Deploy a WAR on Docker Container

CI CD Of Docker Container

**NOTE: MAKE SURE WE INSTALL DOCKER TO ALL CLIENTS AND HOST MACHINES AND START DOCKER SERVICE**

|  |  |
| --- | --- |
| Create a new pipeline-based job | Docker\_War  Go to New Job Select Pipeline Based Job  Click OK |
| Write a pipeline script | **We can generate a pipeline syntax automatically by clicking on Pipeline Syntax (Pipeline Syntax Generator) at the bottom of a Script text area.**   1. **1st Stage: SCM Checkout (Git Checkout)**  * To get the code from the Git:   + Select “git: Git”   + Repository URL: Give Git Repo URL Which Has Java Based Web Application   + Branch: Give branch name if the code is in different branch   + Credentials: If the repository is a private than we need to give a git credentials * For that, Kind: Username with Password, Username: Git Username, Password: Git Password, * ID: Give a Unique ID, * Description: Anything * Press Add * Once we created credentials, we Just need to select the name * To generate a pipeline script, just click on Generate Pipeline Script * After we get a script, we need to add that script to the one of stage of script   A picture containing screenshot  Description automatically generated   1. **2nd Stage: MVN Package:**  * We use Maven to compile unit test and create a WAR file. * That WAR file we include as part of Docker Image   **NOTE: TO GENERATE A SCRIPT FOR MAVEN THERE IS NOT DIRECT STEP CALLED AS MAVEN SO, WE NEED TO USE “sh: Shell Script”**   * First select “sh: Shell Script” * Write a maven command which we want to execute into the text area   + mvn clean package * Click on generate a script   + Add that command under 2nd stage   + sh label: '', script: 'mvn clean package'   **Final Code for this stage:**  stage('Mvn Package'){  sh label: '', script: 'mvn clean package'  }  **IF THIS COMMAND GIVE ERROR REGARDING MAVEN THAN FOLLOW THESE STEPS:**  **Go to Pipeline Syntax Generator**  **Simple Step: Chose “tool: USE a tool from a predefined Tool installation”**  **Tool Type: Select Maven**  **Tool: Select maven variable name which we defined in “Global Tool Configuration” Ex. maven3**  **Click on Generate Pipeline Script**  **tool name: ‘maven3’, type: ‘maven’**  **Into the script we need to write following command:**  **stage(‘Mvn Package’){**  **def mvnHome = tool name: ‘maven3’, type: ‘maven’**  **def mvnCMD = “${mvnHome}/bin/mvn”**  **sh “${mvnCMD} clean package”**  **}**  **NOTE: WHENEVER WE USE INTERPOLATION (${…}) WE NEED TO PLACE IT BETWEEN *DOUBLEE QUOTS* “”**   1. **3rd Stage: Build a Docker Image:**  * We need to use following command under the “Build Docker Image” stage:   + sh ‘docker build -t DockerUserName/ImageName:VersionNumber LocationOf A DockerFile’   + sh ‘docker build -t harshlad726/my-app:1.0.0 .’     - t= tagging     - . is a location of a docker file (if it is in a some folder we need to give a full path of a docker file)   **NOTE:**  **WHENEVER WE ARE GOING TO PUSH THE DOCKER IMAGE, WE MUST BE TAGGED WITH DOCKER HUB USERNAME**  **DOCKER IS INSTALLED ON JENKINS**  **JENKINS HAS A PERMISSION TO CALL DOCKER COMMANDS**  **TO GIVE A PERMISSION TO CALL A DOCKER COMMANDS INTO A JENKINS WE NEED TO RUN FOLLOWING COMMANDS TO THE TERMINAL WHERE OUR JENKINS AND DOCKER IS INSTALLED:**  **sudo usermod -a -G docker jenkins**  **sudo chmod 664 /var/run/docker.sock**  **sudo service jenkins restart OR systemctl restart jenkins.service**   1. **4th Stage: Push Image to Docker Hub (‘Push Docker Image’)**  * To push image to the docker hub we need to add following command to the 4th stage: sh ‘docker push DockerHubUserName/Image:VersionNumber’   **Ex. sh ‘docker push harshlad726/my-app:1.0.0’**  **NOTE: BEFORE WE PUSH THE IMAGE, WE MUST HAVE AN ACCOUNT ON DOCKER HUB AND WE MUST LOGIN TO THE ACCOUNT FOR THAT WE NEED TO ADD FOLLOWING COMMAND AS WELL:**   * + **sh ‘docker login -u DockerUserName -p’**     - **u=user, p=password**   + **WE CAN NOT GIVE PASSWORD DIRECTLY TO THE COMMAND (OR GIVE PLAIN TEXT) SO WE NEED TO ADD A CREDENTIALS TO THE JENKINS AND FROM THERE GET THAT OBJECT AND USE IT:**     - **FOR THAT FOLLOW THESE STEP:**       * **GO TO THE PIPLINE SCRIPT GENERATOR**       * **SELECT “withCredentials: Bind credentials to variables”**       * **Press “Add” and Select “Secret Text”**       * **Variable: AsYourChoice**       * **Credentials: Press “Add” and Select “Jenkins”**       * **Kind: Secret Text**       * **Secret: Give Docker Hub Password**       * **ID: AsYourChoice**       * **Description: AsYourChoice**       * **Click on Add and Generate Pipeline Script**       * **Add that into this 4th stage**   **withCredentials([string(credentialsId: 'Docker\_Pwd', variable: 'Docker\_Pwd')]) {**  **sh “docker login -u harshlad726 -p ${Docker\_Pwd }”**  **}**  **So, the final script for this stage will be,**  **stage('Push Docker Image'){**  **withCredentials([string(credentialsId: 'docker-pwd', variable: 'Docker\_Pwd')]) {**  **sh "docker login -u harshlad726 -p ${Docker\_Pwd}"**  **}**  **sh 'docker push harshlad726/my-app:1.0.0'**  **}**  **A screenshot of a cell phone  Description automatically generated**   1. **5th Stage: Deploy this image (Run container on Dev Server)**   **It means we want a run commands on remote machine not on jenkins server**   * **To run docker image on dev server, we need to run following command:**   + sh ‘docker run -p 8080:8080 -d -name DockerImageName DockerUserName/ImageName:TagName(VersionNumber)’   + p= Port mapping   + d= detach server/backend server   + If we don’t give a name, it will take name randomly   **To run this command on the remote machine, we need to give ssh option and to use ssh option we need to use sshAgent plugin**  **We can generate pipeline script using script generator:**   * **Sample Step: select “sshagent: SSH Agent” (IF THIS OPTION DOESN’T APPEAR, THEN WE NEED TO INSTALL “SSH AGENT PLUGIN”)** * **Click on “ADD” select “Jenkins”** * **Kind: select “SSH Username with private key”** * **Username: Username of user of remote server or username of another instance username (ec2-user or whichever user we added to that user[ans\_admin])** * **Private key: select “Enter directly”**   + **Key: pest private key of HOST**   + **ID: give any ID**   + **Description: give anything and click on “Add” button**   + **Now, select the “dev-server2” which we just created and press “Generate Pipeline Script”**   **The final script for this step will be,**  stage('Run Container on Dev Server'){  def dockerRun = 'docker run -p 9090:9090 -d --name my-app harshlad726/my-app:1.0.0'  sshagent(['dev-server2']) {  sh "ssh -o StrictHostKeyChecking=no ans\_admin@172.31.27.160 ${dockerRun}"  }  }  **NOTE: WE USE 9090 PORT NUMBER HERE BECAUSE 8080 PORT NUMBER IS ALLOCATED TO THE JENKINS MAKE SURE WE ADD THE OTHER PORT NUMBER IN SECURITY GROUP IN INSTANCE, SO THAT WE CAN AVOID ERROR** |
| To check the output/deployment of container | Take client public IP and take the port number which we use in script  For me, its **54.167.53.180:9090** |