**Before Devops :**developer and operational engineer working independently without knowing what each other do .

* Developer will be writing the source code of the product and operational engineer will process the source code and deploy it to test envi. And make sure that the application is running. And later deploy it to the production.
* Since there is no collaboration between developer and operation team , if there are issues in processing the source code , it takes a while to figure out the problem and connect with a developer and it delays the product delivery .
* **Devops** process is introduced to overcome these challenges and fasten the software delivery process.
* Devops Engineer is the combined role of developer and operational engineer
* Devops makes life easier as it supports complete automation solution for application deployments
* by integrating various tools .
* These tools help us to process the source code towards the production deployments.
* We need to understand that Devops is not a tool, Devops is a process /culture that the engineer has to follow to deploy the source code into the production env. And operate it normally .

The source code has to go through different stages to make it production ready .

**WHAT IS DEVOPS?**

Devops is a set of practices that combines software development and IT operations.

**DEVOPS PRINCIPELS:** Devops process primarily works on 3 principals

1. Communication

2. Collaboration

3. Integration

Devops provides communication between developer and operation team and it also provides continous collaboration between teams to fasten the software delivery process & also Devops provides pipeline integration between the tools that are required to process the source code for the production deployments

**Devops Architecture**

**Git-** As local version control system

**Github-** As a Distributed Version Control System .

**Maven –** As a Build Automation tool ,to Build the Java Project.

**Docker-** To create and manage the containers

**Tomcat –** As an Application server to deploy the java applications

**Kubernetes-** As a container Orchestration Tool to manage the deployment of containers as a pods.

**SONARQUBE:**  for code analysis

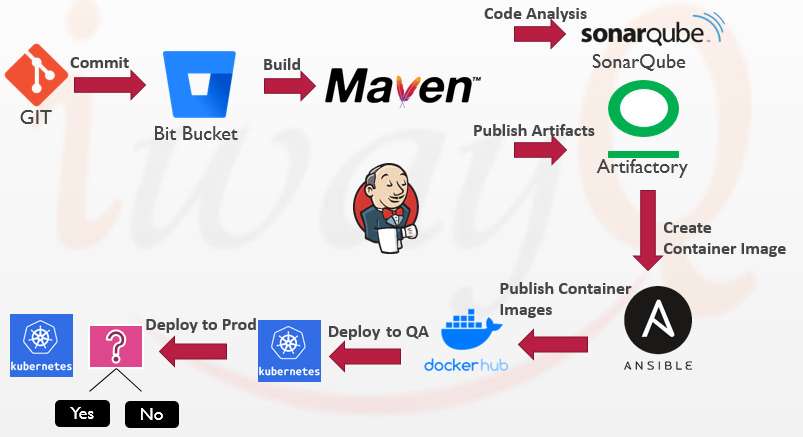
**Jfrog:**  as an artifactory management to store the artefacts that are generated by the maven build process

**ANSIBLE:**  As a configuration management and deployment automation tool

These tools can be run independently to fulfil certain requirements of the application and deployment process .However these tools must be integrated to have a complete automated solution for application deployments. So we use Jenkins as a continuous integration and delivery tool which makes the integration process easier to integrate all these Devops tools

**Jenkins :**As a continuous integration and delivery tool.

We have manual and Automation Deployements



**MANUAL VS AUTOMATION DEPLOYMENTS**

1.Manual deployments

2.Automated Deployments using Jenkins CI/CD Pipeline Code

**Pre-requsites for Devops**

**LINUX BASICS**

**User management-** How to create a user,setup passwords.

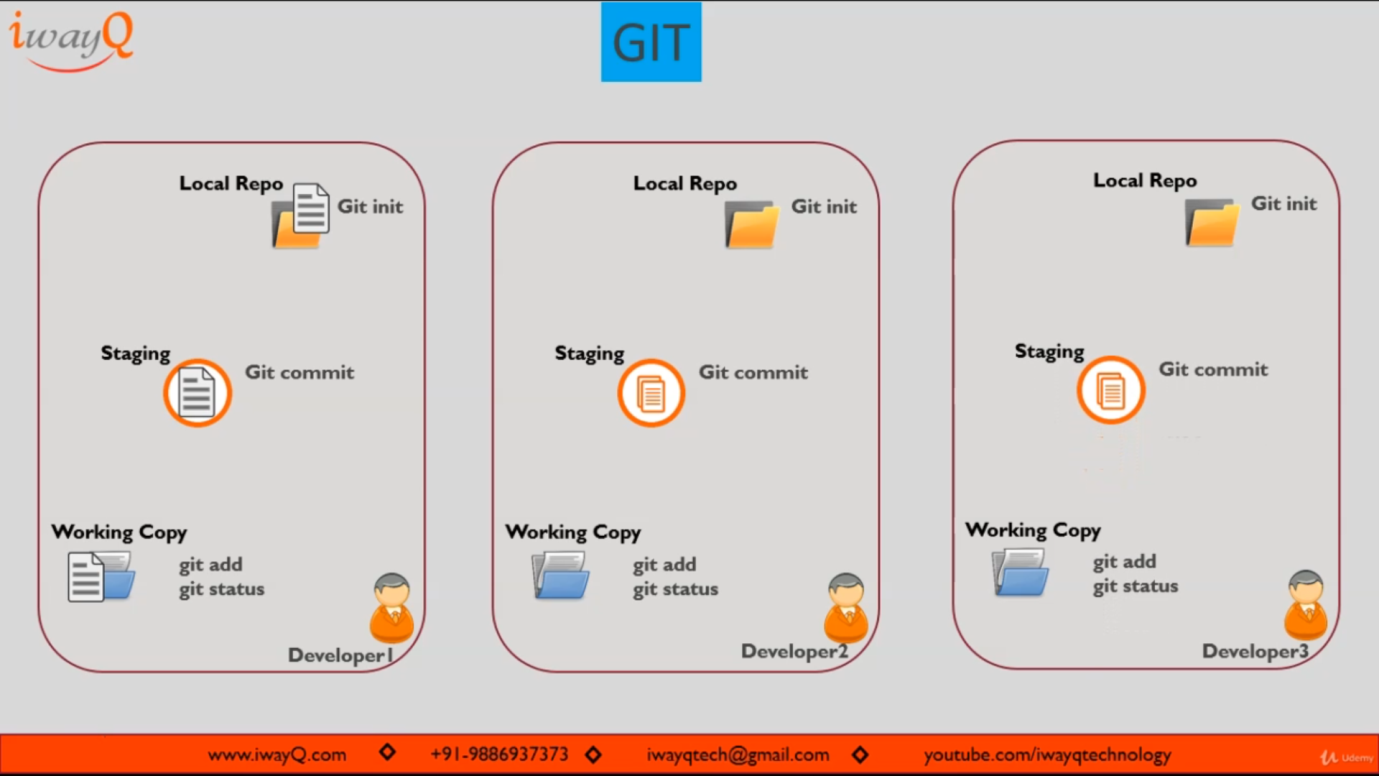
**File Management -**How to create a file,move,copy,Delete files.

**Linux file permissions-** How to set file permissions .

Linux package management – how to install/remove packages

Linux file system- how to create file system, mount, unmount File system.

**GIT**

****

* Git is a Local version control system , Developer has to follow some best practises to ensure the integrity of the source code .
* In certain times the current version of the code is not working , then has to revert back to the previous state .
* Also it is best practise to keep track of changes happening to the project, And How many times the code has modified and when it got modified..
* We are using git as a local version control ro keep track the project changes
* Example:
* Lets say Multiple developers are working on the same product, each developer has their own envi. Where they can develop the project.
* Each developer is working on their own development system
* Developer 1 , will be creating a directory where they can start writing the code that directory is called **WORKING DIRECTORY.**
* And the developer1 is keep on adding multiple core files to the same project dir.
* Same way dev2 and dev3 created working dir and started writing the code .
* To keep track of the changes happening to the project code in the working dir we use local versioning tool **GIT.**
* That means developer has to create the local repo where it can keep track of project changes.
* Whatever the developer making changes to the working copy those changes will be tracked and those changes will be stored into the local repo .
* Since loal repo is holding all the changes with respective to the timestamps ,it is possible to restore to the previous version ,it required.
* To get started creating a local repo using git we need to use **git init** cmd.
* **git init cmd-** which initialize the directory to be used as a local repo.
* Then whatever changes made to the working copy the developer has to run **git add cmd**
* **git add cmd:**  To add the changes to the staging area .
* then git commit cmd to commit those changes to the local repo.
* **What does that mean?**
* Git is a vcs to keep track of the changes that are happened to the working dir,however git not track changes automatically
* Its developer responsibility to ensure that whenever there are some changes happen to the working copy those changes must be reported to the git to store changes to the git local repo by using git add and git commit commands.
* We can revert back the changes to previous state from local repo if required .
* Git is a local vcs, the local repo which is created by the devloper1 will hold only the changes made by the devlpr1 on their working copy .

**Git Installation/lab in Linux ec2 instance**

yum install git -y

git - -version

for the above scenario ..to initialize git , first we need to create working copy using

mkdir devops-project

cd devops-project

now devops-project directory is called as an working copy .

to make this as git local repo we use

git init

ls –a now git created a .git folder (which is the local repo) where it will keep all the project related changes in .git folder.

Cmds to configure local git repo

Git config – is the command is used to configure the developer name and email id to be used for every git commit that is happening to the git local repo.

git config - - globa user.name “murali90”

git config - - global user.email [muralidevops90@gmail.com](mailto:muralidevops90@gmail.com)

Project folder:

git clone https://bitbucket.org/iwayqtech/devops-pipeline-project.git

and edit some files

git status cmd – provides the comparison between the local repo and working copy

git treats the changed files as untracked files (means changes never been updated to the local repo).

Hence we need to commit the changes to the local repo

git add . makes untracked files (shows in red colour) to tracked files(git will show tracked files in green colour)

git status

now the changes are added to staged area by using git add cmd ..and now changes are waiting to be committed to local repo

git commit -m “commit message”

while committing the changes its best practise to create a commit message why we are committing these changes to the repo

git status – now its says that nothing to commit ,working tree is clean(it says that the working copy status and local repo status are in sync).

git log cmd shows the commit id (shows the developer who committed this change with name and at what time the commit is made and the commit message)and how many times changes are committed to this repo.

To revert:

Make some changes to a file

git status

git add .

git status

git commit –m “commit message”

so now I want to revert back to previous state, now take the commit id to which you want to go and use **git reset cmd.**

git reset commit id

git log now it shows that only two commit ids

**GITHUB/BITBUCKET AS DISTRIBUTED VERSION CONTROL SYSTEM**

As above example 2 developers working individually on same project but in individual machines, now changes made by the devloper1 are not visible to the dev2 vice versa.

At certain time they need to collaborate ,to know what each other doing ,without collaboration it may lead to code integrity issues, when we merge the code base of dev1 and dev2 .

To avoid this what dev1 changing must be known by dev2 vice versa, to solve this we can use a Distributed Repository.

**Distributed Repository:** is a repo to keep our code base/source code .

Github and Bit bucket are one of the distributed repo web hosters.

The developers who are on the same project, who are pushing the code to the same repo are now treated as contributors.

Dev2 wants to know what are the changes made by dev1 , dev2 will just pull the changes from the distributed repo by using git pull cmd

He will review the changes made by dev1 and pushes the changes made by him to remote repo by using git push .

Now dev1 can pull the changes made by the dev2 .

It is best practise to pull the code before pushing , in order to avoid the merge conflicts.

git remote -v cmd shows the remote repo urL to pull/fetch source code

**Pull request**

dev2 made some changes to the code and pushed to remote repo, this changes should not cause any integrity issues ,so a prior review should be needed .

so whenever developers push the changes to the remote repo , those changes should not be merged automatically , and that requires some peer review or the approvals.

in order to avoid code integrity ,dev2 creates a new branch and starts working in created new branch

git checkout -b feature

Now performs changes and adds it to stage and later to local repo

git add .

git commit –m “changes made”

git status

Working tree is clean: means working dir and local repo are in sync

git push origin feature -- changes are pushed to feature branch of remote repo.

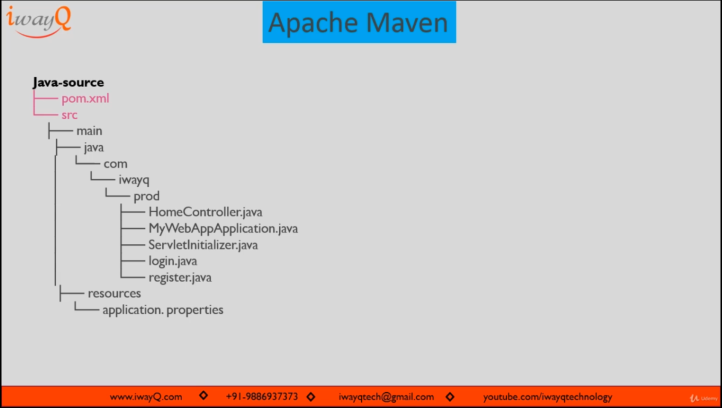
Now to merge the feature branch with master branch, dev2 must raise a pull request for approval and add reviewers in the pull request section

Now reviewer see’s what are all changes made to the code and once he is ok with the changes he will approve the pull request. Once the approvals are granted then we merge feature with master branch .(so now feature branch changes will be merged with master branch ,so now master branch will have latest healthy code written by multiple developers)

**APACHE-MAVEN(Build-Tool)**

Developers Pushed the code to remote repo, now we need to Compile ,Build,Test&Depoly it to test env.

So we are using maven as a build tool for java application, so maven expects to be in a specified folder structure as fallows



So maven know how to build this java source code and resolve the dependencies ,pom.xml is a project definition file that delivers all the required details to the apache maven, how the project source code will be build in terms of artifact names, the version compatibility ,and the dependencies that are required to download from various repository sources.

Now we submit the java code to maven for build process, and will process the code and creates the output as artefacts’ (jar,war,ear).

Those artefacts’ are deployed in further stages.

**APACHE-MAVEN-LAB**

amazon-linux-extras install java-openjdk11 –y

yum install git –y

java –version

download apache maven from <https://maven.apache.org/download.cgi>

Binary tar.gz archive [apache-maven-3.6.3-bin.tar.gz](https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz)

Download it to /opt directory

wget <https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

tar -xvzf apache-maven-3.6.3-bin.tar.gz

**SONAR-QUBE**

Now code is built, but code may have some repetitive lines or vulnerabilities, code smells, we can avoid duplicates,

So we use sonarqube as static code analysis reporter

Once the build is successful , maven will push the code to the sonar scanner for static code analysis .

Now sonarqube scans the code and the analysis reports will be stored in a database.

Sonarqube will integrated with postgres sql/oracle db for backend database to store this analysis reports

**Sonarqube-lab**

Install and configure postgres sql

Install and configure sonarqube scanner

Sonarqube expects 2 core processer and 4gb ram, so choose t2.medium

In security groups select postgres sql –port no5432

Custoum tcp ,sonarqube default port no 9000

http port no 80 and anywhere for all .

sonarqube installation prerequisites—java-openjdk 11

amazon-linux-extras list

amazon-linux-extras install postressql11 –y

now install postresql11 server package

yum install postgresl-server postgresl-devel postfressql -y

now initialize the postgresql db by this cmd

/usr/bin/postgresql-setup --initdb

Now start postgresql

systemctl start postgresql

postgres automatically creates a user in db called postgres, we can check it in tail -1 /etc/passwd

how to connect to postgresql ?

su – postgres

and now run psql ,now we are successfully connected to db

now create a user and granting the privilege for access the db, sonarqube will connect to the db and publish the analysis reports .

CREATE USER sonar WITH PASSWORD ‘sonar’ ;

ALTER USER sonar WITH SUPERUSER –cmd to to give superuser privilege

To check weather the user is created is not use cmd \du

\? For help

\q to quit

exit

clear

vi /var/lib/pgsql/data/pg\_hba.conf –edit this file to provide access for posting the reports

inside this file go to METHOD for first peer and ident and change it to trust

:wq!

Now create a db to store the anlaysis reports for sonarqube scanner

su postgres

psql

CREATE DATABASE sonar;

TO CHECK \l

Now grant the access for sonar user ON THIS SOANR DB

GRANT ALL PREVILAGES ON DATABASE sonar TO sonar;

\l to list databases

Now download sonarqube community edition

cd /opt

wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.6.1.40680.zip>

tar –xvzf sonaqube

after done with all configuration login to sonarqube

un:admin

password:admin

create a new project 🡪project name-iway🡪Generate a Token🡪copy the token (this token is required to integrate sonarqube with maven)🡪 specify the programming language which your project is using to do the code analysis 🡪choose buid technology(maven)🡪 sonarqube auto generated the command which we need to run at maven level to publish the source code to sonarqube for code analysis.

What we need to look in sonarqube analasyis is ?

Wheather the code is passed Quality Gate are not.

Quality Gate: is a gate that we define threshold at which parameter/measurement that sonarqube can consider that codecan be passed .

We can define our custom quality gates at Quality gates section in sonarqube

And we can specify the conditions at which scenario the code can be passed the quality gate by mentioning the threshold for each metric

**ARTIFACTORY**

Once the artifacts are generated those artifacts will be deployed into the application servers, end-user can start accessing the applications.

Every time there is a bug fix/ enhancing certain features of the application , it requires code need to be changed and code will be processed for build process and new artifacts will be generated and redeployed into the application servers, that means how many times code updated that many times new artifacts will be generated and the new war file will be deployed into the application servers .

Mean is that the artifacts are also be version controlled ,Those artifacts should be stored in a separate repository ,that repo is call as an artifactory .

Example of artifactory’s are JFROG artifactory, sonatype nexus artifactory.

Nexus acts as a artifactory to keep track of all the different versions of artifacts in a version control system , if we need to deploy the artifacts to the application servers ,we can get the latest artifact from nexus.

We can rollback by fetching the previous version of artifact and deploying it into the application servers.

Nexus also uses database to store the artifacts pushed by maven ,now nexus is called as an artifacts version system

**ARTIFACTORY-LAB**

Artifactory default port number :8081,8082 to acces the dashboard

Prerquisites: java jdk

Amazon-linux-extras java-openjdk11

Snapshot vs Release artifact

Snapshot: The Artifact is ready for deploying into the non-production environment.

Release: Release means the artifact is ready for deploying into the production environment.

Artifactory provides a specific unique url to fetch the artefacts .

To deploy the artifacts to nexus , we need to integrate maven and nexus by giving the configuration of nexus like url,username,password under userprofile 🡪.m2 directory🡪settings.xml

**APPACHE-TOMCAT**

Tomcat is a popular open source application server that can run java based applications, can be run as standalone server or we can run as a docker container.

Default port number is 8080

We can change the port number under tomcat/conf/server.xml

vi server.xml

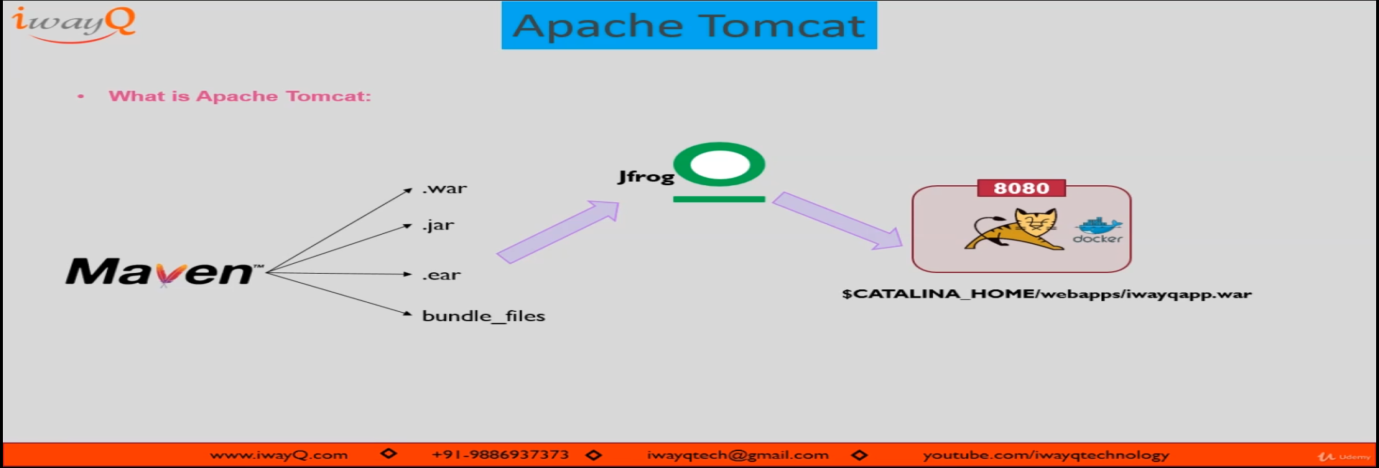
go to default connector and change the port number.

Then restart tomcat by sopping it and starting it at /bin/stop.sh

tomcat/bin/start.sh

and see the log files go to /tomcat/logs/catalinas.log.date…

artifcats will be fetched from artifactory and deployed into /tomcat/webapps folder and restart the tomcat to make the deployed application up



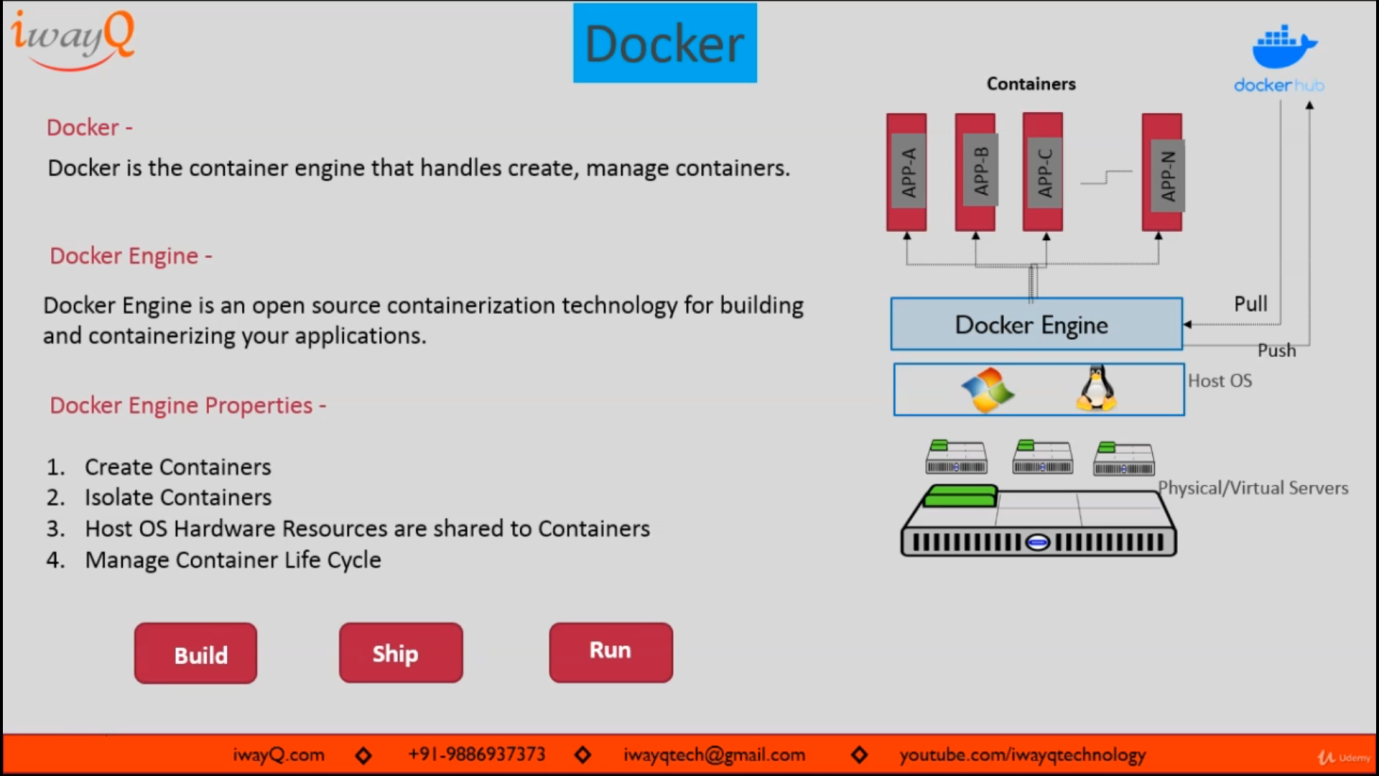
The infrastructure can be improved by deploying a proxy or a load balancer between the client and the apache tomcat application server to route the traffic to the specific port number 8080 , that way application tomcat start serving the web pages to the client

**DOCKER**

We know what is a physical server and how we can create virtual servers on top of it ,we can install operating system on top of a physical server or a virtual server which will run as a independent/isolated server for you.

Docker : when we talk about docker containers ,we won’t run applications directly on the host OS, rather we are deploying a docker engine software installed and configured on the host machine and create containers on top of it.

That means docker engine will take care of creating multiple containers n top of a single host os ,when we create multiple containers and run applications inside a container, the docker engine will takes care of how to provision underlayed hardware resources from host machine to the container application .



It is possible to run multiple containers on a single host machine ,docker engine will allocate the resources to containers from host machine.

The application A,B,C can be run as an individual independent, isolated application inside a container though all these applications run on a single host machine .

The advantage of running a application inside a container is we can have we can have multiple similar applications run at a single host os without having any conflicts in terms of the diversity configuration that it is expecting.

And all these applications can be run on a same port no. as this applications are visible within the container and all these containers are isolated and independent to run a specific single application.

Docker Image: is the once which holds the minimal os configuration and the application software and the required dependencies to run the application within the container level .

These docker images are hosted in a registry called DockeHub.

We can pull the image from the dockerhub and run as a container.

**Advantage of docker:**

We can create custom docker image using Dockerfile and can push/ship it to the dockerhub and can we run it wherever we need it and can run the application and it solves the infrastructure issue.

**Building custom image using dockerfile**

Create instance and in security group allow custom tcp allow port number 8080

yum install docker –y

docker - -version

systemctl start docker

systemctl enable docker

system status docker

docker images- shows the images in the machine

to create custom docker image

vi Dockerfile

FROM tomcat:latest

MAINTAINER “PR –Reddy - iwayQ”

COPY ./iwayq.war /usr/local/tomcat/webapps

:wq

docker build . - - tag iwayq-tomcat

docker build commands builds the custom image by running the Dockerfile

to check the created image

docker images

to run a container from the custom image

docker run –d - - name iwaq-tomcat –p 8080:8080 iway-tomcat

docker ps cmd to see the running containers

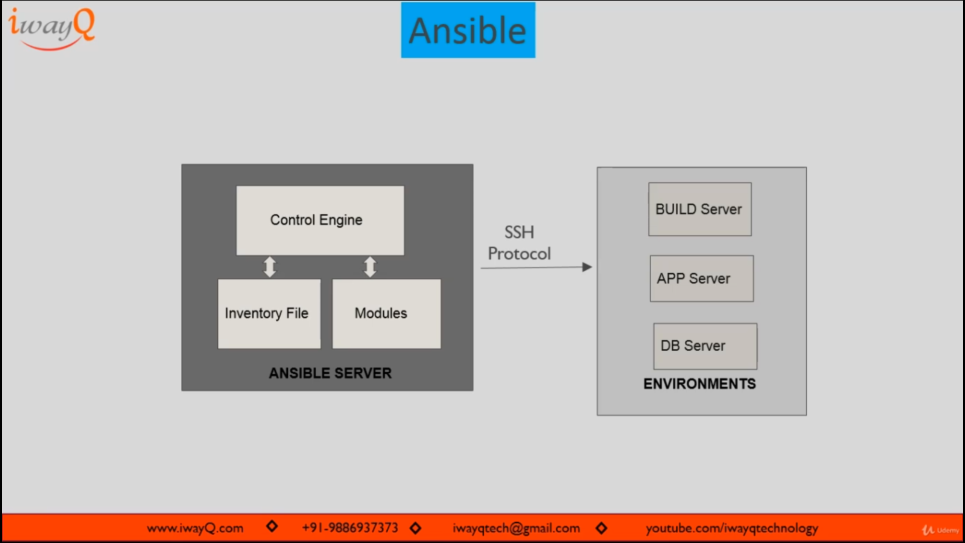
COMMAND SECTION of docker ps tells the command that runs when u start the container to start the application.

**ANSIBLE**

We are using Ansible as a configuration automation tool in this project, we are using Ansible to create docker images automatically and push the custom images to dockerhub registry .

Ansible can be used as a configuration automation tool or can be also used as a software automation tool .

Eg: lets say we have number of ec2 instances in our environment which can be labelled as build server, app server, db server inorder to perform configuration changes in on all these instances that’s tricky and tome consuming to perform the configuration changes manually logging into each server, performing series of tasks manually hence Ansible solves the problem to have configuration automation deployments in all the instances at one go, to achieve configuration automation, we need to have Ansible server where we deploy Ansible server packages and it installs the Ansible control engine to takes care of configuration automation.



**INVENTORY FILE:**  inventory file will have list of clients where configuration has to be deployed

**Modules:**  Modules will tell what needs to be deployed

\*\*\* Ansible is a agent less automation tool that means we need to install Ansible server package on the server side and there is agent needed to be installed and configures at the client side ,Ansible server connects to the clients using ssh protocol , since every client is predefined with the ssh protocol to have admin logins, we need not to have a custom configuration that is required to have client side to have a communication from the server side to client .

Password less authentication will be set using ssh key pairs between Ansible server and client .

Ansible Playbook written in yaml which is the set of instructions to be deployed on the client side.

Ansible playbooks can be run on remote machines or can be run on Ansible server itself .

We are using Ansible playbooks to build a custom docker images and push the image to docker hub registry .

**ANSIBLE-LAB**

How to run a Ansible playbook to create custom docker images automatically by pulling the nexus artifactory.

Create ec2 instance and install Ansible to make it as Ansible server

amazon-linux-extras install ansible2 –y ---this cmd will vary based on the linux os distribution this is for amazon linux ami.

ansible - - version

ansible requires some python modules for creating custom docker images.

Install dependicies

yum install python-pip –y -----it will install pip package

pip install docker-py –this module required to deal with docker to create custom docker images using ansible playbook .

**now write the playbook :** playbook file name create-container.yml

#Create Custom Container Image

#Push to DockerHub Registry

---

- hosts: localhost

tasks:

- name: Pull .war artifact from Artifactory

become: true

get\_url:

url: http://3.84.217.9:8082/artifactory/libs-release/com/iwayq/iwayQApp/1.0-RELEASE/iwayQApp-1.0-RELEASE.war

dest: /home/ec2-user

url\_username: admin

url\_password: Admin123

- name: Rename .war Artifact

command: mv iwayQApp-1.0-RELEASE.war iwayq.war

- name: Pull Tomcat Container Image

docker\_image:

name: tomcat:latest

source: pull

- name: Build image and with build args

docker\_image:

name: iwayq

build:

path: /home/ec2-user

pull: no

args:

listen\_port: 8080

source: build

- name: Log into DockerHub

docker\_login:

username: iwayqdockertest

password: P2$$w0rd

- name: Push to docker hub

docker\_image:

name: iwayq

repository: iwayqdockertest/iwayq:1.0

push: yes

source: local

In the Ansible server perform this steps

Mkdir /dockercode

Vi /dockercode/Dockerfile

FROM tomcat:latest

MAINTAINER “PR –Reddy - iwayQ”

COPY /artifactory/iwayq.war /usr/local/tomcat/webapps

:wq

Vi play.yaml

Mkdir /artifacts

And now run the ansib-le playbook using the following command

ansible-playbook play.yml

and now check in dockerhub that the custom image has been uploaded-or not.

**Kubernetees**

What is kubernetees?

where will we use it in Devops project ?

what happens if the docker container dies?

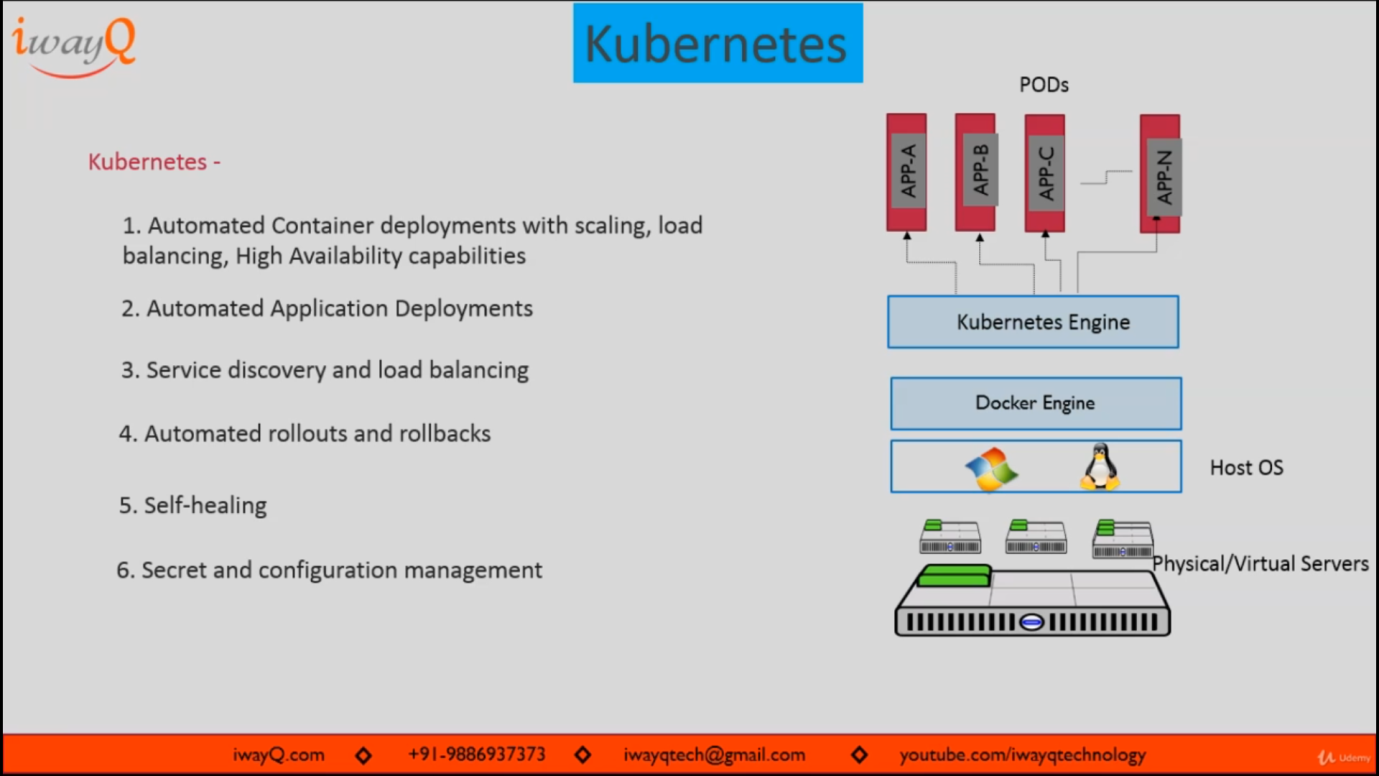
How to scale the docker containers?

How to maintain the high availability of the docker containers?

Above tasks will be solved using Container orchestration ,k8s is the container orchestration tool

What is container orchestration?

Creating a container, replacing a container, providing high availability of a containers



How k8’s work?

On top of docker engine k8s engine will be installed and configured to manage the containers.

K8s engine will act as a container orchestration tool ,it takes the responsibility of managing how to create a container, where to create a container , how to replace a container it the container dies and providing high availability, scalability features .

K8s engine as which is called as master node can deploy the containers in anyone of the worker nodes, that means the first container that running application A can be on a different working node, and second container running application B will be on different worker node, k8s engine manages the containers in terms of pod wrapper.

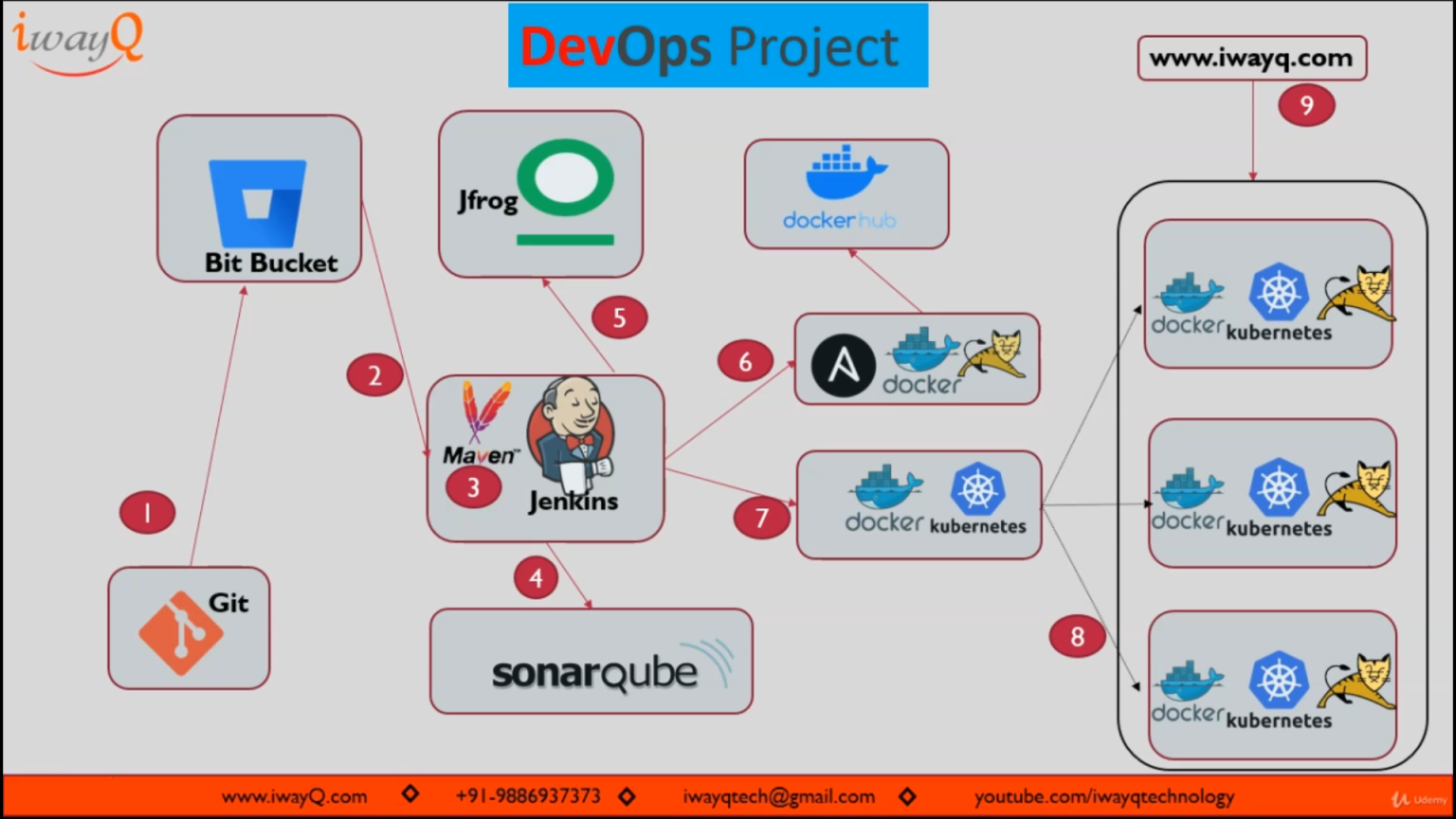
**POD:**  pod is the smallest individual compute unit within the k8s cluster.

the containers are managed within the pod

**K8s Properties:**

1. **Automated container deployment tool which provides scaling, high availability capabilities and load balancing for an application.**
2. **K8s manages application deployment automatically .**
3. It can automatically detect the traffic and performs **load balancing across the pods**
4. **Automated Rollouts and rollbacks** ,k8s is most popular for rolling update strategy, it is simple to rollback to previous versionif the current version is not woring as expected .
5. **Self healing** – means it can perform the health checks if the pod is unhealthy it can be replaced automatically .
6. It supports **secret and Configuration management, it also sup**ports to store any secrets securely .

These are the high level features that k8s is offering to manage the containers .



1. Once the is ready developers push the code to the github repository,github solves the code integrity and collaboration issues between the contributors, when multiple developers are working on same source code developers can have a collaboration github distributed repository .

Developers can raise a pull request, once the code is reviewed and got the approvals the changes will be merged with the master branch.

2&3. Once the code is ready, Jenkins will clone the code locally and Jenkins will call maven by using maven commands to perform the build process, apache maven is build automation tool use for java based applications, maven can compile build, resolve the dependencies and package the source code into a artifact, once the build is successful, Jenkins will ask maven to publish the source code to the sonarqube for code quality analysis.

4. Sonarqube will perform static code analysis and generate the reports in the dashboard, once the quality gate is passed, we will publish the artifact to the nexus repository.

5. Maven can also publish the artifacts to the sonatype nexus artifact repository manger, we are using nexus artifactory as a repository manager to keep track of the artifacts that are generated by the apache maven as they are version controlled. Once the build is successful and the code passes the quality gates at sonarqube then maven pushes/publishes the artifacts to nexus.

6. Once the artifact is ready in artifactory, we can use those artifacts to deploy into test environment using ansible playbook, the application server can be a standalone or it can be a docker container, in order to deploy the artifact into the containerized tomcat application we need to create our own custom container image.

We have used ansible playbook to perform the automation of creating a docker container image which is including our artifact .ansible playbook will pull the artifact from the sonatype nexus artifactory and will create a custom docker image. That image will be pushed back to the dockerhub registry. This custom image we can use for later deployments.

7. For deployment purpose we are using kubernetes cluster, k8s cluster will use docker engine to manage the deployment of containers, when we have the deployment and the service manifest file definitions, Jenkins will run those deployment manifest and service manifest on the k8s cluster, once we deploy the deployment and the services, the k8s cluster will start creating the application instances on the worker nodes, finally we can see our application is successfully deployed and running inside a pod which is managed by the k8s cluster, now we can access our application using specific url which is mapped to the specific endpoint.

To make this process fully automated we need to depend on continuous integration and continuous delivery tool which is Jenkins in this Devops project.

**Stages in the above diagram flow**

1. Develop the code and push to github
2. Checkout the github repository
3. Perform the maven build process
4. Publish the source code to the sonarqube for code quality analysis
5. Publish the artifact to the artifactory
6. Run the ansible playbooks to create customized container image and publish to docker hub registry
7. Call the deployment and service manifest definition’s on k8s master node
8. K8s will takes care of deploying the application into the worker node as a pods
9. End-user can starts accessing of the application

**Jenkins-setup**

Create a instance and open port 8080 in security groups

Install jdk , apache maven, git , wget, curl

Jenkins initial admin password in

/var/lib/Jenkins/secrets/initialAdminPassword

**Install following plugins**

Install these plugins for tool integration

Sonarqube scanner plugin for sonarqube

Nexus artifact plugin for sonatype nexus plugin

Ssh agent plugin for running the ansible playbooks

**And now go to Global tool configuration and configure java and maven path .**

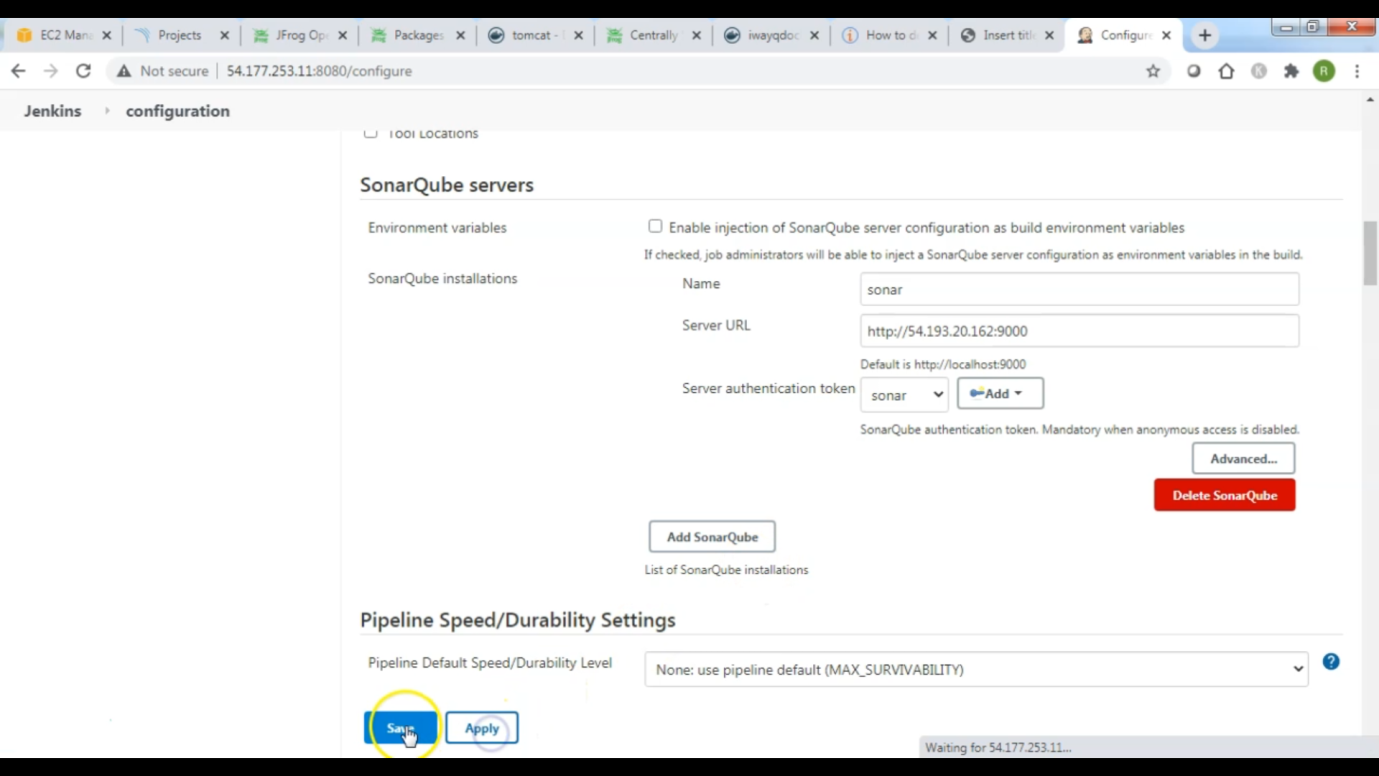
**And now to go Global credentials**

Add token generated while creating the project in sonarqube dashboard and in manage credentials section🡪 global credentials🡪add credentials🡪secret text🡪paste the token🡪Id sonar🡪ok

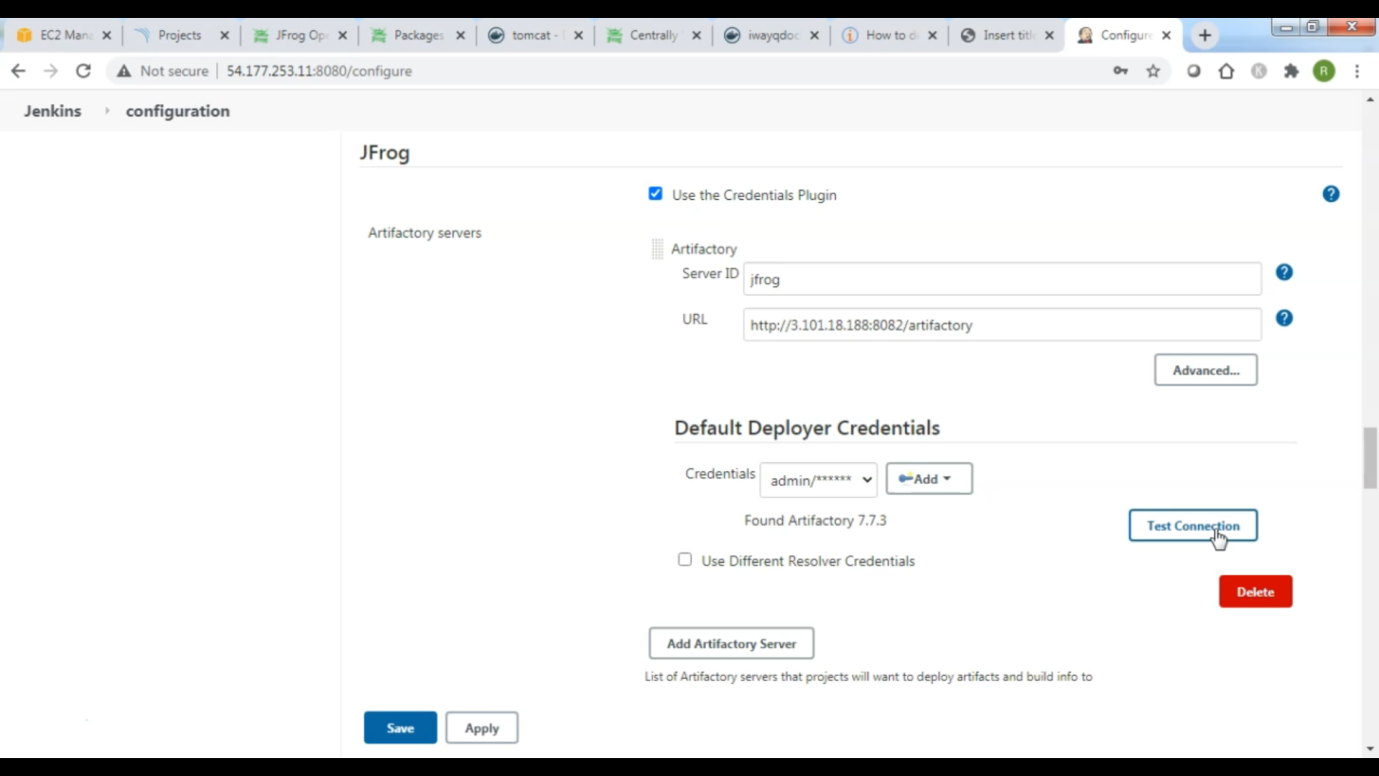
**And configure nexus credentials**

manage credentials section🡪 global credentials🡪add credentials🡪username and password 🡪username: admin🡪password:admin123🡪Id nexus🡪ok

**Manage jenkins -->configure system🡪 sonarqube configuration**

****

**Manage jenkins -->configure system🡪 nexus configuration**



Test connection

**Ssh agent plugin configuration:** Now add pem key file in global credentials to authenticating with the ansible server when we are running the ansible playbooks from Jenkins machine .

**Creation ssh key pair in Jenkins server:**

Ssh-keygen –t rsa (rsa is the type of the key pair to generate)

And give path as /var/lib/Jenkins/.ssh/id\_rsa 🡪once the keys are created

cat /var/lib/Jenkins/.ssh/id\_rsa

**Adding pem key file in global credentials** manage credentials section🡪 global credentials🡪add credentials🡪kind🡪ssh username with private key 🡪username: admin🡪Private key : of Jenkins server and copy paste it in key section 🡪add🡪ok

**And configure gihub credentials**

manage credentials section🡪 global credentials🡪add credentials🡪username and password 🡪username: murali90🡪password:admin123🡪Id git 🡪ok

**Copy The Public key of Jenkins and add it to ansible server and later in k8s server**

**Create a user in ansible server**

adduser admin

cd /home/admin

mkdir .ssh 🡪 cd .ssh 🡪 vi authorized\_keys and now paste the public key of Jenkins server in this file for password less authentication

chmod 600 authorized\_keys

and give sudo access to this user got get the root privilages

visudo

admin ALL=(ALL) ALL

:wq!

Now login into k8s master server and do the same

adduser admin

cd /home/admin

mkdir .ssh 🡪 cd .ssh 🡪 vi authorized\_keys and now paste the public key of Jenkins server in this file for password less authentication

chmod 600 authorized\_keys

and give sudo access to this user got get the root privileges

visudo

admin ALL=(ALL) ALL

:wq!

**Github webhook configuration :**

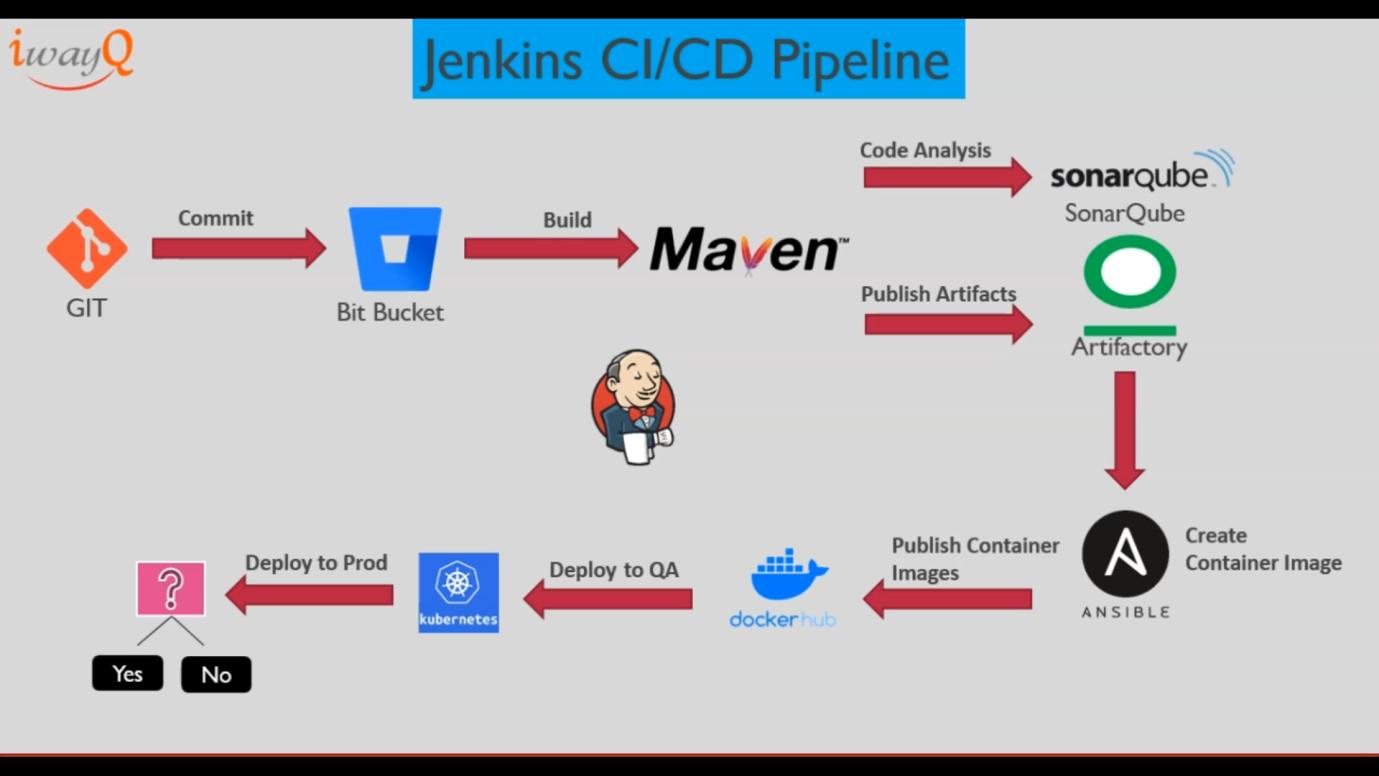
Install git plugin if not present

Then go to github settings🡪 go to github web hook section 🡪create webhook🡪 payload url :paste the jenkins url/git-webhook/

Application format: json

And select push events and save

**JENKINS CI/CD PIPELINE**



**Now write declarative pipeline in Jenkinsfile and commit it to the github repo**

pipeline {

agent any

tools {

maven 'maven'

}

stages {

stage ('Checkout SCM'){

steps {

checkout([$class: 'GitSCM', branches: [[name: '\*/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'git', url: 'https://iwayqtech@bitbucket.org/iwayqtech/devops-pipeline-project.git']]])

}

}

stage ('Build') {

steps {

dir('java-source'){

sh "mvn package"

}

}

}

stage ('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar') {

dir('java-source'){

sh 'mvn -U clean install sonar:sonar'

}

}

}

}

stage ('Artifactory configuration') {

steps {

rtServer (

id: "jfrog",

url: "http://3.84.217.9:8082/artifactory",

credentialsId: "jfrog"

)

rtMavenDeployer (

id: "MAVEN\_DEPLOYER",

serverId: "jfrog",

releaseRepo: "libs-release",

snapshotRepo: "libs-snapshot"

)

rtMavenResolver (

id: "MAVEN\_RESOLVER",

serverId: "jfrog",

releaseRepo: "libs-release",

snapshotRepo: "libs-snapshot"

)

}

}

stage ('Deploy Artifacts') {

steps {

rtMavenRun (

tool: "maven", // Tool name from Jenkins configuration

pom: 'java-source/pom.xml',

goals: 'clean install',

deployerId: "MAVEN\_DEPLOYER",

resolverId: "MAVEN\_RESOLVER"

)

}

}

stage ('Publish build info') {

steps {

rtPublishBuildInfo (

serverId: "jfrog"

)

}

}

stage('Copy Dockerfile & Playbook to Ansible Server') {

steps {

sshagent(['sshkey']) {

sh "scp -o StrictHostKeyChecking=no Dockerfile ec2-user@34.229.61.202:/home/ec2-user"

sh "scp -o StrictHostKeyChecking=no create-container-image.yaml ec2-user@34.229.61.202:/home/ec2-user"

}

}

}

stage('Build Container Image') {

steps {

sshagent(['sshkey']) {

sh "ssh -o StrictHostKeyChecking=no ec2-user@34.229.61.202 -C \"sudo ansible-playbook create-container-image.yaml\""

}

}

}

stage('Copy Deployment & Service Defination to K8s Master') {

steps {

sshagent(['sshkey']) {

sh "scp -o StrictHostKeyChecking=no create-k8s-deployment.yaml ec2-user@54.162.42.75:/home/ec2-user"

sh "scp -o StrictHostKeyChecking=no nodePort.yaml ec2-user@54.162.42.75:/home/ec2-user"

}

}

}

stage('Waiting for Approvals') {

steps{

input('Test Completed ? Please provide Approvals for Prod Release ?')

}

}

stage('Deploy Artifacts to Production') {

steps {

sshagent(['sshkey']) {

sh "ssh -o StrictHostKeyChecking=no ec2-user@54.162.42.75 -C \"sudo kubectl apply -f create-k8s-deployment.yaml\""

sh "ssh -o StrictHostKeyChecking=no ec2-user@54.162.42.75 -C \"sudo kubectl apply -f nodePort.yaml\""

}

}

}

}

}