

Devops Assignment

1st Use Case

- 1) build php website on github

- Step 1:- Getting the code from github
 - 1) Both Clone the github repository to local.
 - sudo git clone <copy the url from github>
 - 2) If the folder already exists, then forcefully remove the directory
 - sudo rm -rf <folder name>
 - 3) Again use the clone command

Step 2:- Setting up jenkins

- 1) If jenkins not up and running then use this command.
 - sudo systemctl start jenkins
- 2) Jenkins is up & running
password = jenkins@123

Step 3:- Setting up the slave

- 1) Go to jenkins
- 2) Select Manage jenkins
- 3) Scroll down to "Configure Global security" - select
- 4) Scroll down to "Agents" option
- 5) Select the "fixed" option and write 9000 for the port
- 6) Save
- 7) Again in manage jenkins select "Manage Nodes and Clouds"

- 8) Select + New Node (top left corner)
- 9) Give Node name = slaveone_ip address of slave server
- 10) Select permanent Agent
- 11) Save
- 12) Again select that created node and configure details
- 13) Number of executors = 1
- 14) In the slave machine create a folder jenkins.
in the slave machine run this command
• sudo mkdir jenkins
folder will be created in /home/msis
(When use 'ls' command to list the contents
folders will be displayed in blue, text files in
white)
- 15) In the master jenkins while adding the
configuration to slave node, in the Remote root
directory : /home/msis/jenkins
- 16) Usage: Use this node as much as possible
- 17) Launch method: Launch agent by connecting it to the
controller
- 18) Save
- 19) Go to 'Configure System' in manage jenkins
- 20) Go to Jenkins Location
- 21) In the Jenkins URL change the ip from localhost
to the master IP address
ex:- http://172.16.51.27:8080/
- 22) Save
- 23) Click on the created node
- 24) You will get curl and java command
- 25) Copy the curl command to slave & execute
- 26) Copy the java command use sudo and copy
the java command and re execute
- 27) Nodes will be connected to master.

Step 4:- Creating and hosting php website on slave

- 1) Select + New item
- 2) Select Retr Retrict where this project can be run
- 3) In Label expression select the slave node created
- 4) Source Code management mco option
- 5) Select Get
- 6) In the repository URL copy the url from github and paste it.
- 7) Checks if the branch in jenkins and in github both are same.

If the branch name given in github is master same has to be given for while copying the github URL. If its main then main has to be given in Jenkins

- 8) Go to Build steps
- 9) Select Execute shell option
- 10) Copy the the shell script file contents from deployish file in github and paste it
- 11) Save it
- 12) Build the project
- 13) Go to the slave machine
- 14) Try to hit localhost:80
- 15) This should open bookalbum website
- 16) bookalbum folder should be copied /var/www/html in the slave machine,

Step 5: Pushing

Step 5 :- Containerizing a php application and pushing the docker image to docker hub.

- 1) Create docker credentials
→ Go to manage Jenkins

- Go to "Manage Credentials"
 - Clicks on any "system" option
 - Click on the "Global" option
 - Clicks on "+Add Credential"
 - Kind has to be "Username with password"
 - Scope - Global (Jenkins, nodes, items, all child items)
 - Give your dockerhub username
 - Give p"your password for your dockerhub profile
 - ID - give a name to identify the credentials.
- 2) Create a freestyle object by selecting + New Item
 - 3) Restrict where this project can be run - select this option
 - 4) Write the slave node name → ubuntu_2.7-server
 - 5) Source code management - select git
 - 6) Copy the github url and paste it
 - 7) In build steps - select Docker Build and Publish
 - 8) In repository name - Give dockerhub username and a repo name that has been created by you in dockerhub. Example: nishisrao/dockerdemo → This has to be already created in dockerhub
 - 9) In Tag - you can mention version like v1
 - 10) Registry credentials : Select the dockerhub credentials created in the first step.
 - 11) Save it and build the project
 - 12) You can check in the dockerhub the a new dockerimage with the given tag will be created in your repository.

Step 6 :- If we want to pull image to slave node and run the docker image

Copy all the steps from Step 5

- 1) In build steps select execute shell, any name
- 2) sudo docker run --name demodocker -it -d -p 4012:80 nishisrao/dockerdemo:v1

The newly created image in dockerhub

- 3) In the slave machine go to `localhost:4010`
- 4) You will get bookalbem website.

Step 7: Kubernetes deployment

- 1) Go to + New item
- 2) Create a new freestyle project
- 3) Copy the get URL
- 4) In the build step select Deploy to Kubernetes
- 5) In the repo textbox provide your username
- 6) In the build step select Deploy to Kubernetes
- 7) In the kubeconfig textbox provide the kubernetes key name that you have added in the credentials
- 8) In the tc config files give `deployment.yaml`, `service.yaml` files name as comma separated values
- 9) Save it and build
- 10) Go to terminal
- 11) `ssh 172.16.51.53@`
- 12) `ssh chefserver@172.16.51.53`
 password chef@123
 ip = 172.16.51.53
 username = chefserver
- 13) `sudo kubectl get pods`
 → This will show the newly created pod for the bookalbem
- 14) `sudo kubectl get svc`
 → This will show the running services with its port number
- 15) Hit the above kubernetes node with the above shown ip. It will show the bookalbem website.
 Slave ip address = 172.16.51.50
 Username = ansible
 password = s01s@123

steps to create the credential for kubernetes

- 1) SSH to chef server @ 172.16.51.53
- 2) cd /kube
- 3) sudo vi ~/.kube/config
- 4) Copy this file content completely
- 5) Go to Jenkins
- 6) Go to Manage Jenkins
- 7) Go to credentials
- 8) In kind select: Kubernetes configuration (Kubeconfig)
- 9) In Isbconfig select enter directly
- 10) The file content copied should be pasted here
- 11) Create the key

steps to add kubernetes plugin if not present :-

- 1) Go to github.com/Nidhi-S-Rao/Plugin
- 2) Download the .hpi file
- 3) Go to Manage Jenkins
- 4) Go to manage plugin
- 5) Go to advanced
- 6) Select the downloaded .hpi file and deploy.

UseCase 2 :-

Maven Application Build, compile, install & package

Step 1:- Maven clean compile
Get repository :- MyMavenApp

- 1) Create a freestyle project with name "maven_compile"
- 2) Source code management

Get : copy the URL from github maven repo

- 3) Build steps :-
Select → Invoke top-level maven project target
If this option is not available we have to add the plugin.

- To add the plugin

→ Go to Manage Jenkins

→ Go to Manage Plugins

→ Go to available plugins & select Maven Integration plugin and ~~install~~ without restart

- 4) Once we select "Invoke top-level maven target" we have to add the maven version

- To add the maven version

→ Go to Manage Jenkins

→ Go to Manage Credentials

→ Go to System

→ Select the Global Credential

→ Add C

→ Go to Global Tool Configuration

→ Scroll down to Maven

→ Select Maven installations

→ Give name to the maven version

→ Save

In Maven Version select the maven name that you gave in step 4.

In Goals give clean compl compile

6) Post Build Actions

→ Build other projects

→ projects to build :- give next project name which you want to run after the successful completion of maven compile demo.

7) Save

Step 2: Maven test demo

1) Create a freestyle project with name "maven-test-demo"

2) Source code management

git : copy the URL from github mymavenapp

3) Build steps

- Invoke top-level Maven targets

Maven version: maven

Goals : validate test

4) Post build actions

- prebuild HTML reports

- Reports → HTML directory to archive → in this give target

- Index page = *.xml

- Report title = HTML report

5) In Post build actions

- prebuild JUnit test result report

- Test report XMLs

target / s target / surefire - reports / *.xml .

Save

6) Step 3 :- Maven install package

- 1 & 2 steps are same as before

3) Select build steps - In that select invoke top-level Maven targets

4) Select Maven & in goals = install package

5) Build step select execute shell

Give this command

→ cd target

→ java -jar my-app-1.0-SNAPSHOT.jar

How to create a CI/CD pipeline :-

1. Install pipeline plugin

→ Go to manage Jenkins

→ Go to manage plugins

→ In available plugins search build pipeline & install.

2. In Dashboard click next to All

3. Select build pipeline view

4. In the upstream/downstream config project that initiates the pipeline select the first one it is maven-compile-demo.

UseCase 3Simple - Java - Maven - AppStep 1:- Simple Java clean compile :-

- 1) Create a freestyle project simple java clean
- 2) Copy the getURL <simple-java-maven-app>
- 3) Build steps - select Invoke top level maven target
- 4) Select the maven version
- 5) Add Goals - clean compile

Step 2:- Simple Java code review :-

- 1) Create the freestyle project
- 2) Copy the get URL.
- 3) Build steps - select Invoke top level maven target
- 4) Select maven version.
- 5) Add goals = -P metrics pmd:pmd checkstyle:checkstyle findbugs:findbugs
- 6) In post build action select "Record compiler warnings and static analysis results"
Tool → select checkstyle
Add tool → select findBugs
Add tool → select pmd PMD
- 7) Save

Step 3:- Simple Java Test :-

- 1) Create a freestyle project
- 2) Copy the get URL
- 3) Build steps - select invoke top level maven target
- 4) Select maven version
- 5) In Goals select test

N. 114

- 6) In post-build Actions
select publish HTML reports
In reports :-
HTML directory to archive : target
Index pages = *.xml
Report title = HTML Report

- 7) In post-build Actions
select Publish JUnit test result report
Test Report XMLs = target/surefire-reports/*.xml

- 8) To check if this file copied to local or not
the path *check*
→ cd /var/lib/jenkins/workspace/
→ In this, find your jenkins project name
→ Go to the folder
→ ls
→ cd target
→ ls
→ In this you will find the .xml file

Step 4:- Java maven - package

- 1) Create freestyle project
 - 2) Copy git url
 - 3) Build steps - Invoke top level maven project target.
 - 4) Maven version
 - 5) Goals - ~~test~~ install package
 - 6) In post build actions
 - 7) In build step select execute shell
 - 8) Add this command
- = cd target
= java -jar my-app-1.0-SNAPSHOT.jar
- This details checks in pom.xml file

Use case :-

Node JS application :-

Alt

- 1) Create a free style project
- 2) In build step select execute shell
- 3) Write these commands

sudo apt-get install nodejs npm -g -y
sudo npm install
sudo npm install -g pm2
sudo pm2 start index.js

check the file name in
github

- 4)
- 5) In the js file check for the port number
Set the IP address and the port number
you saw in js file.

Ansible :-

- 1) git clone AnsibleWork Updated git URL.
- 2) cd AnsibleWork Updated.
- 3) cd lampansible
- 4) sudo nano lamp.yml
- 5) In this check the host name.
- 6) Remove the version specified
- 7) save exit
- 8) sudo nano hosts
- 9) Add the host name of lamp.yml and
username & ip of the other system.
[lampserver]
172.16.51.47 ansible_ssh_user=msis
- 10) sudo ssh-keygen -t rsa
give a filename = bdatey
then enter → enter
sudo ssh-copy-id -i bdatey.pub msis@172.16.51.47
- 11) sudo ansible-playbook lamp.yml -i hosts
-c ssh --ask-pass -k.
Go to the host that connected system IP address