**CI-CD pipeline for Jenkins**

**What is DevOps?**

*DevOps is the set of practices that collaborate with Developers and IT Operation Roles. i.e DevOps is not a Tool, it is the process or culture that the engineer has to follow certain procedures that establish collaboration between Developers and IT Operational Engineer to fasten the application delivery in a fully automated fashion.*

**Uses of DevOps :**

*DevOps primarily works based on below three principles to fasten the application deployment.*

**Principals:**

1. *Communication*
2. *Collaboration and*
3. *Integration*

**DevOps Tools for building the application**

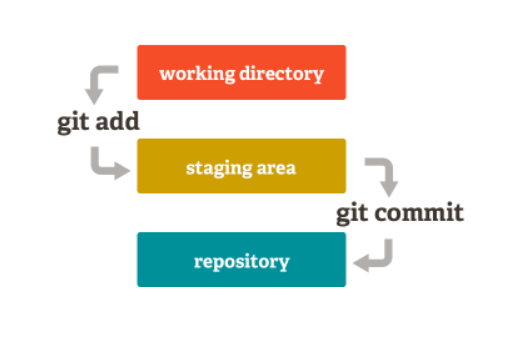
* *Git*
* *Bit Bucket*
* *Maven*
* *SonarQube*
* *Jfrog*
* *Ansible*
* *Docker*
* *Kubernetes*
* *Jenkins*

**Git :**  

*Git is a Distributed version control tool or source code management tool,*

**Version control system:**

*Version control software keeps track of every modification**to the code in a special kind of database. If a mistake is made, developers can turn back the clock using rollback or revert and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.*



**Working Directory**: *This is the project home folder where the source code files are stored.*

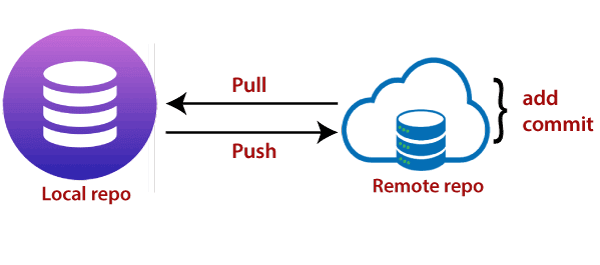
**Staging: staging:** I*s the temporary space in the Git control before actually committing the changes to the Git Repo.*

**Git Repository:** *This is the .git folder in the project folder and it keeps all the code changes and below metadat*a.

**GitHub/BitBucket**

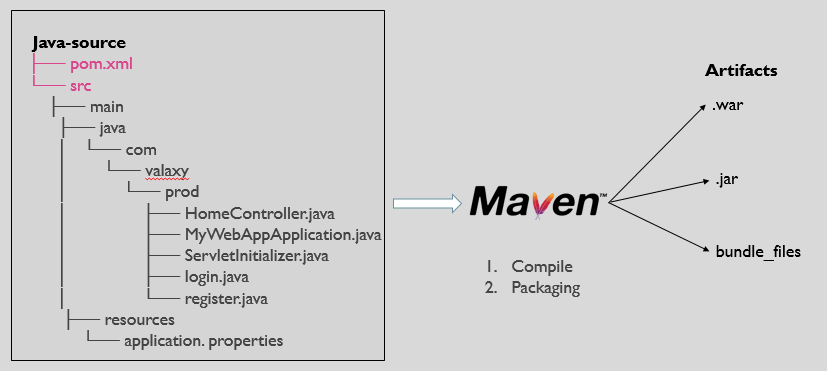
*Bitbucket is the distributed version control system to store the source code changes.*

*This is a hosting service.*



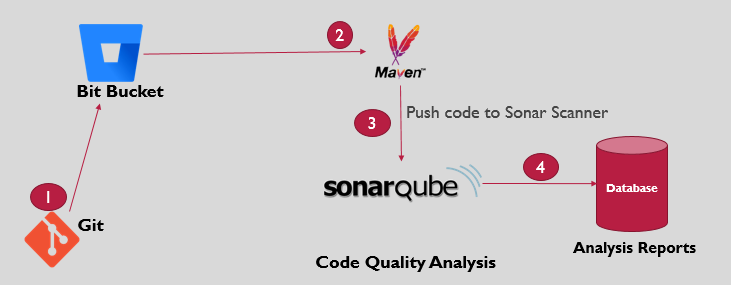
**Maven**

*Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.*

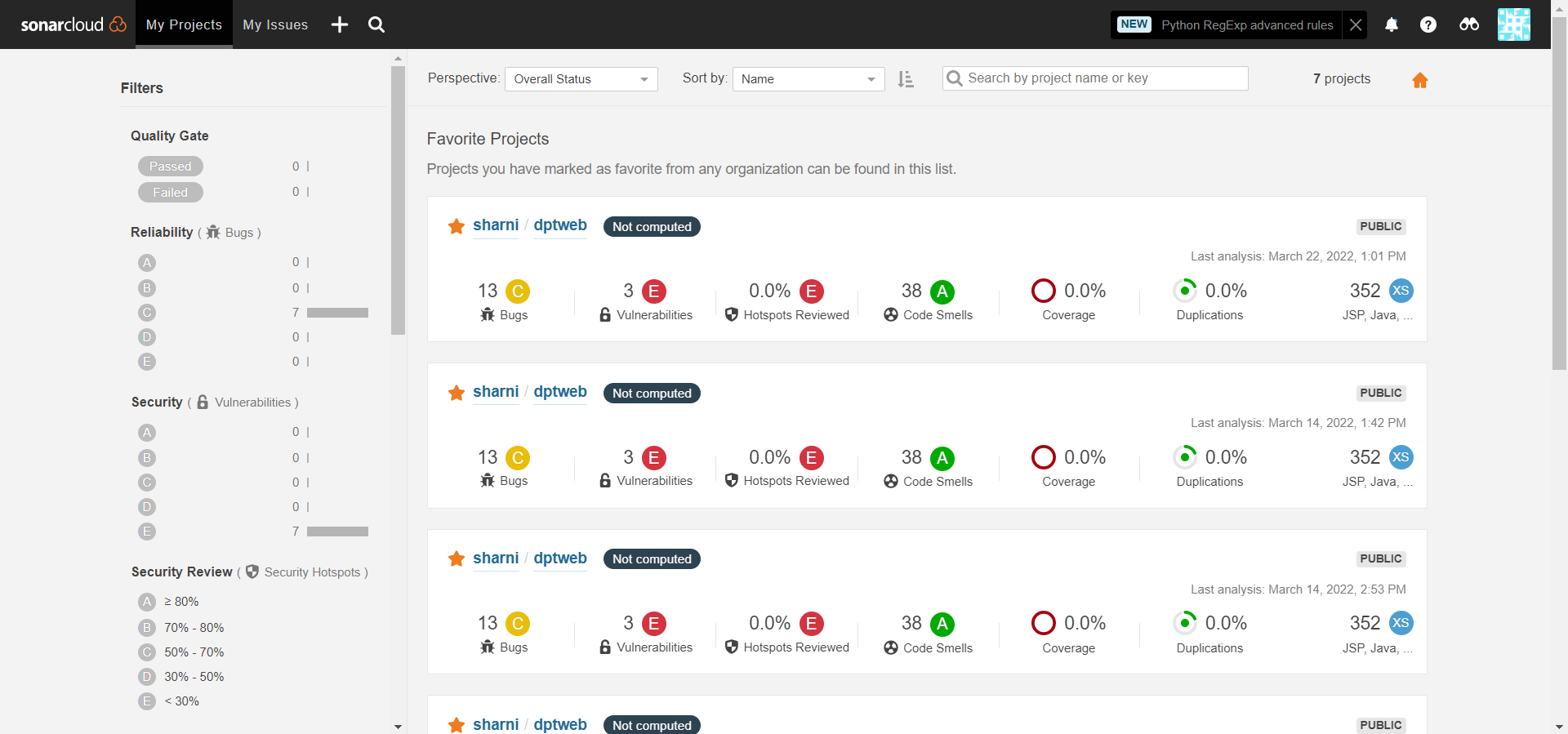


**SonarQube**

*SonarQube is an open-source to scan the source code for code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.*



SonarQube Page

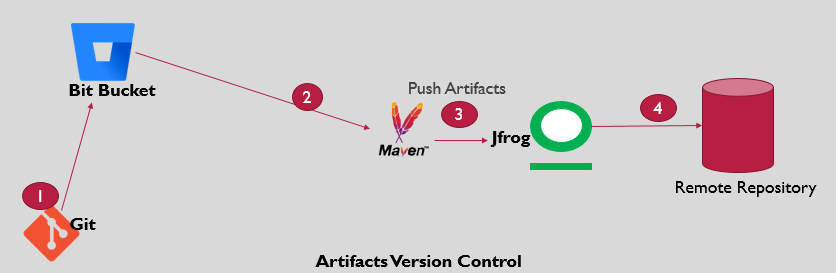


**JFrog**

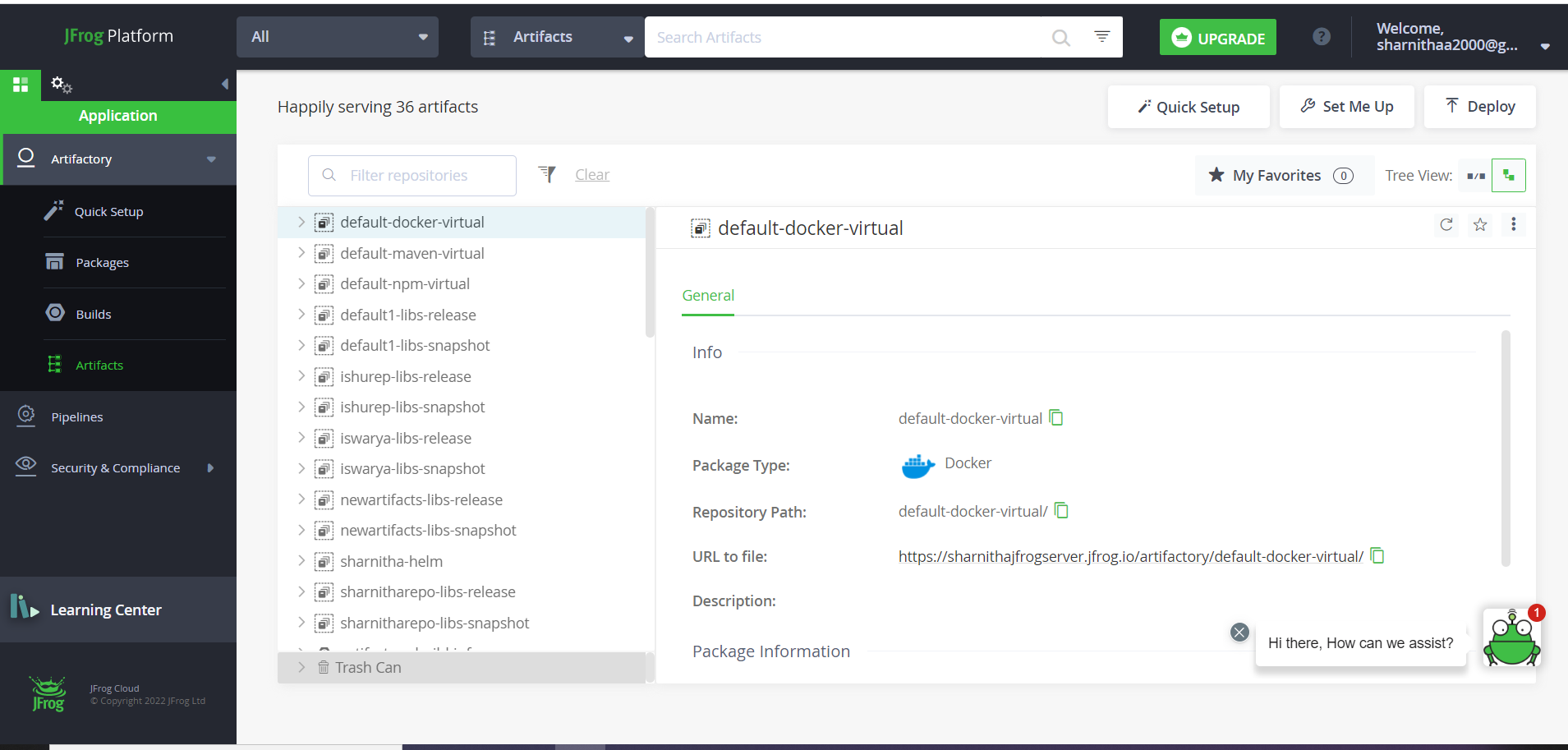
*Jfrog Artifactory is the Artifact repository management to store, move and delete artifacts that are deployed by Apache Maven Build in this DevOps Project. Artifactory can store the artifacts in version control.*

*Repositories that auto configured for Maven Builds:*

* *libs-release*
* *libs-snapshot*
* *libs-release-local*
* *libs-snapshots-local*



JFrog Page



**Docker**

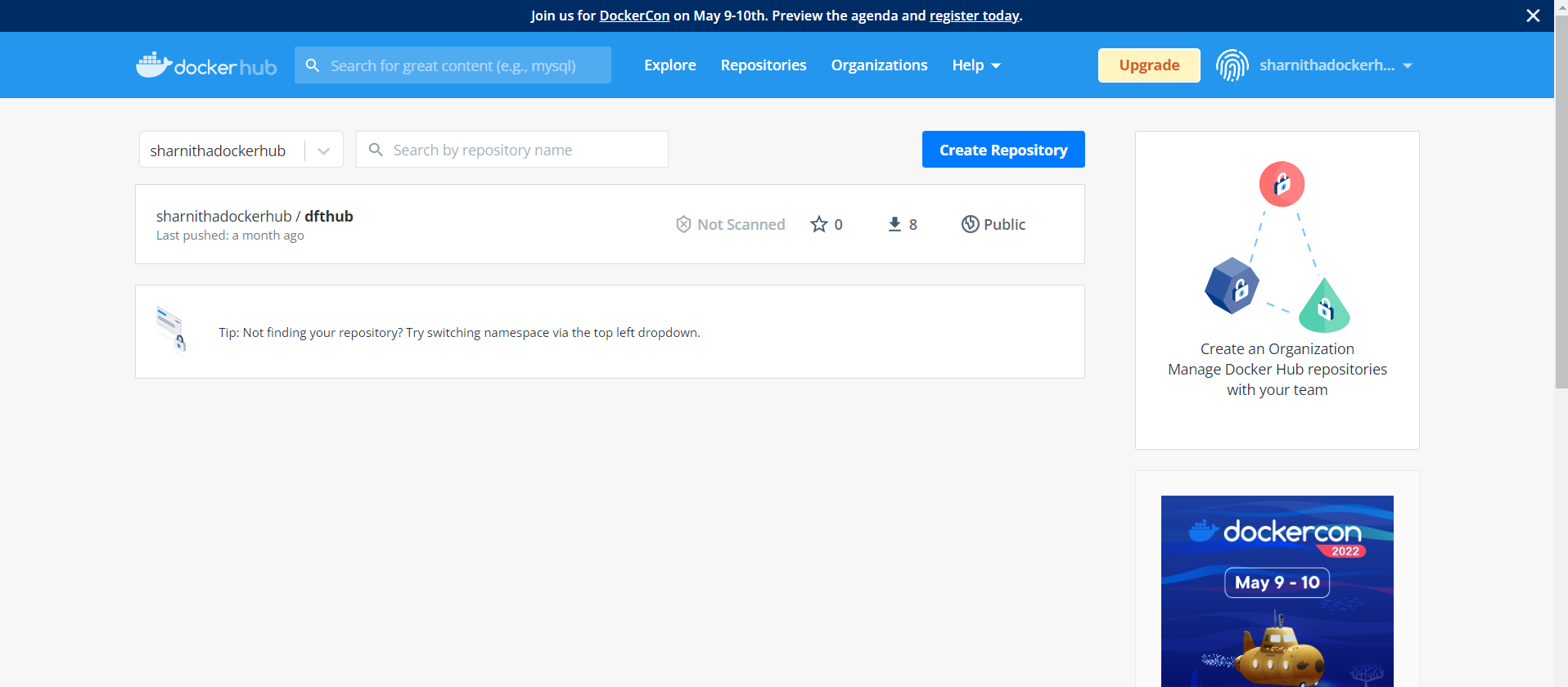
*Docker is the container engine that handles create, managing containers. Docker Engine is an open source containerization technology for building and containerizing applications.*

***Container:***

*It is a smallest unit of software that can run application and dependencies*

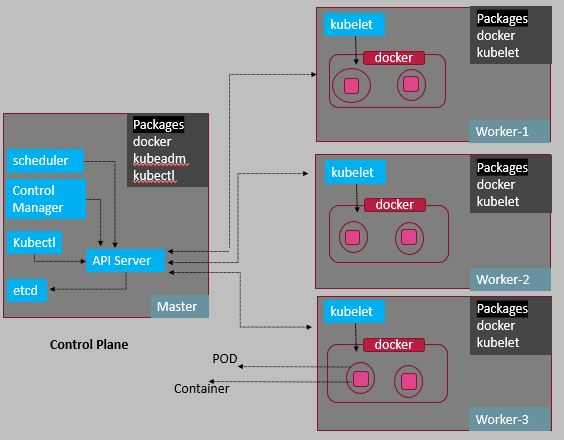
***DockerHub***

*Docker Hub is* ***a hosted repository service provided by Docker for finding and sharing container images with your team***



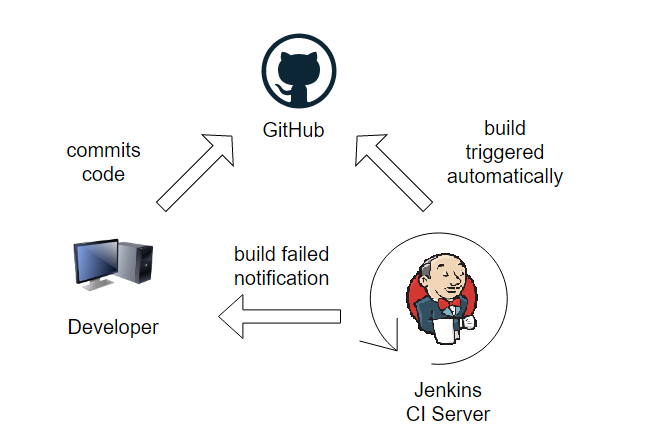
**Kubernetes**

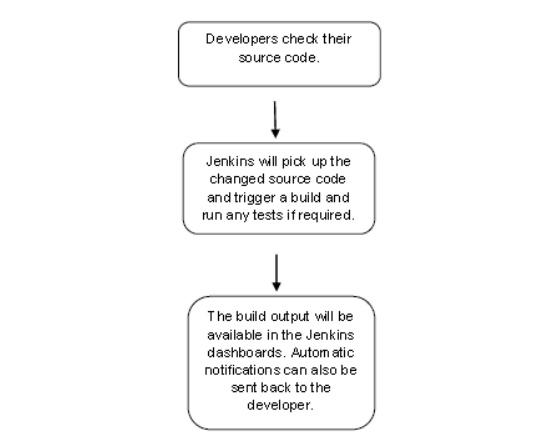
* *Kubernetes is an open-source, portable, cluster managed orchestration framework. Kubernetes allows containerized applications to run on multiple clusters for more reliable accessibility and organization.*
* *Kubernetes supports large scale deployments hence is best suited for enterprise-level containers and cluster management.*



**Jenkins**

*Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used* ***to implement CI/CD workflows, called pipelines****.*





**Implementation**

**CI-CD Pipeline for Jenkins in AWS Console**

**Prerequires:**

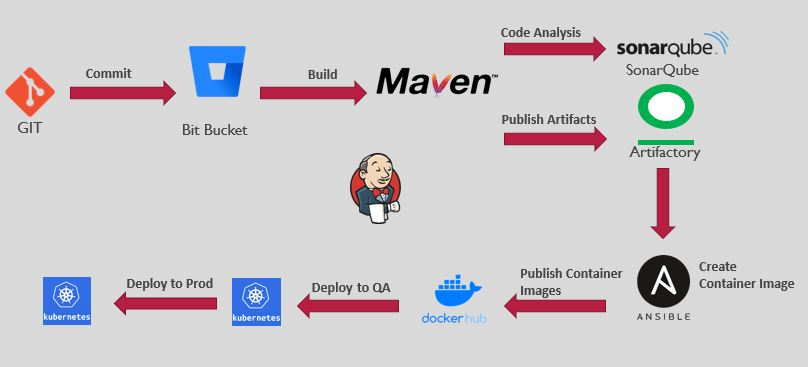
**1.Install the Jenkins**

**Note: The Jenkins CI-CD is divided into two parts,**

**1. CI Pipeline 2. CD Pipeline**

**Jenkins CI**

*Continuous integration (CI) helps developers merge their code changes back to a shared branch, or “trunk,” more frequently—sometimes even daily. Once a developer’s changes to an application are merged, those changes are validated by automatically building the application and running different levels of automated testing, typically unit and integration tests, to ensure the changes haven’t broken the app. This means testing everything from classes and function to the different modules that comprise the entire app. If automated testing discovers a conflict between new and existing code, CI makes it easier to fix those bugs quickly and often.*



**Explanation :**

1. *The Developer push the code in the Bitbucket or GitHub etc..(Hosting service),*
2. *we should build the code by using maven*
3. *We should check the code quality by using SonarQube*
4. *We should deploy our artifacts in the JFrog artifactary*
5. *We should create an image by using the docker and deploy the war file in the DockerHub*
6. *Helm*
7. *Kubernetes*

**Real-Time Commands:**

1*.Create an instance using the aws console*

*2. open the putty or mobaxterm using the private key*

*3.yum install git –y ----->install git*

*4. amazon-linux-extras install jdk11 ------------>Install Java*

*5. https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.zip (Maven)*

*6.cd /opt*

*7.Unzip the maven binary zip file*

*8. git clone< Url>*

*8.1 Give the Credentials*

*9./opt/apache-maven-3.8.5/bin/mvn package ---------------->For Build the code*

*10****.https//SonarCloud.io*** *------------->(For checking the code quality in sonar )*

*10.1 SONAR\_TOKEN*

*10. 2 Value of the environment variable*

*10.3* *Update your pom.xml*

*(Credentials)*

*11./opt/apache-maven-3.8.5/bin/mvn configure -----------------> For checking the code quality in sonar*

*12****. https://JFrog.com***

*/opt/apache-maven-3.8.5/bin/mvn deploy ------------->Deployed the war file in the JFrog artfactary*

*13. Docker is used for run the containers [TO RUN CONTAINER ,THE CONTAINER IMAGE IS IMPORTANT]*

*Container image has all the information about the application*

***Scenario: To run the apache tomcat as a container***

*Amazon-linux-extras install docker –y*

*cd /var/lib/docker*

*Ls –l*

*Service docker status*

*Service docker start*

*Pwd*

*To Create a Container*

*In Docker Hub,----------> search tomcat image*

*Docker pull tomcat*

*Docker image*

*Docker run –d –name cont1 tomcat*

*To access it from the browser,Copy the IPV4 address:8080/Sharniweb-3.0*

*1. Check the Security group is enabled in 8080*

*Even it doesn’t work,*

*2. Do Port mapping*

*Docker run –d –name cont1 –p 8080:8080 tomcat:8.5*

*To push the image in the Docker Hub*

*Docker tag image name username in dockerHub / Image name*

*Create pods using kubernates*

*Kubectl get pods*

*Kubectl get nodes*

*Wget* [*https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz*](https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz)

*Gunzip* [*https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz*](https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz)

*Git clone<url>*

*Cd linux-amd64/*

*./helm version*

*Cd kubernates/*

*Ls –l*

*Cd helm/*

*Cd dpt-helm-sample*

*Cd ..*

*Helm package dpt-helm-sample*

*Helm install webapp dpt-web-app-1.tgz*

*Helm list*

*Curl –u sharnithaa2000@gmail.com:Sharnitha123@ -T Sharnitha-helm* [*https://sharnithajfrogserver.jfrog.io/artifactory/sharnitha-helm/dpt-web-app-1.tgz*](https://sharnithajfrogserver.jfrog.io/artifactory/sharnitha-helm/dpt-web-app-1.tgz)

**\*\*CI-CD Pipeline for Jenkins in AWS Console\*\***

**Jenkins Implementation:**

**Instead of going all the steps , we using Jenkins,, Basically jenkins is a CI/CD integration tool, we just want to select the required plugins to run our application**

**Prerequires:**

**Copy the Credentials at one location that we have to use in the application**

**1.Install the Jenkins**

*Create one EC2 instance in aws console called Jenkins\_server , select instance type as t2.medium , launch instance*

*To install is:*

* *Git*
* *Jdk*
* *Maven*
* *Docker*
* *Kubectl*
* *Helm*
* *Jenkins*

*Commands:*

*JDK 11 – amazon-linux-extras install java-openjdk11 –y*

*Git – yum install git –y*

*Maven –cd /opt*

[*https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.zip*](https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.zip)

*Docker – yum install docker*

*Kubectl – curl –LO -*[*https://dl.k8s.io/release/$(curl*](https://dl.k8s.io/release/$(curl) *-L -s* [*https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"*](https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl%22)

*Chmod +x kubectl*

*mv kubectl /usr/local/bin/kubectl*

*Kubectl version*

*Helm -* [*https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz*](https://get.helm.sh/helm-v3.8.2-linux-amd64.tar.gz)

*Jenkins -* [*https://pkg.jenkins.io/redhat-stable/*](https://pkg.jenkins.io/redhat-stable/)

*service jenkins start*

*Service jenkins status*

*Ipv4:8080 -----> we can see the jenkin page*

***Required Plugins:***

* *Git*
* *BitBucket*
* *Maven*
* *Sonar*
* *Docker*
* *Kubernetes*
* *Credentials binding*

***Credentials Requirements:***

***Go to Manage Credentials, give accces to***

* *BitBucket*
* *Docker*
* *JFrog*
* *Sonar token*

***Jenkins CD :***

**Continuous Delivery (CD)**

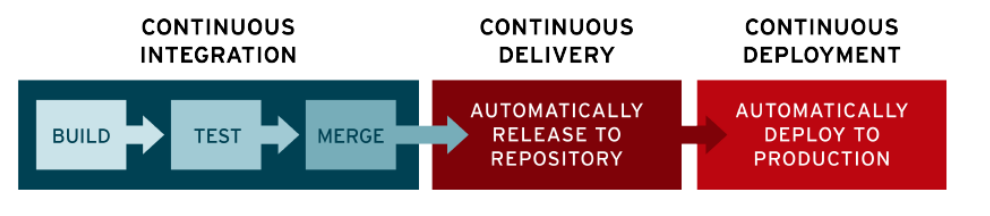
*Continuous Delivery (CD) refers to the set of practices that Development and Operations teams use to produce and deliver software for business teams*

*Prior to CI/CD, software releases are often held up by the unpredictable nature of developing complex changes, bugs/defects during testing, poor code quality, and by processes originally designed for ancient, mainframe-based software systems.*

*Continuous Delivery pipelines improve this process from start to finish. Despite being released at a higher velocity, applications can also be built and deployed to meet business needs with greater ease than in the past*

**Continuous Deployment**

*Automatically Deploy to production*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The End \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*