Day 27 Revisit

1. Docker
2. AWS Regions & AZ (Availability Zones)
3. Git Branching & Merging

Git branching – Creating a new branch to add some enhancement, new features to the project without affecting the existing code.

Git Merging – Process of combining the branched files with the master files. Some Merge conflict will occur when more than one user trying to update a same line with different code in a file.

AWS Regions – Physical Locations where the Data Centers are situated.

Each Region consist of multiple availability zones each of it are interconnected.

Docker – It’s a DevOps Tool. – Helps to create containers of the application.

Container – Project source code along with the required libraries and all dependencies to run it.

Docker Architecture (Docker Engine/Docker Server/Docker Daemon ---- Docker Client/Docker Desktop/ Docker CLI (Stand-alone tools) --- Docker Hub / Docker Registry (cloud environment)

Docker – It’s a container Management tool ( Create Images, Create Containers, Run the images , Build the images, DockerFile, DockerSwarm, DockerCompose, Kubernetes)

K8S/ Kubernetes – Container Orchestration Tool (Enables Auto Scaling, Load Balancing etc.,)

Using a Single image, multiple containers can be created

Image -- It’s a bundle of source code, required libraries & all dependencies

Both Official & Verified images are hosted in hub.docker.com (docker cloud/ docker registry)

Docker images are available for various category (OS, Database, Security, Programming Lang, Application Framework & Tools)

Docker can also be using Docker CLI (Docker commands are available)

CLI – Command Line Interface. [docker -v, docker build, docker run, docker compose up]

Docker -- help (All the list of available commands)

A docker image can be created with the help of DockerFile.

DockerFile will have all the details about the dependencies and libraries.

Dockerfile will not have any extension.

DevOps Tool (Jenkins)

CI/CD Tools

CI – Continuous Integration

CD – Continuous Delivery / Deployment

SonarQube – Code Quality Analyzing Tool

SDLC – Software Development Life Cycle

Different phases of SDLC in Water Fall Model

1. Planning – BRD (Business Requirement Document)
2. Designing – DDD (Detailed Design Document) – UML, Flowchart, DB ERD,
3. Coding – Project Source code
4. Testing – Unit Test, Integration Test, Performance Test, UAT (User Acceptance Test)
5. Release – Released to all
6. Maintenance – Bug Fixing, enhancement, updates, patches

Agile Methodology

1. Sprint – duration 1 week to 1 month – (little bit of planning, little bit of designing, little bit of coding, little bit of testing, little bit of releasing)
2. Incrementally increasing the same for each sprint

Agile will not give more importance to documents,

The Documents required in AGILE methodology

1. Product Backlog ( All the project related requirements in the form of user stories)
2. Sprint Backlog (High priority items from product backlog to sprint backlog -)
3. Burn Down Chart / Effort Utilization Chart)

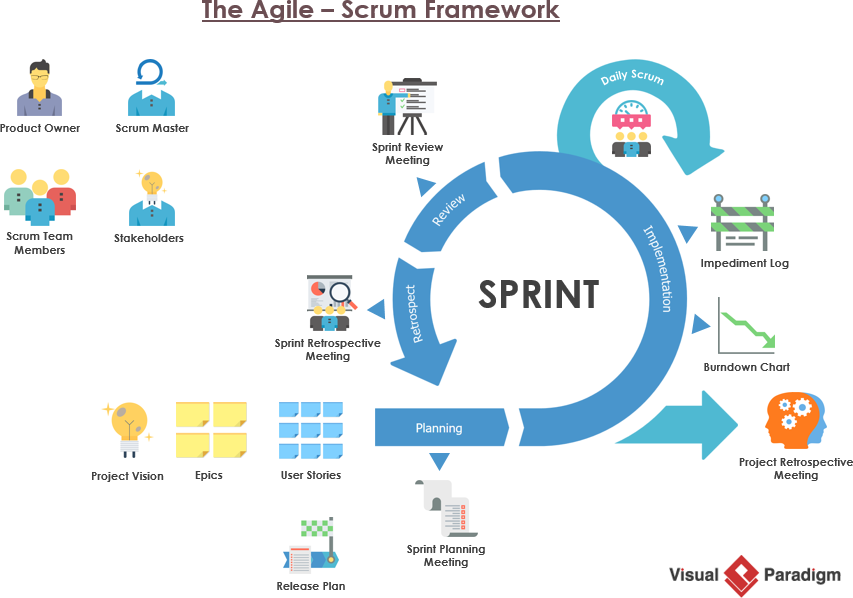
JIRA – Is a project Management Tool

In Agile

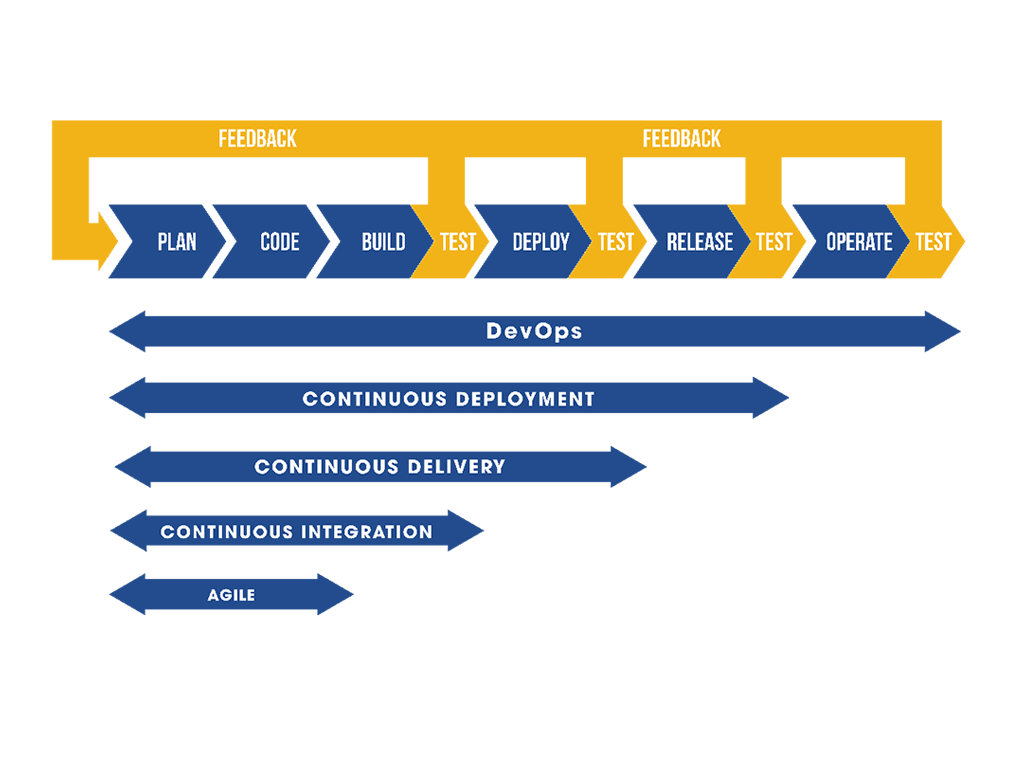
1. Product Owner (Who gives us the requirements)
2. Scrum Master (Who manages the entire scrum team)
3. Scrum Team

Events in Agile

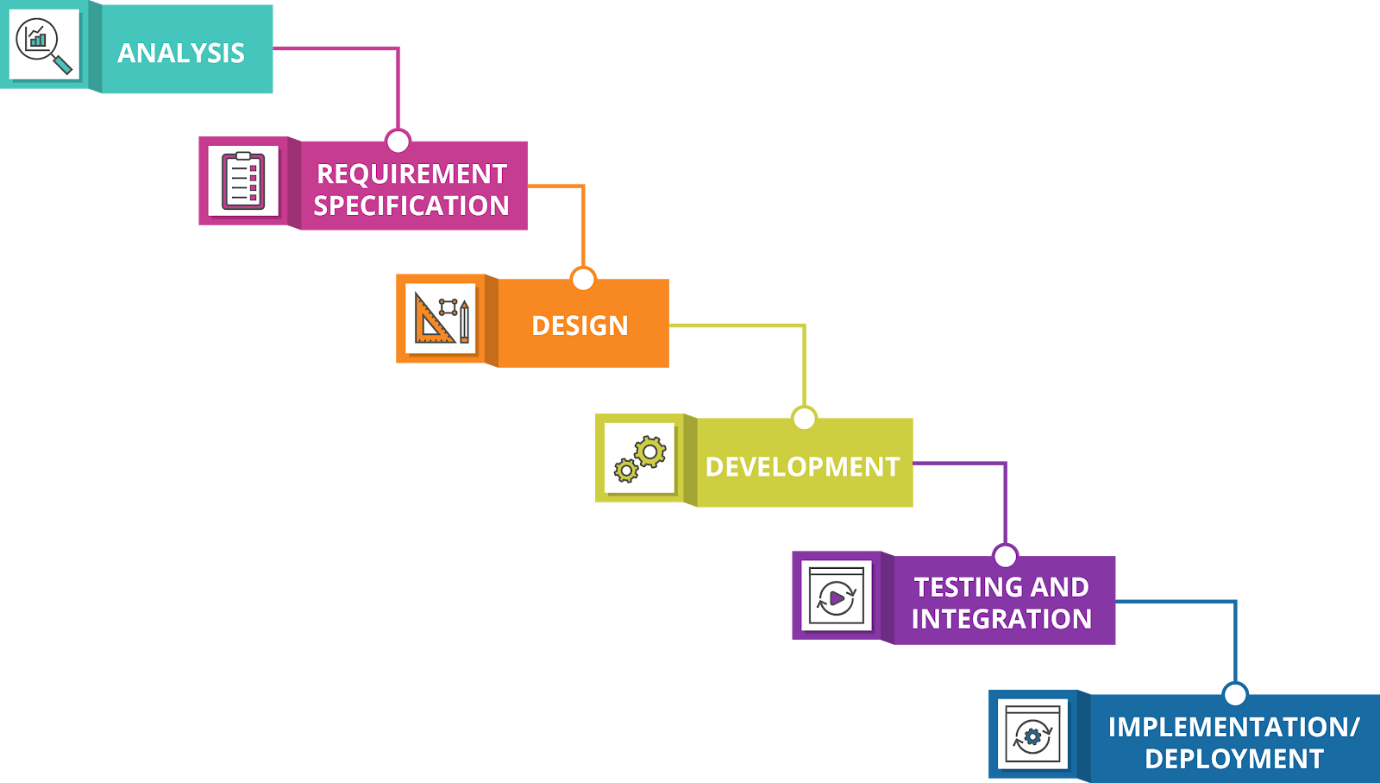
1. Daily Stand up (10 minutes, Every one needs to stand, It happens every day – What is done, what are you planning to do, any road blocks)
2. Sprint Planning (At the beginning of a sprint – one time activity for each sprint)
3. Sprint Retro (Retrospective) (At the end of each sprint – MVP – Minimum viable product)



CI – Continuous Integration



Water Fall Model (SDLC)



<https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>

Atlassian – bitbucket, jira

TDD – Test Driven Development (Write the Test case first before writing the application code – Initially all test will fail, Then write the code to make all the test pass. & enhance it) JUnit

BDD – Behavior Driven Development (Cucumber Tool, Gherkhin Language)

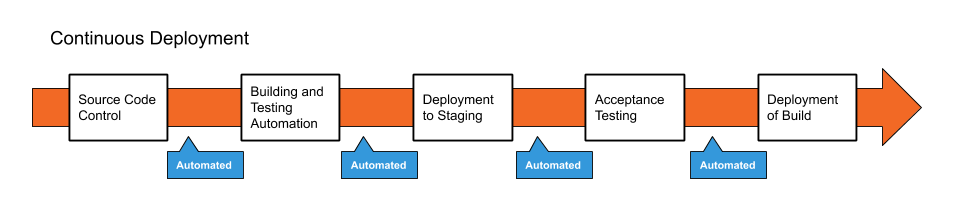
Continuous Integration (CI) - Continuous Integration is the process of regularly and consistently merging code into a central repository and reviewing new code to ensure that it integrates well within the previously established code base.

CI Tools

* Jenkins
* TeamCity
* Travis CI
* Bamboo
* Gitlab CI/CD
* CircleCI

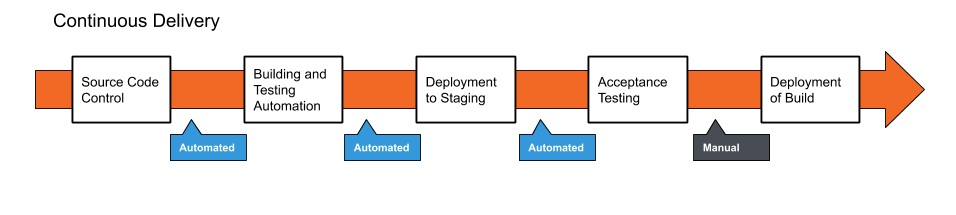
**Continuous Deployment** is a process of releasing software in which changes are tested for stability and correctness automatically. This results in immediate, autonomous deployment of code to production environments.

**Automation** – Automated Test, Automated Build, Automated Deployment, Automated Delivery



**Continuous Delivery**

Continuous Delivery is a paradigm in which the building, management and testing of produced software is automated such that deployments can be performed at the push of a button.



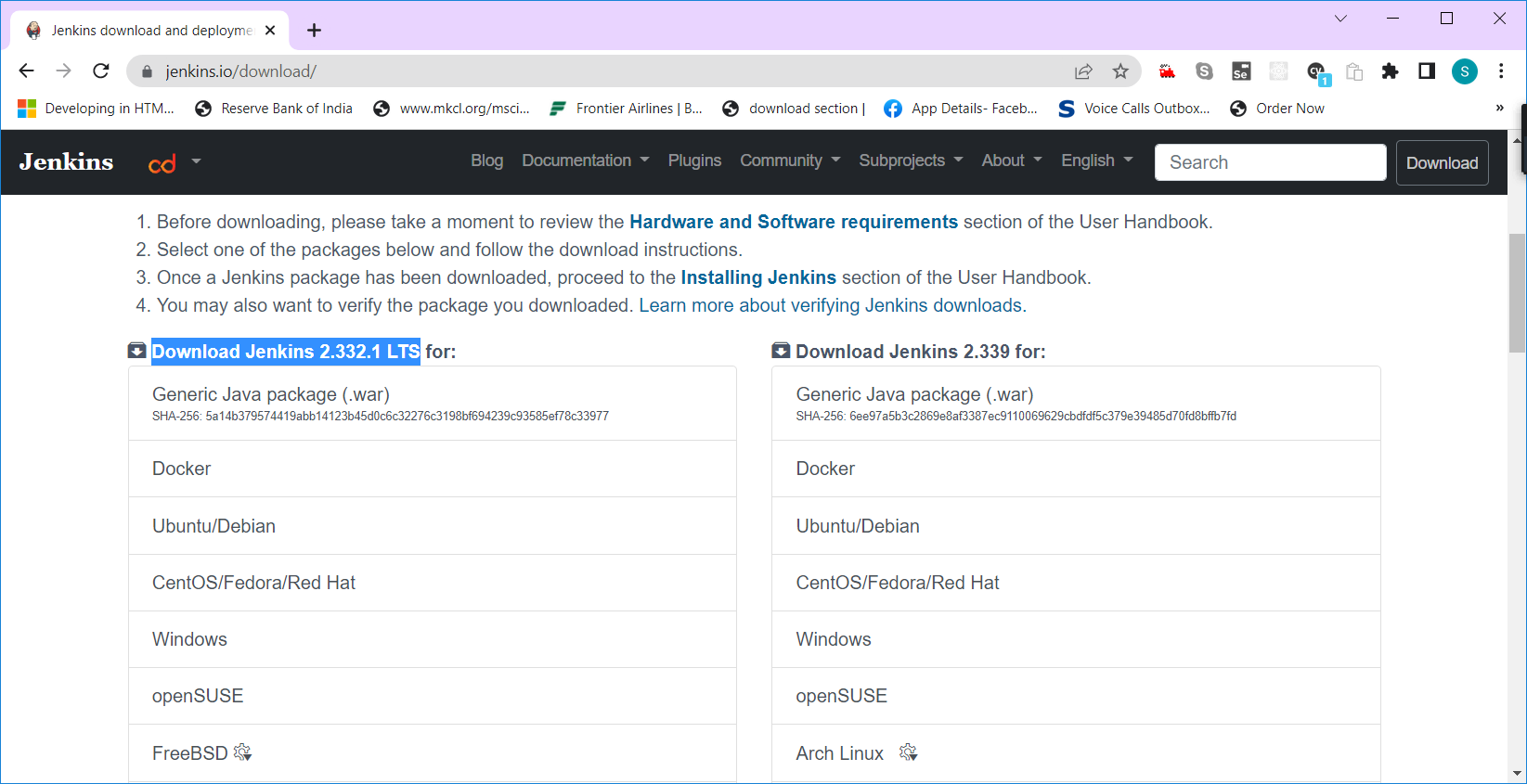
# DevOps Tools: Jenkins

Jenkins is a self-contained, open source automation server, which can be used to automate the building, testing and deployment of software. Jenkins can be installed standalone, through native system packages, or using Docker.

Official Site of Jenkins -- <https://jenkins.io>

Download Link -- <https://www.jenkins.io/download/>

**Download Jenkins 2.332.1 LTS – (Long Term Support)**



JAR – Java Archive

WAR – Web Archive

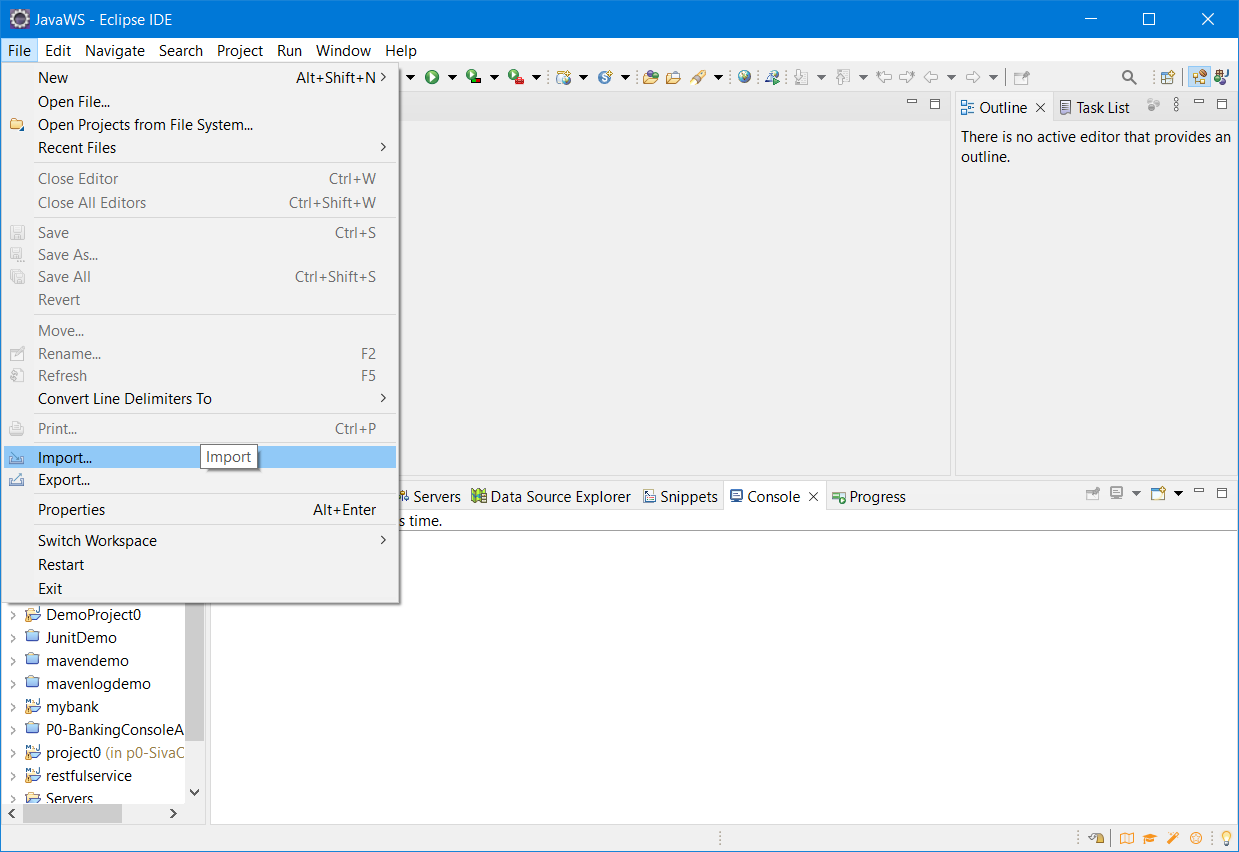
Different Ways of Using Jenkins

1. Stand -alone Jenkin (Download & Install)
2. Web Version (WAR file)
3. Dockerized Image

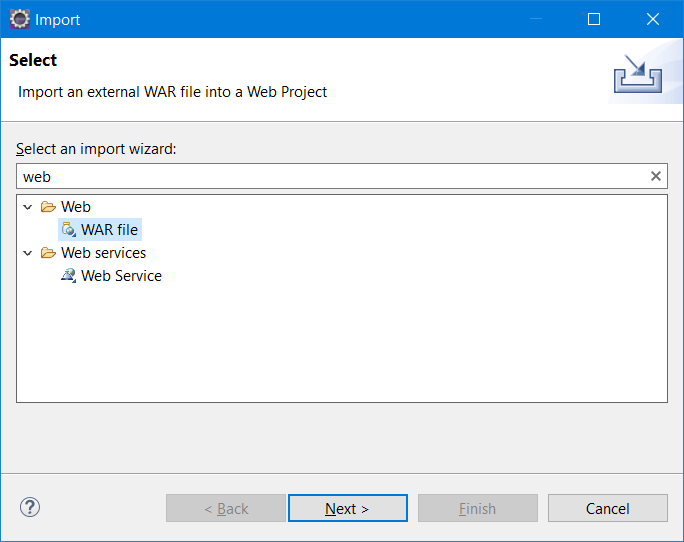
Method 2 : Using the WAR file

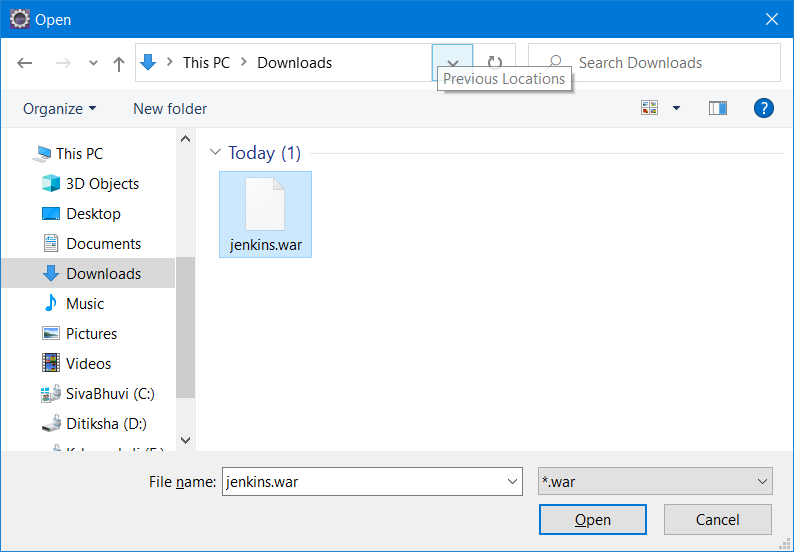
* Open Eclipse
* Import the WAR file (File 🡪 Import Project 🡪 War (under web)

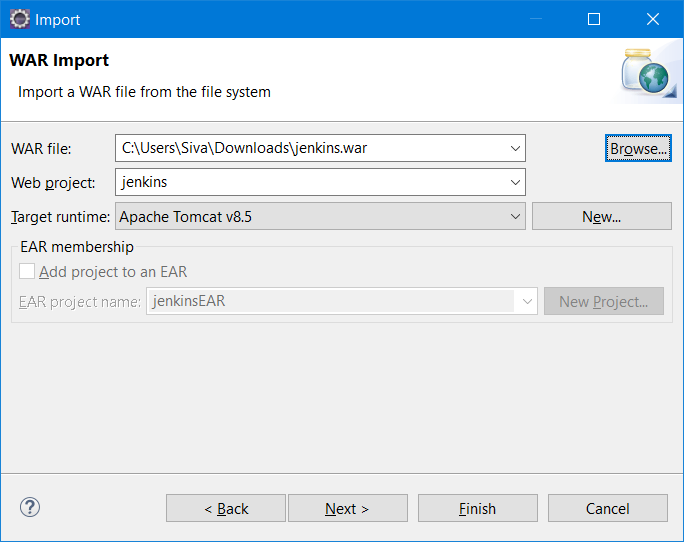




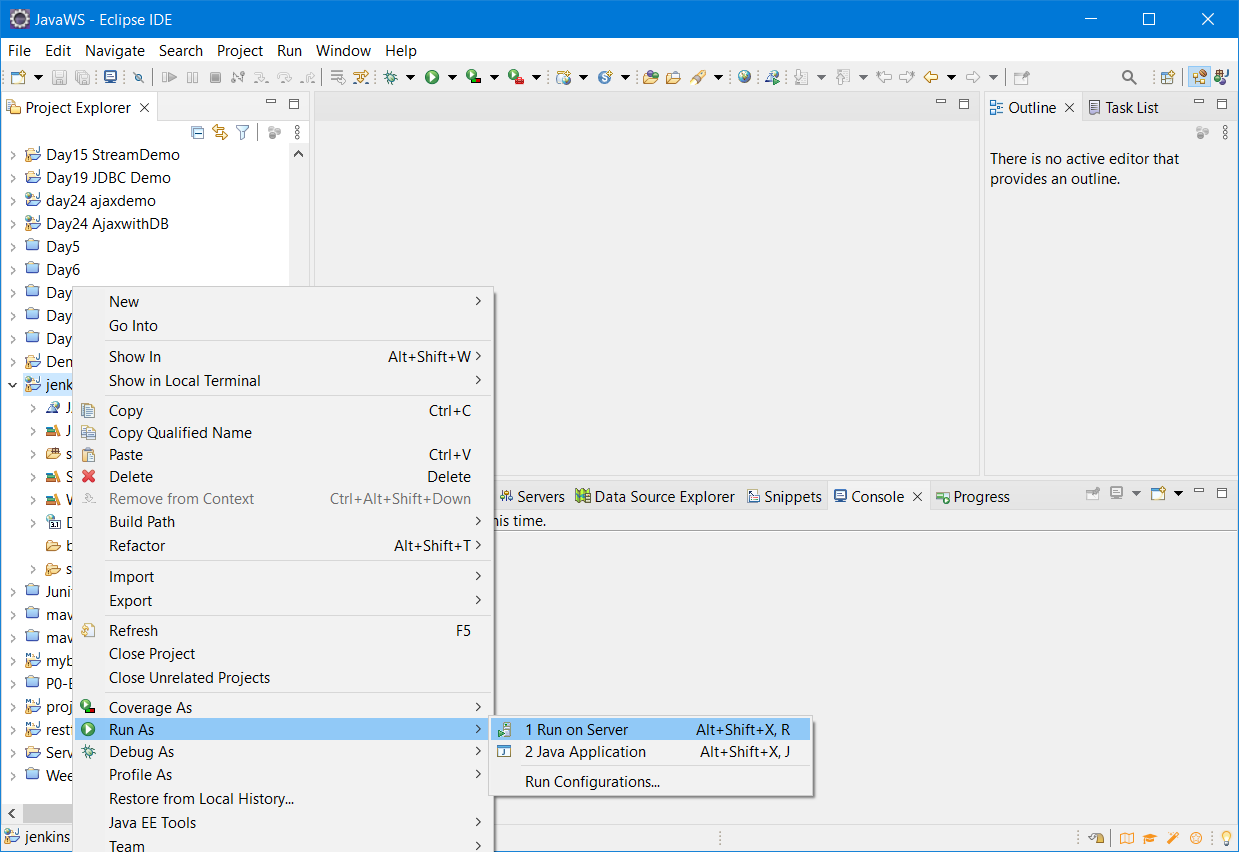


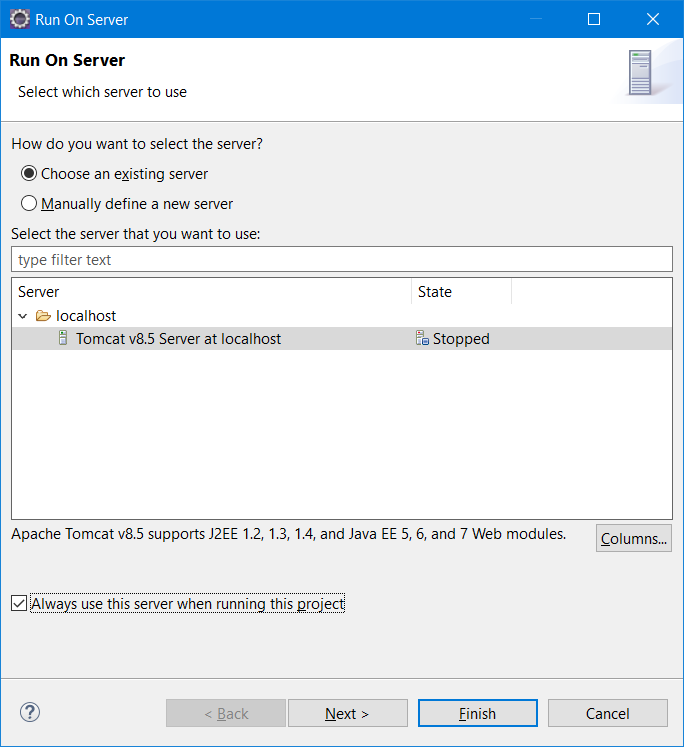






* Click on Finish button.

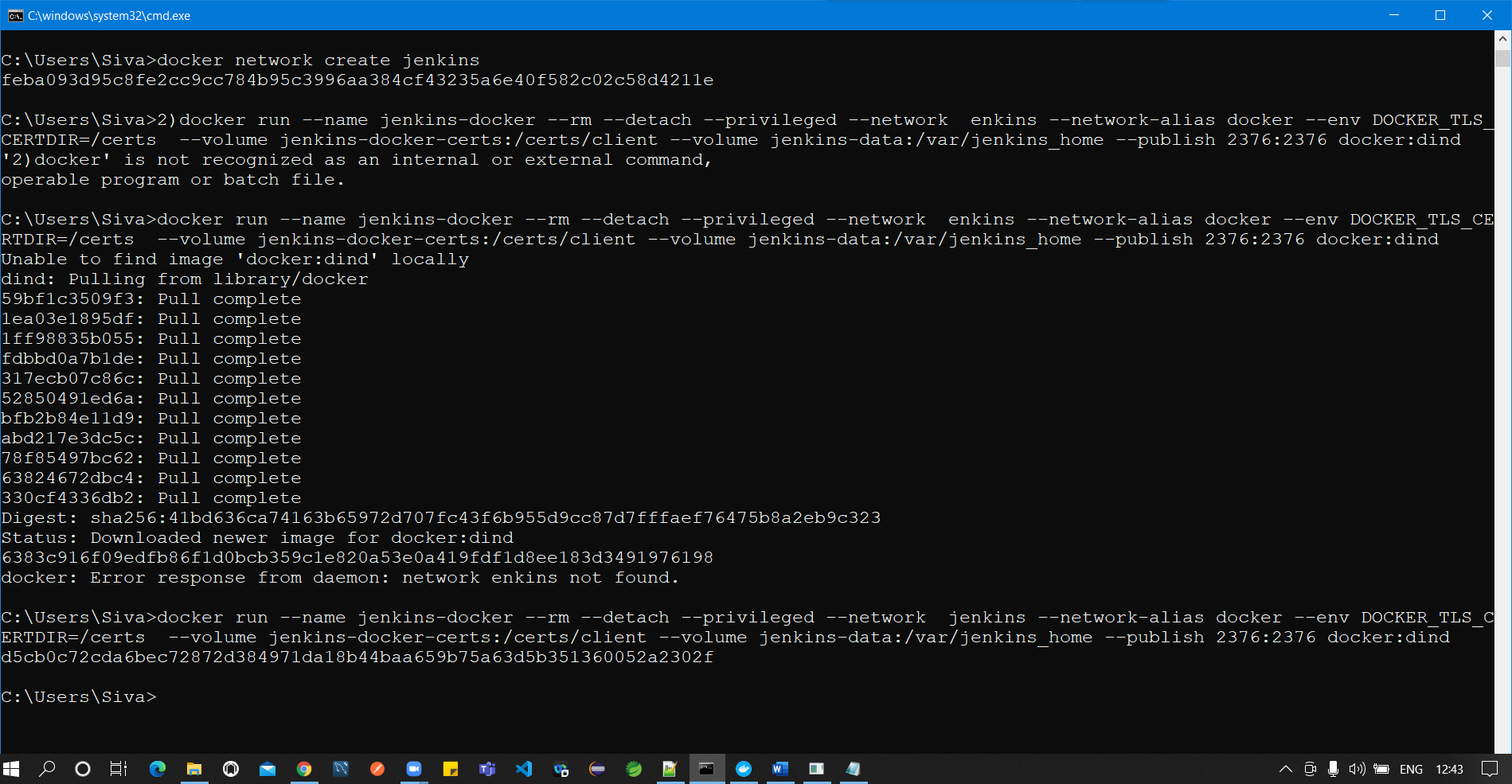




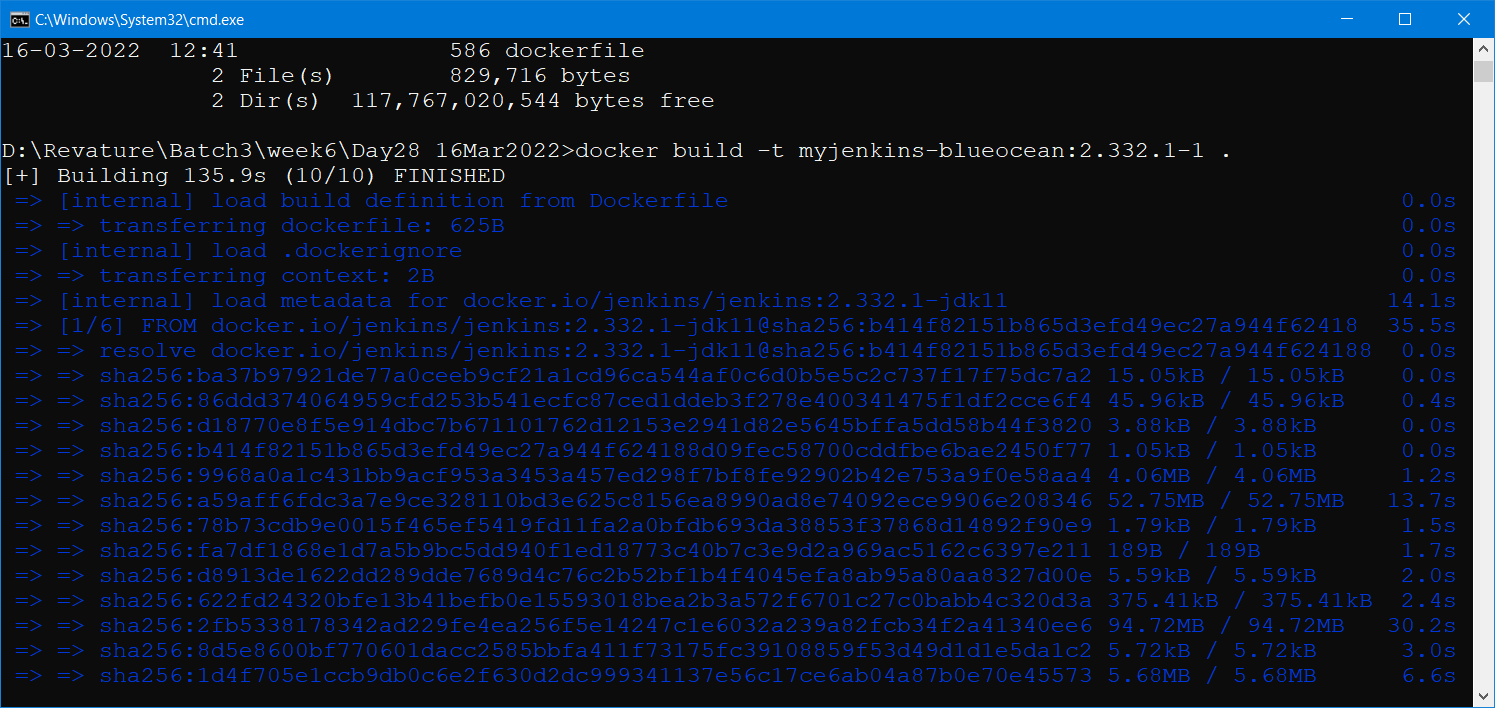
Method 3)

docker pull jenkins/Jenkins

1. Docker create network Jenkins
2. docker run --name jenkins-docker --rm --detach --privileged --network jenkins --network-alias docker --env DOCKER\_TLS\_CERTDIR=/certs --volume jenkins-docker-certs:/certs/client --volume jenkins-data:/var/jenkins\_home --publish 2376:2376 docker:dind
3. create a dockerfile with the following content “ “



1. docker build -t myjenkins-blueocean:2.332.1-1 .



1. docker run --name jenkins-blueocean --rm --detach --network jenkins --env DOCKER\_HOST=tcp://docker:2376 --env DOCKER\_CERT\_PATH=/certs/client --env DOCKER\_TLS\_VERIFY=1 --volume jenkins-data:/var/jenkins\_home --volume jenkins-docker-certs:/certs/client:ro --publish 8080:8080 --publish 50000:50000 myjenkins-blueocean:2.332.1-1

Maven – Project Management Tool

1. Build (Converting .java to .class)
2. Test (Executing all the Junit test cases)
3. Package (Creating a single jar/war file)
4. Deploy (deploying the jar/war in a server)
5. Manage Dependencies (pom.xml)
6. Generate Code (inside the target folder)
7. Clean (clean the target folder contents)

Maven is a Open Source Java based project management tool. Maven will help to automate many deployment steps.

Maven Lifecycle:

* When Maven builds your project, it goes through several steps called phases. The default maven lifecycle is:

1. Validate => project is correct and all necessary information is available (mvn validate)
2. Compile => compiles project source code (mvn build)
3. Test => tests all compiled code (mvn test)
4. Package => packages all compiled code to WAR/JAR file (mvn package)
5. Integration => performs all integration tests on WAR/JAR
6. Verify => runs checks on the results of integration tests (mvn verify)
7. Install => installs WAR/JAR to local repository (mvn install)
8. Deploy => copies final WAR/JAR to the remote repositor (mvn jarfile/warfile)

The back bone of maven project is pom.xml

POM – Project Object Model.

Mvn command can be executed in the folder where pom.xml file is available.

SONAR – It’s a Quality Checking Tool – It will help you to analyse the code quality of any project.

Official site -- <https://www.sonarqube.org/>

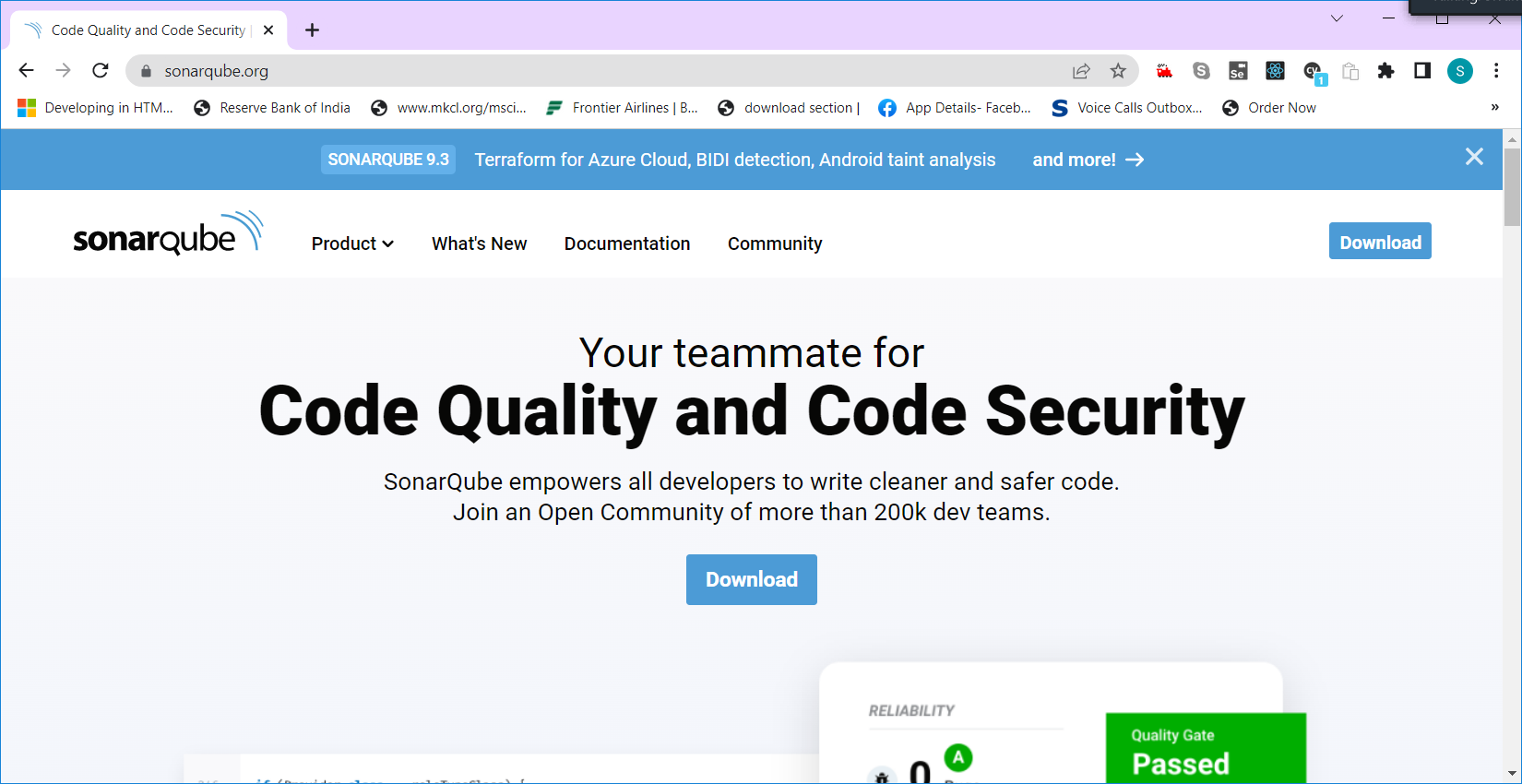
To Run SonarQube – JDK11 is the minimum requirement

JDK11 download link –

SonarQube – Stand-alone Server (Download & Install)

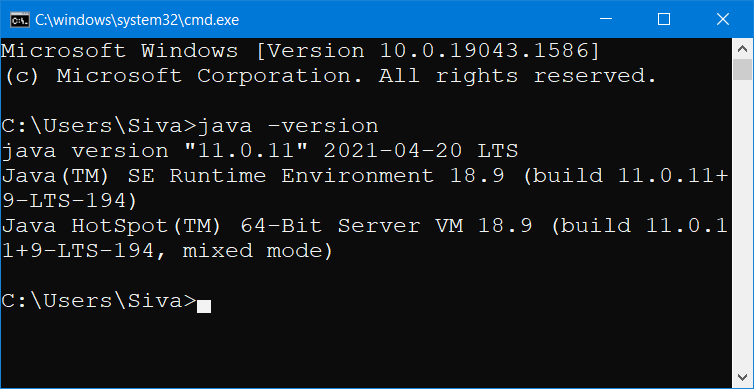
Sonar Cloud – Cloud version of SonarQube (No need to download, directly access it in browser)

SonarLint - Eclipse Plugin (Integrate with Eclipse)



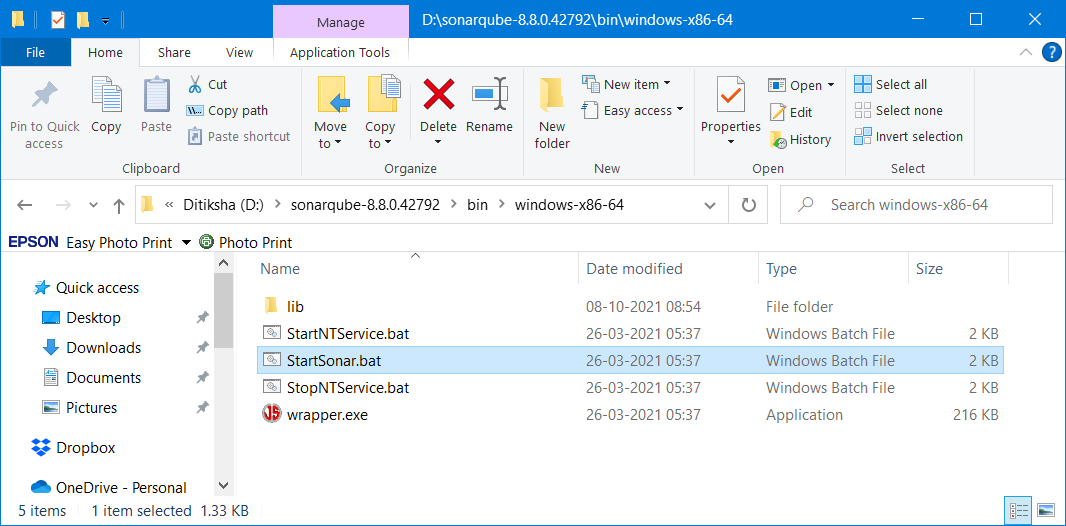
To Run the SonarQube JDK11 is the minimum requirement.

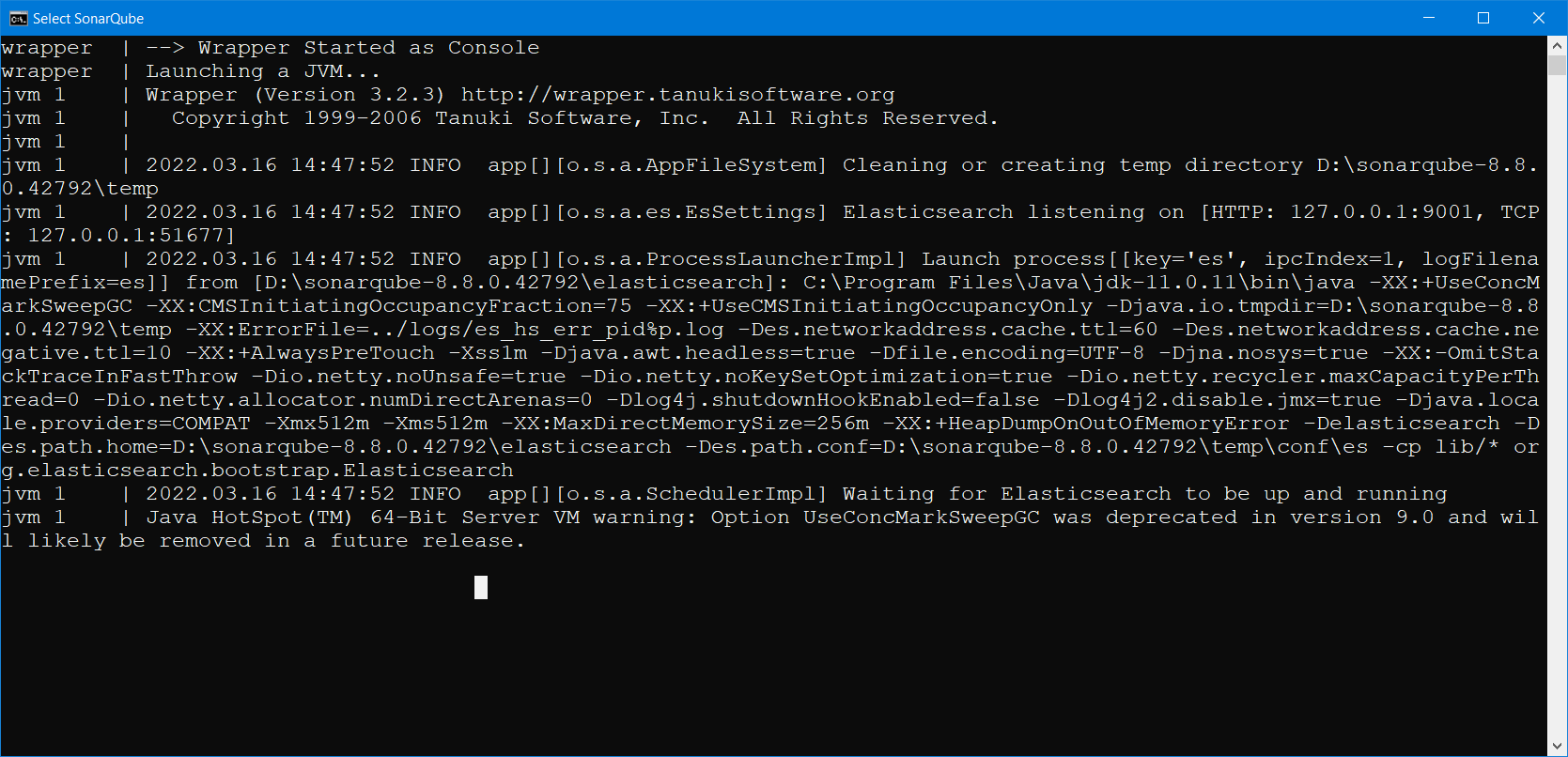
In System environment variable Update JAVA\_HOME to JDK11 or add jdk11 to path.

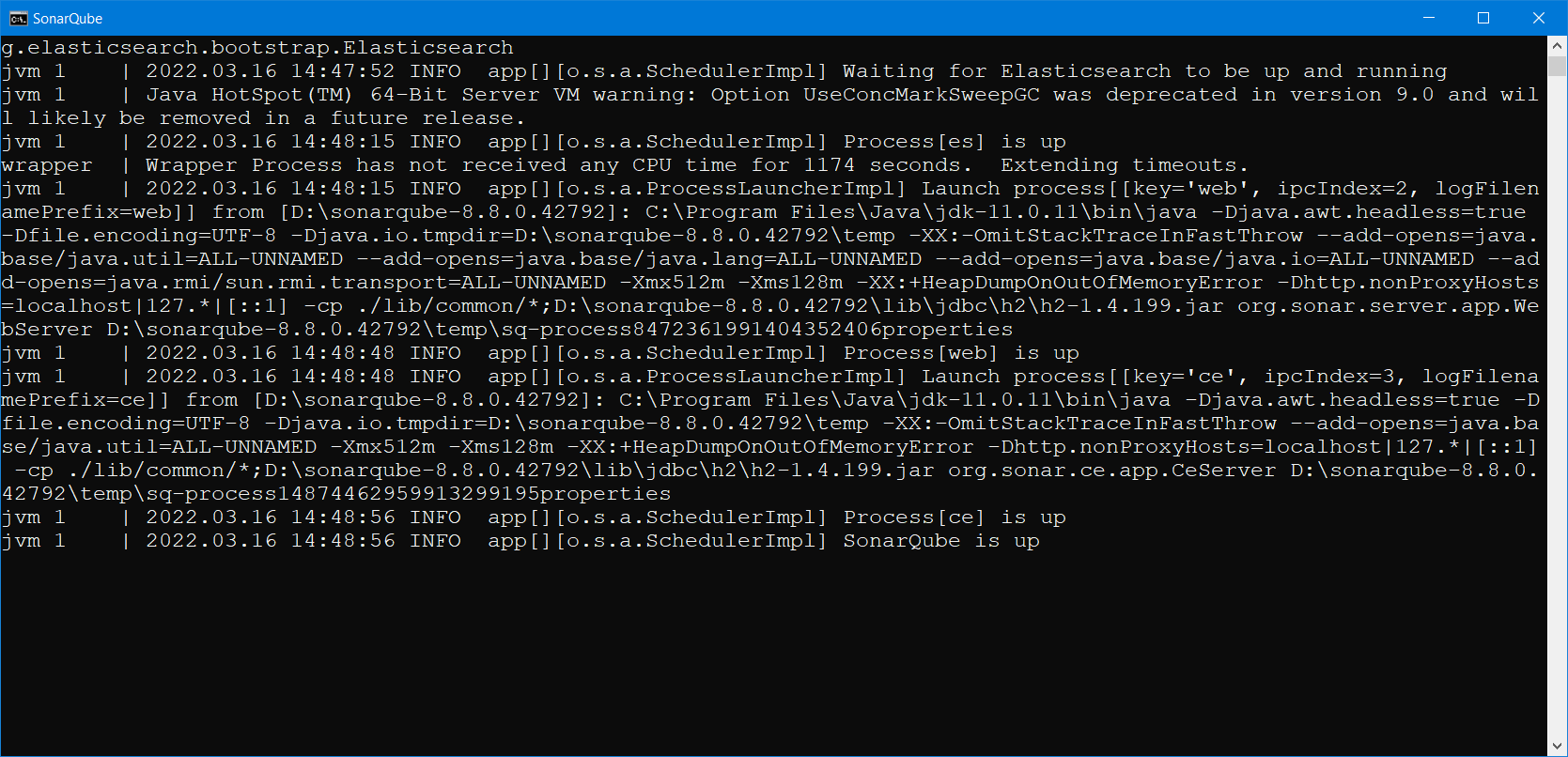


Download & Extract SonarQube zip file.

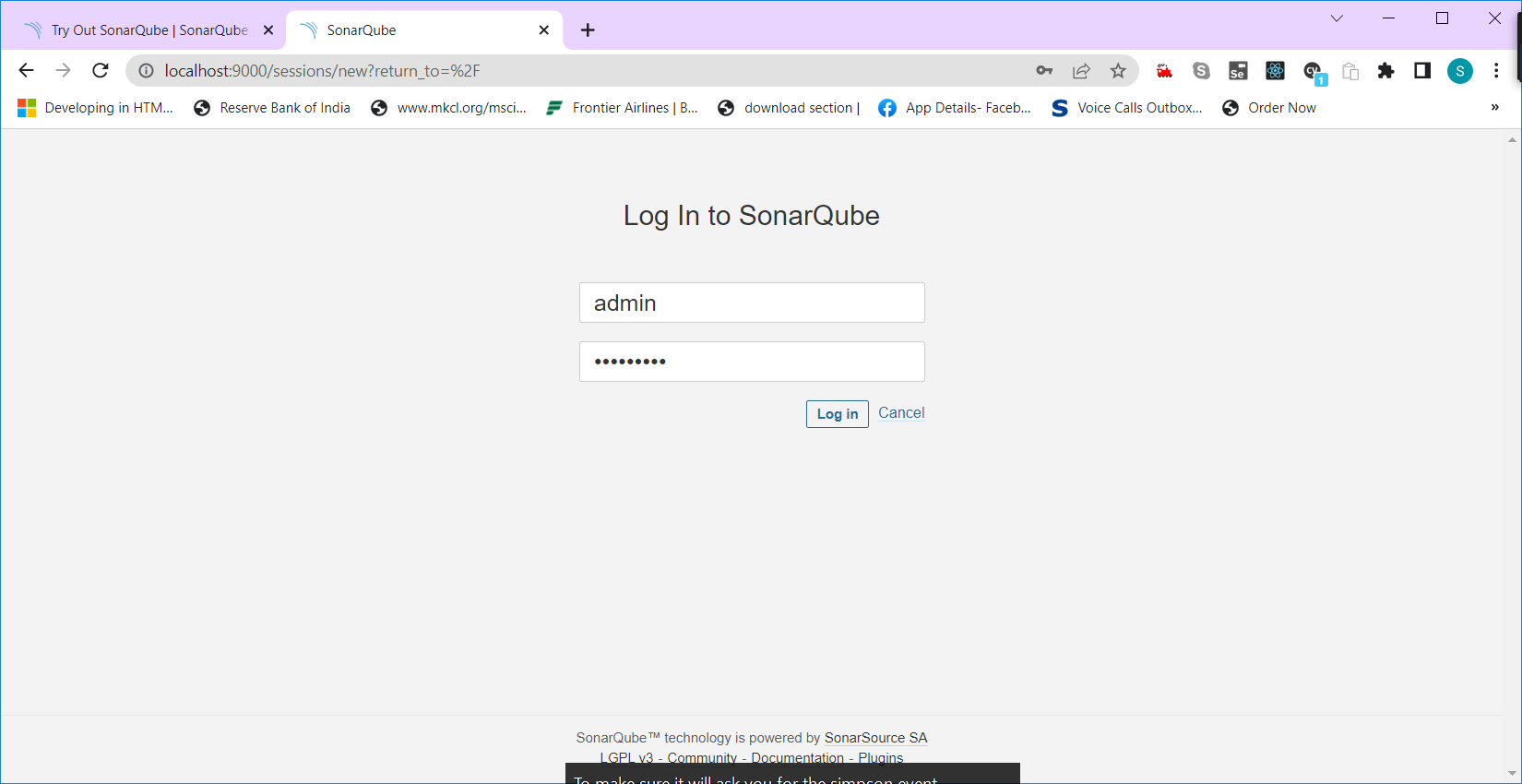
Open the folder



Run “StartSonar.bat”  




Open localhost:9000

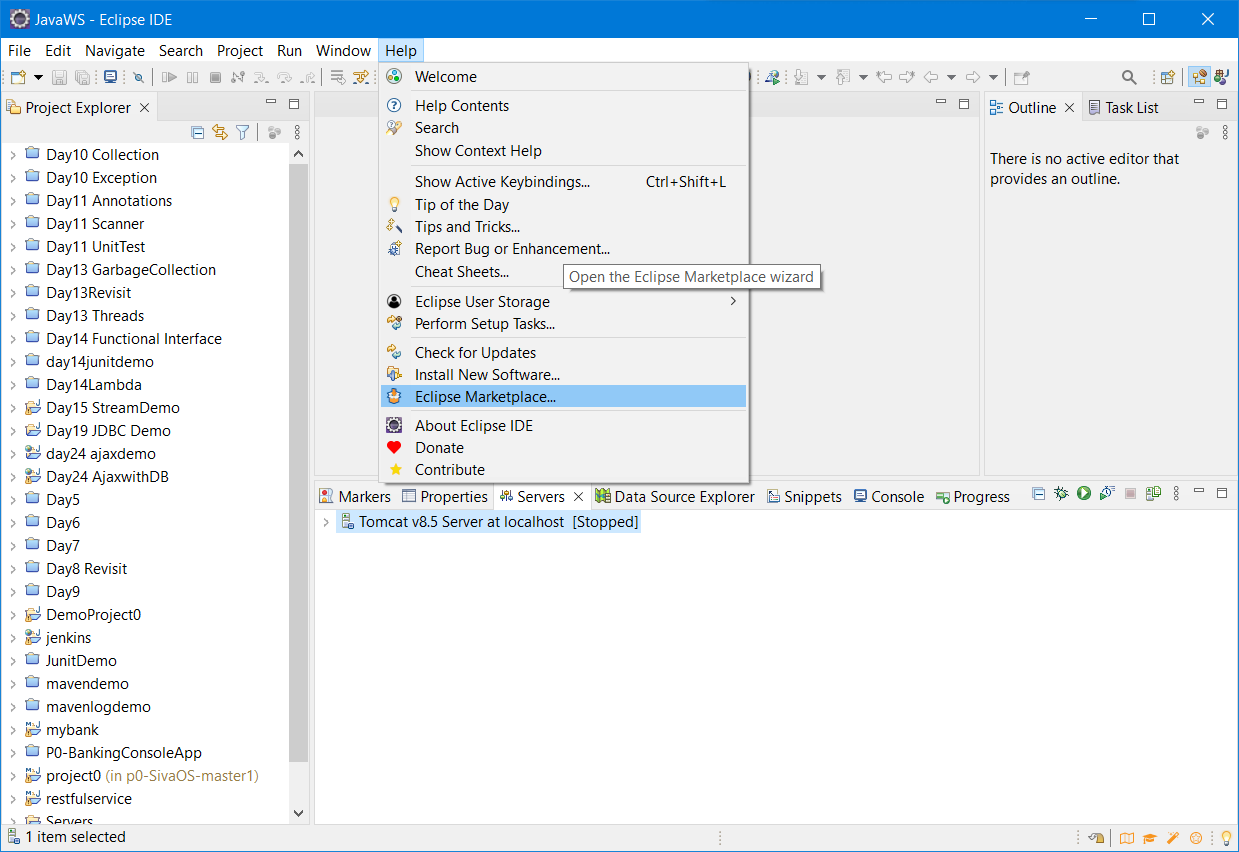


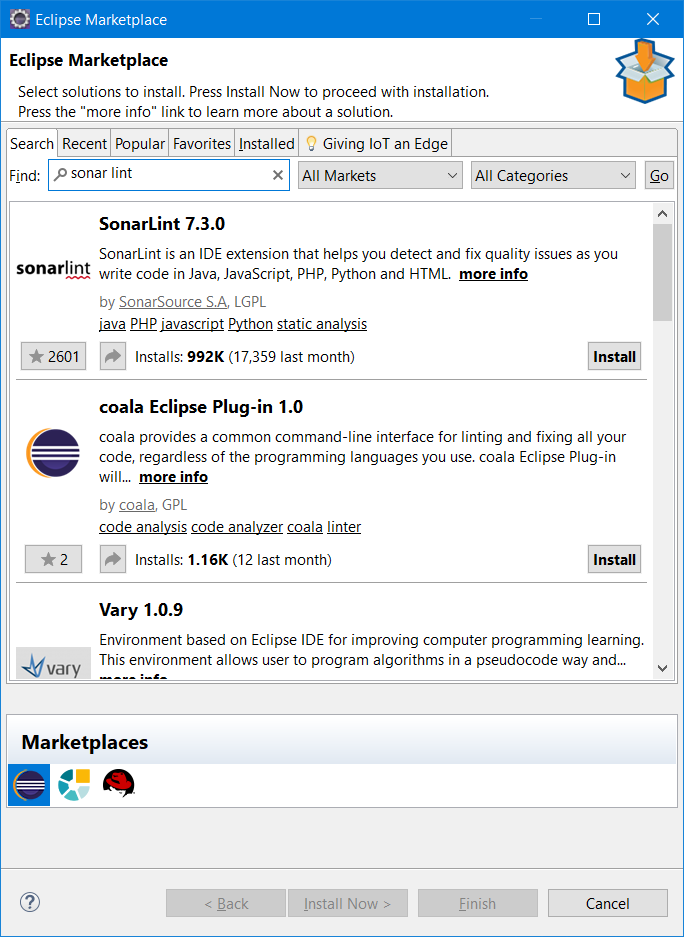
Default username/password is admin/admin

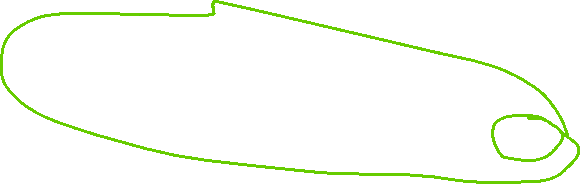
After successful login change the password.

Using Sonar Lint in Eclipse

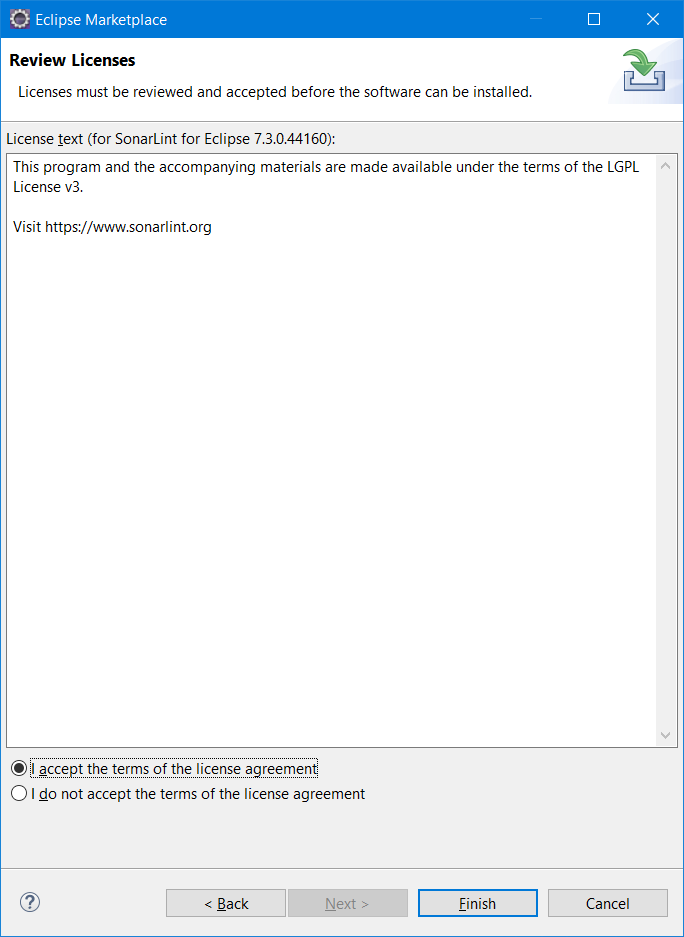
* 1. Open Eclipse
  2. Open “Eclipse Market Place” in Help menu

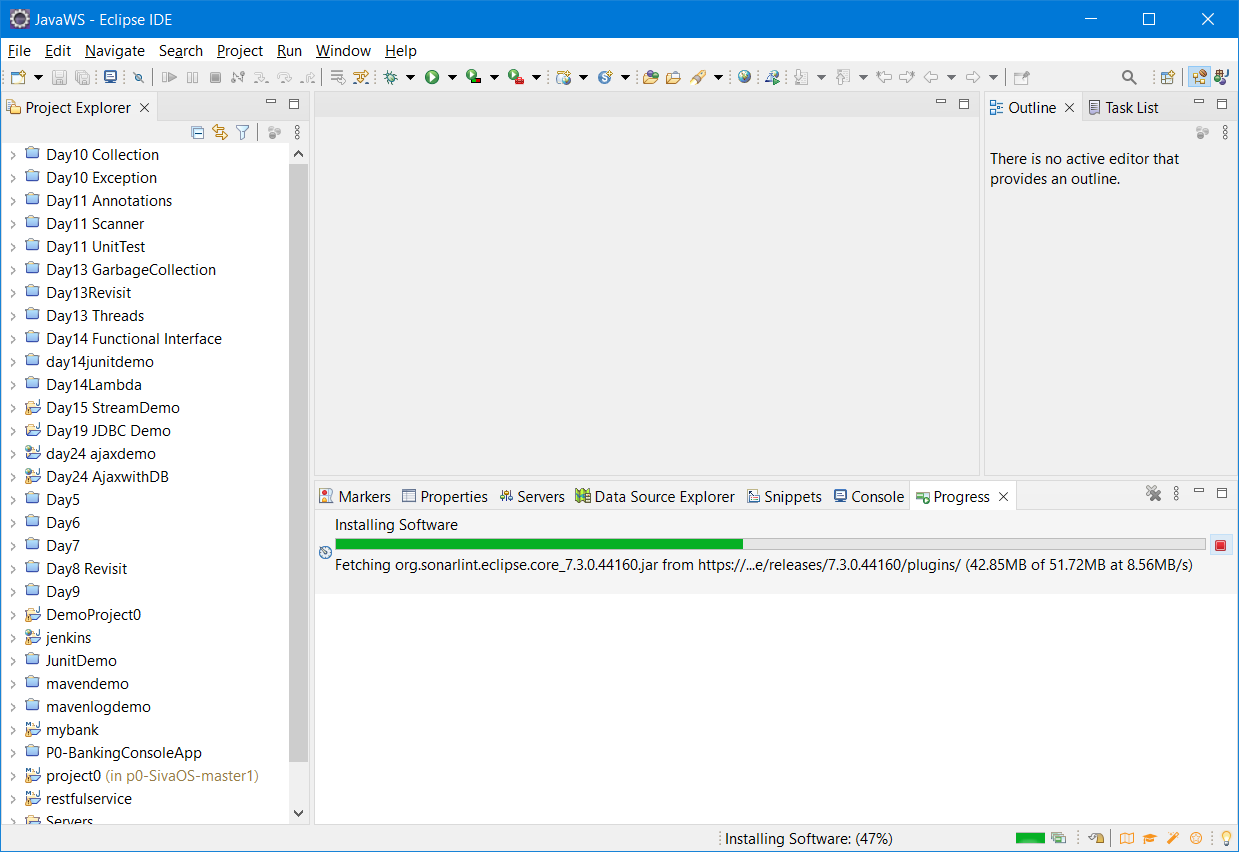


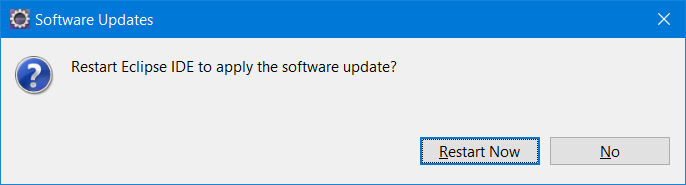


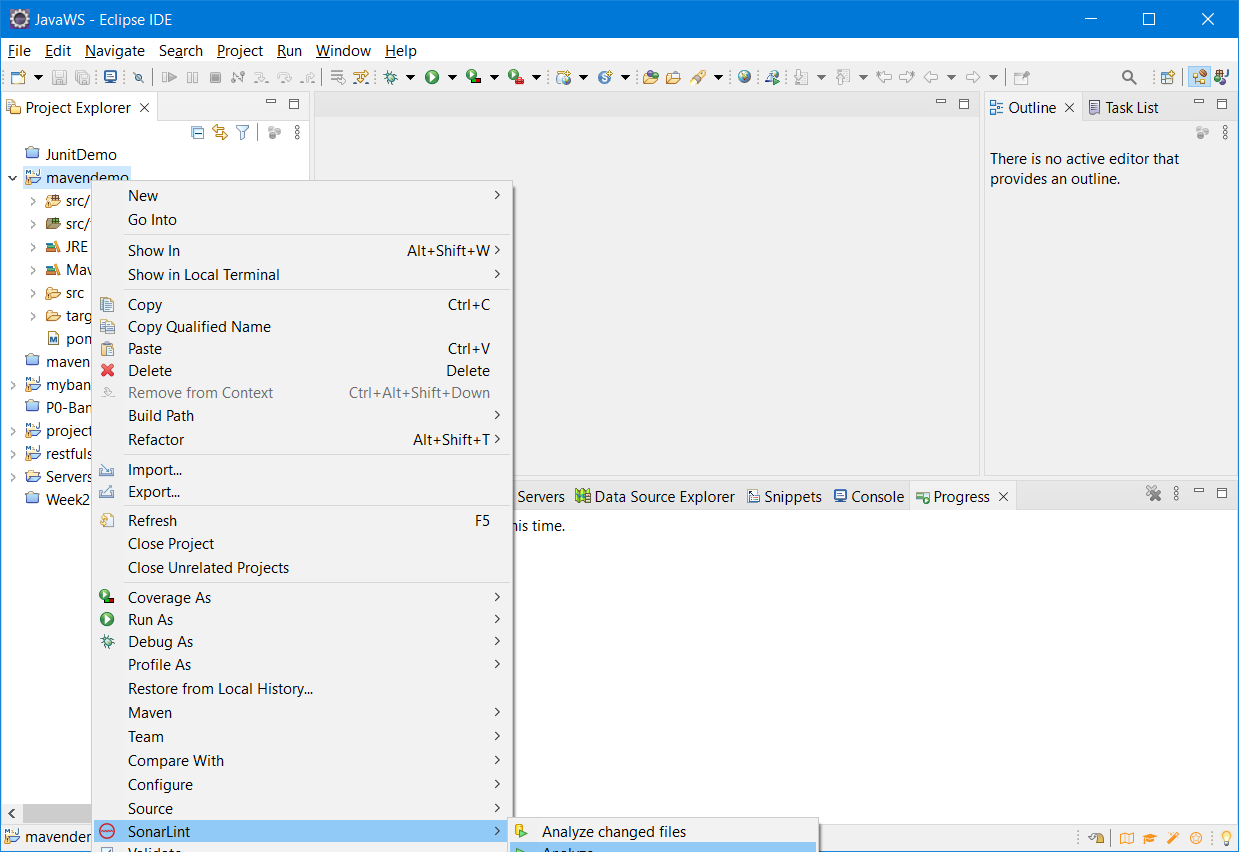


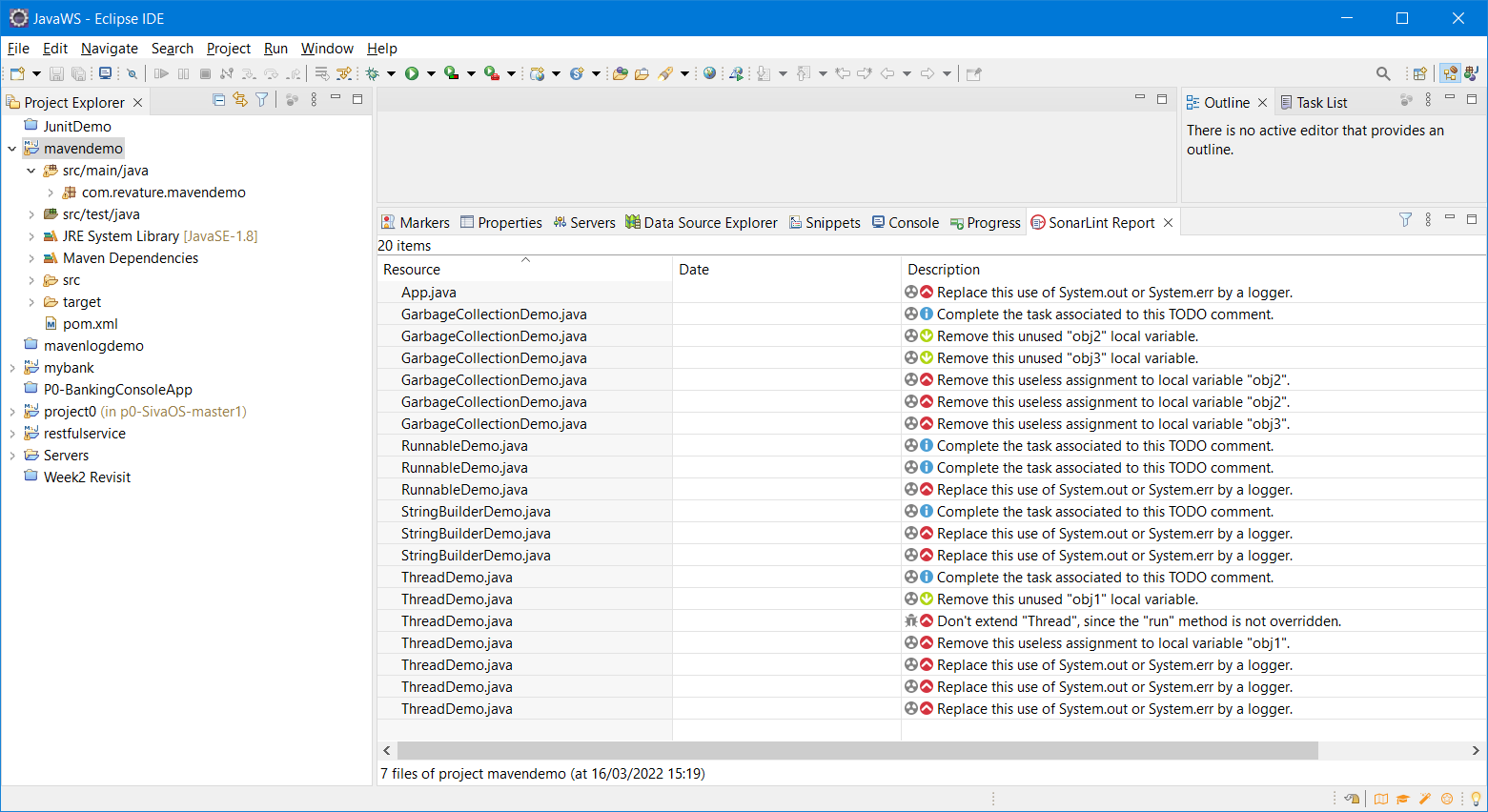
Click on “Install” button.











CI/CD – DevOps, Maven, SonarQube, Sonar Lint, Sonar Cloud (Cloud version of Sonar Qube)