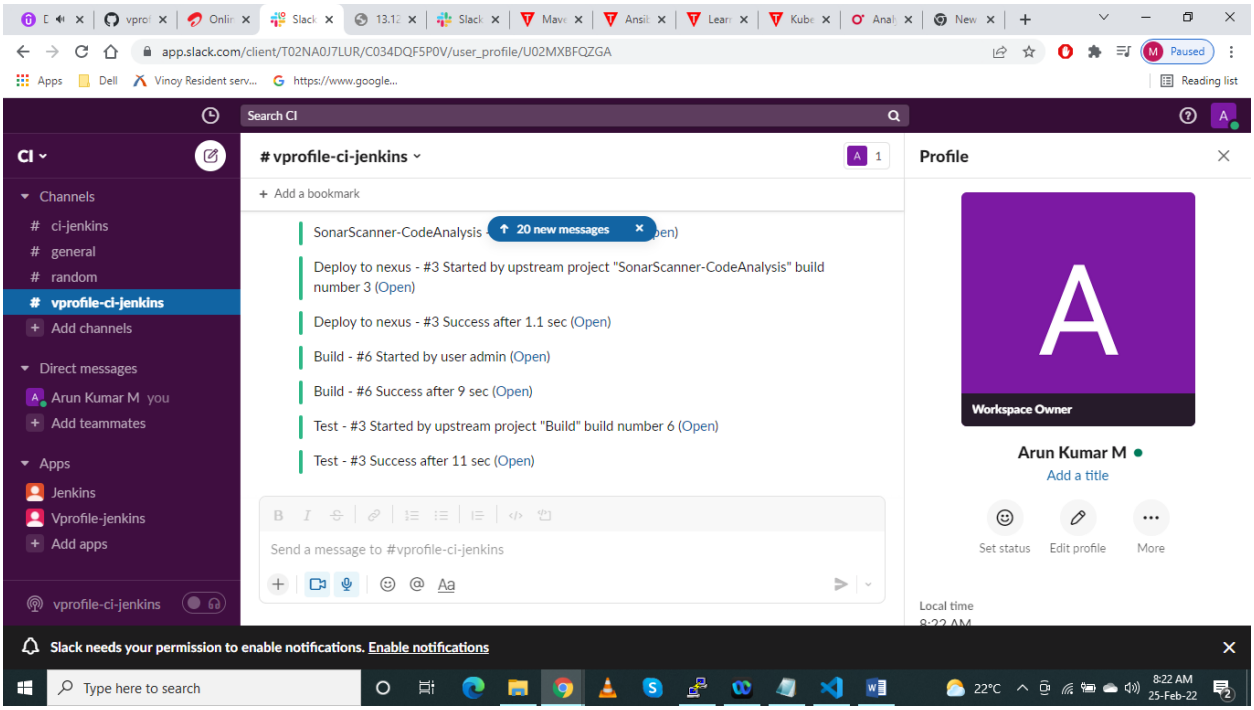


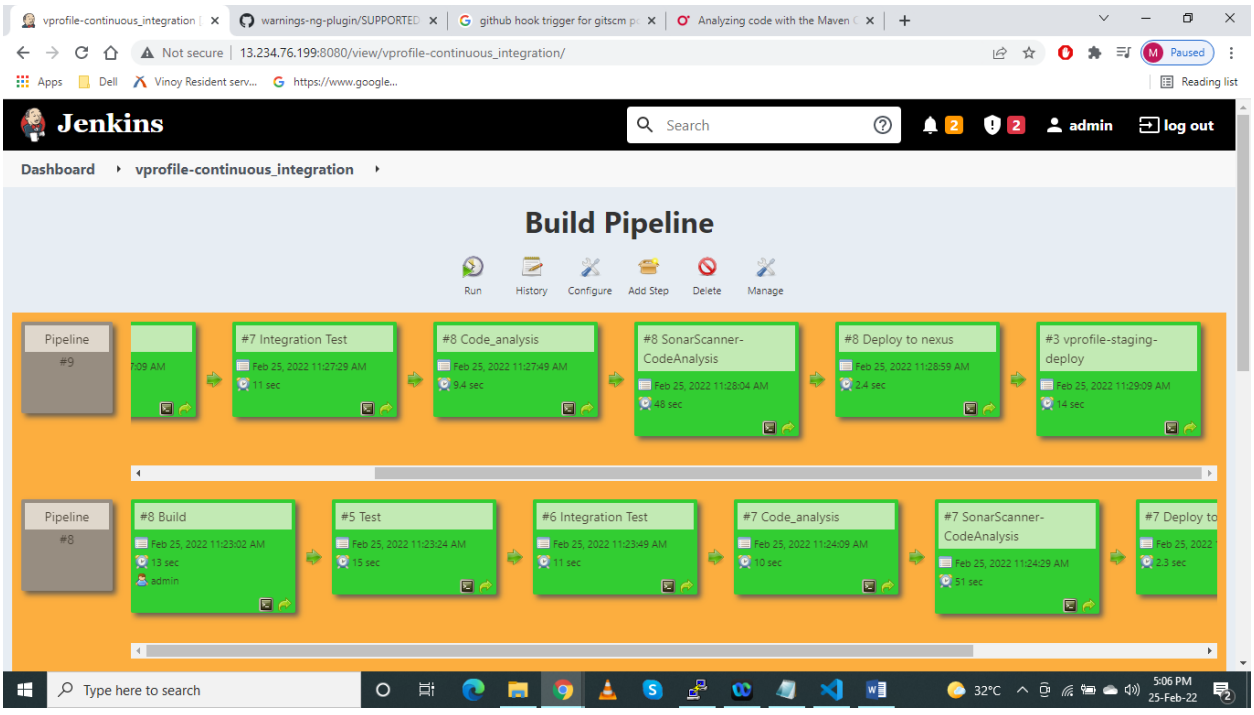
Screenshot of pipeline jobs created such as Build, Test, Integration Test, Code Analysis, Sonar Scanner Analysis, and Deploy to Nexus



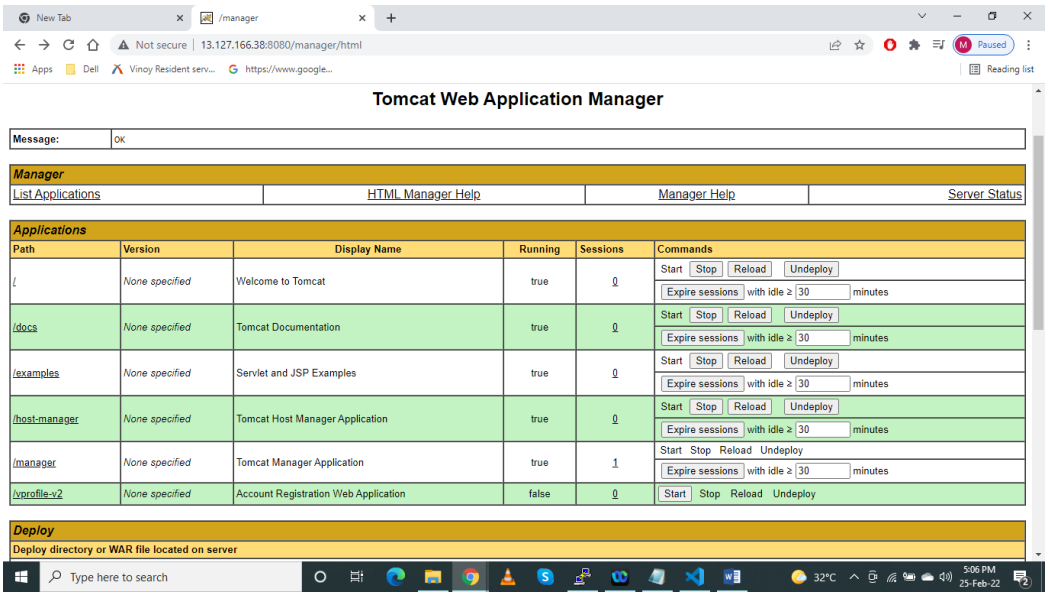


Screenshot of Slack notifications while committing each code changes, which triggers the automated build and rest of pipeline jobs

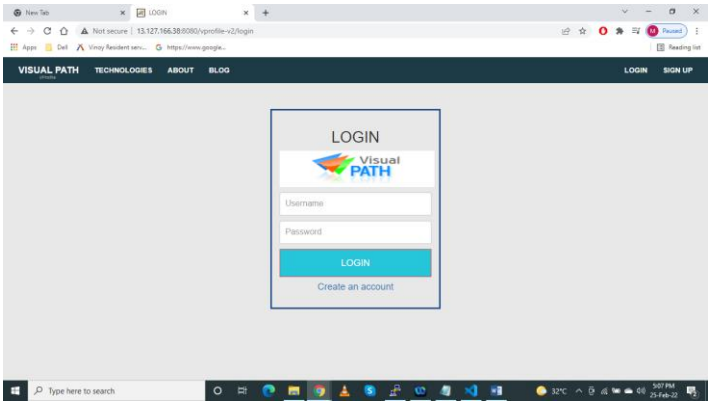


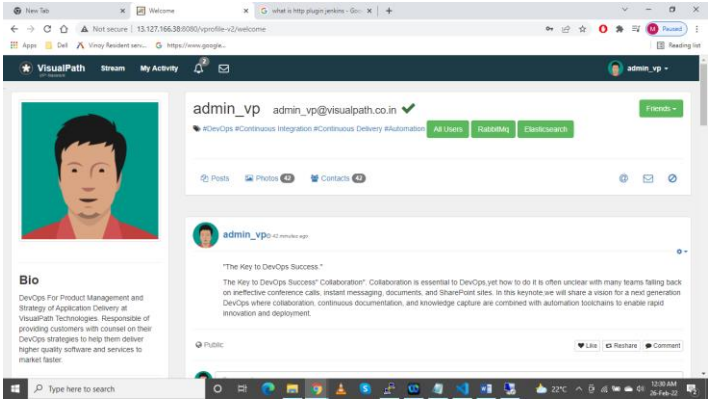


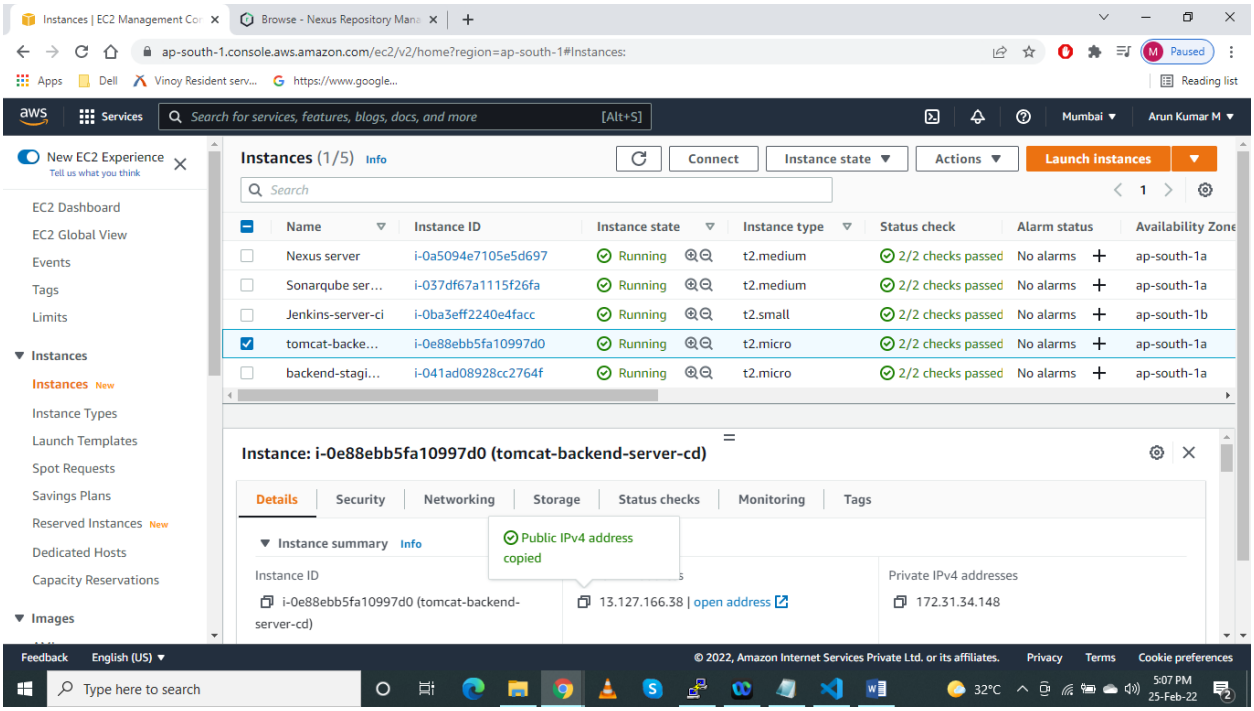
Placing WAR file on tomcat application, and validating



Screenshot of web application home page, which is running from the tomcat server installed fro continuous delivery.







Screenshot of software testing job performed in windows server, which will take screen shot of login page function test.