JENKINS DOCUMENTS

------------------

What is Continuous Integration?

In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process and hence the name continuous integration/deployment.

A Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to host, monitor, compile and test code, or code changes, like:

Continuous Integration Server (Jenkins, Bamboo, CruiseControl, TeamCity, and others)

Source Control Tool (e.g., CVS, SVN, GIT, Mercurial, Perforce, ClearCase and others)

Build tool (Make, ANT, Maven, Ivy, Gradle, and others)

Automation testing framework (Selenium, Appium, TestComplete, UFT, and others)

Jenkins Plugins

By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git. In fact, for integration with tools like Maven, Amazon EC2, you need to install respective plugins in your Jenkins.

Conclusion:

In Continuous Integration, after a code commit, the software is built and tested immediately

Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions

Before Jenkins when all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.

After Jenkins the code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day

By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to install plugins related to Git

The biggest pros of Jenkins is that it is managed by the community which holds public meetings and take inputs from the public for the development of Jenkins projects

The biggest con of Jenkin is that Its interface is out dated and not user friendly compared to current UI trends.

Software Requirements:

Since Jenkins runs on Java, you need either latest version of Java Development Kit (JDK) or Java Runtime Environment (JRE).

What is a Jenkins freestyle project?

A Jenkins project is a repeatable build job which contains steps and post-build actions. The types of actions you can perform in a build step or post-build action are quite limited. There are many standard plugins available within a Jenkins freestyle project to help you overcome this problem. They allow you to configure build triggers and offers project-based security for your Jenkins project.

What is Jenkins Pipeline?

In Jenkins, a pipeline is a group of events or jobs which are interlinked with one another in a sequence.

In simple words, Jenkins Pipeline is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins. A pipeline has an extensible automation server for creating simple or complex delivery pipelines "as code," via pipeline DSL (Domain-specific Language).

BUILD --- DEPLOY --- TEST ---- RELEASE

What is a JenkinsFile?

Jenkins pipelines can be defined using a text file called JenkinsFile. You can implement pipeline as code using JenkinsFile, and this can be defined by using a domain specific language (DSL). With JenkinsFile, you can write the steps needed for running a Jenkins pipeline.

The benefits of using JenkinsFile are:

You can create pipelines automatically for all branches and execute pull requests with just one JenkinsFile.

You can review your code on the pipeline

You can audit your Jenkins pipeline

This is the singular source for your pipeline and can be modified by multiple users.

JenkinsFile can be defined by either Web UI or with a JenkinsFile.

Declarative versus Scripted pipeline syntax:

There are two types of syntax used for defining your JenkinsFile.

Declarative

Scripted

Declarative:

Declarative pipeline syntax offers an easy way to create pipelines. It contains a predefined hierarchy to create Jenkins pipelines. It gives you the ability to control all aspects of a pipeline execution in a simple, straight-forward manner.

Scripted:

Scripted Jenkins pipeline runs on the Jenkins master with the help of a lightweight executor. It uses very few resources to translate the pipeline into atomic commands. Both declarative and scripted syntax are different from each other and are defined totally differently.

Jenkins Pipeline Concepts

Term Description

Pipeline The pipeline is a set of instructions given in the form of code for continuous delivery and consists of instructions needed for the entire build process. With pipeline, you can build, test, and deliver the application.

Node The machine on which Jenkins runs is called a node. A node block is mainly used in scripted pipeline syntax.

Stage A stage block contains a series of steps in a pipeline. That is, the build, test, and deploy processes all come together in a stage. Generally, a stage block is used to visualize the Jenkins pipeline process.

Step A step is nothing but a single task that executes a specific process at a defined time. A pipeline involves a series of steps.

What is CI?

Continuous integration is a software development method where members of the team can integrate their work at least once in a day. In this method, every integration is checked by an automated build to search the error. The CI concept was first introduced over two decades ago to avoid "integration hell," which happens when integration is put off till the end of a project.

Master Slave configuration

--------------------------

In Slave install openjdk and docker

# apt-get install default-openjdk ( all slave)

# apt-get install docker.io

Deploy two servers ( slave1 and slave2 )

Go to jenkins master – jenkins – configure – global security

Agents : randam

Save

Manage notes

Add new nodes slave1

Permenent agent

Slave1 configure

Launch method – launch agent via java web start

Current working dir /home/ubuntu/jenkins

Save

Same as slave 2

Click on salve1

Download agent.jar

Copy agant.jar file to slave1 server /home/ubuntu/jenkins

Login to slave1 server

( select node slave1 and copy java –jar agant.jar ……………………………… )

Cd /home/ubuntu/jenkins

Run that command ( java –jar agant.jar …………)

After run this command don’t close the terminal

Now slave are connected to jenkins server.

Go to jenkins

Create job

Test and freestyle project

Go to configure github project

Project url : <http://github-server/.../project.git>

Select – source code management

Repository usrl : <http://github-server/.../project.git>

Enable restickt where this project can be run ( this job only run slave1)

Slave1

Save and build

Build – Add built setup – execute shell

Command : Docket rm –f ${sudo docker ps –a –q}

docket build /home/ubuntu/jenkins/workspace/test –t test

Docket run –it –p 82:80 –d test

Create new project

Prod and freestyle project

( same as slave 1)

Go to configure github project

Project url : <http://github-server/.../project.git>

Select – source code management

Repository usrl : <http://github-server/.../project.git>

Enable restickt where this project can be run ( this job only run slave2)

Slave2

Save and build

Command : Docket rm –f ${sudo docker ps –a –q}

docket build /home/ubuntu/jenkins/workspace/prod –t test

Docket run –it –p 82:80 –d test

Go to test job and configure

Post-build actions

Projects to build : prod

Select trigger only if build is statble

Select + sign in dashboard ( for pipeline )

Install buildpipeline

Select + sign in dashboard

View name : CICD

Slect build pipe line view

Click on ok

Then build pipe line view title : CICD

Select initial job : test

For auto deploy

Go to test job – configure - Build trigger

Github hook trigger to gitscm polling

Configure web hook in github

PHP Unit Testing

Go to Jenkins server terminal

# apt-get update

# apt-get install phpunit

Go to Jenkins URL s

Install junit plugin

Manage plugin – plugin manager – available

Select junit plugin and install

Select job first-project

Go to build Trigger – execute shell

Command : phpunit CalculaterTest.php –log-junit php-result.xml

Post build action : publish junit test result report

Test report xml : php-result.xml

Integrate bitbucket status API

First create OR key

Go to bitbucket – profile (user) - setting - Access management – OAuth

Select Add consumer

Name : first-project

Calback URL : <http://jenkins-ip:8080/>

Repository : select read and write

Install bitbucket OAuth plugin

Go to jenkins URL

Jenkin – manage plugin – plugin manager – available

Select bitbucket OAuth and bitbucket bukld status notified plugin and install

Jenkins - first-project – post build actions

Enable bitbucket notify build status and notify build start , notify build finish

Credentioal - Add OAuth Keys ( from bitbucket server )

Install FTP server [ QA server ]

# apt-get install vsftpd

# vim /et/vsftpd.conf ( go to line no 31 and uncommand write enable = yes)

#systemctl restart vsftpd

# useradd john

# su – john

# mkdir data

# cd data

Install git ftp in Jenking Server Terminal

# apt-get install git-ftp

Install credentioal binding plugin

Jenkins – credentioal – add credentioal

Jenkins – first-project – build Environment – use secret text

Binding – user name and password seprated

Username variable GITUser

Password variable GITPASSWORD

Credential -

Execute shell

Git ftp init –-user $GITUSER –GITPASSWORD <ftp://qaserver-ip/data> ( init command will execute first time only)

Git ftp push –-user $GITUSER –GITPASSWORD <ftp://qaserver-ip/data>

tomcat with jenkins

----------------------

Step1: Make Sure you have Git and Maven installed

In Jenkins UI, Goto Manage Jenkins -> Global Tool Configuration git and mavan

Step2: Install Deploy to Container Plugin

Manage Jenkins -> Manage Plugins -> Available -> Deploy to Container Plugin

Note:\* For the Next step we have selected a Maven Job as our Choice.

you can also create a Free Style Project and use Gradle or Ant as your build tool .

Step3: Create and Configure a Maven Job with Source Code Management (Github)

New Item -> Maven Project

In the Configuration Section, Under Source Code Management Fill your

Github/BeanStalk/Gitlab Repository URL

Step4: Configure the Post-build Action and Specify the Tomcat Server Details

Drag to the bottom and Go to the Post-build Actions section

Step5: Click on Add post-build action button

On the available options click on the Deploy war/ear to container

Choose the Context Path in which the application should be installed.

It would rename the WAR file before deploying to the server and thereby the application context root would be changed.

Tomcat URL http://[Tomcat Server Host]:[Primary http port]/

DOCKEAR

# How to start and stop the container

docker stop <containername/id> # to stop the container which is running in background mode

docker container start <containername/id> # to start the already stopped container

# monitoring the Container

docker container list # list the containers created

docker container ls -a #to list the containers including not running containers

docker ps #to list the running container

docker info #docker engine status including container running/paused/stopped containers list

docker container stats <containername/id> #prints the CPU and MEM usage data of the container name

docker container stats #prints the CPU and MEM usage of all running containers

docker container top <containername/id> #executes the top command in the container specified,

fails if the specified container is not running

# How to remove/delete the container

docker container rm <containername/id> # Remove the Container

# How to remove/delete the image

docker container rmi <imagename/imageid> # Remove the image

# How to view the logs of the running container

docker container logs <containername/id> #

FROM centos

MAINTAINER aksarav@middlewareinventory.com

RUN mkdir /opt/tomcat/

WORKDIR /opt/tomcat

RUN curl -O https://www-eu.apache.org/dist/tomcat/tomcat-8/v8.5.40/bin/apache-tomcat-8.5.40.tar.gz

RUN tar xvfz apache\*.tar.gz

RUN mv apache-tomcat-8.5.40/\* /opt/tomcat/.

RUN yum -y install java

RUN java -version

WORKDIR /opt/tomcat/webapps

RUN curl -O -L https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war

EXPOSE 8080

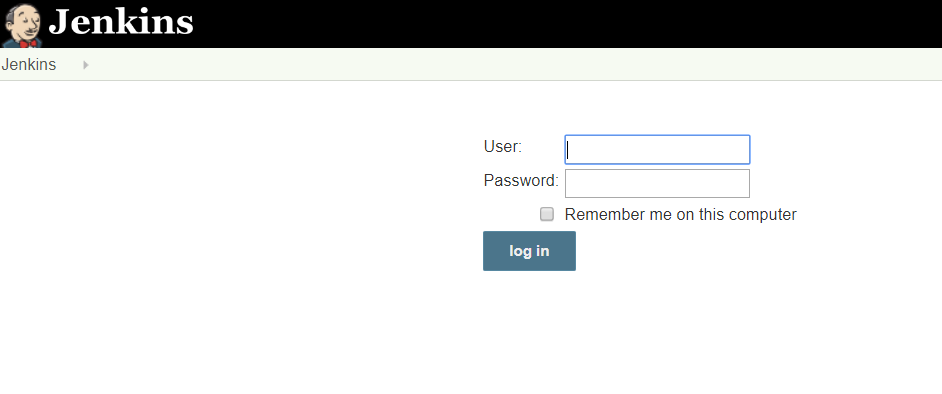
CMD ["/opt/tomcat/bin/catalina.sh", "run"]

## Configure apache using Jenkins:

Our PHP code resides on a SCM (Source code Management) server, which is in this case is Github. Jenkins (CI server) will fetch the code from the github account automatically upon check-in. Jenkins will run the build and will deploy the code in the /var/www/html directory of the Apach2 web server. To accomplish this Jenkins instance has to connect to the Apache2 instance via SSH to trigger the deployment process.

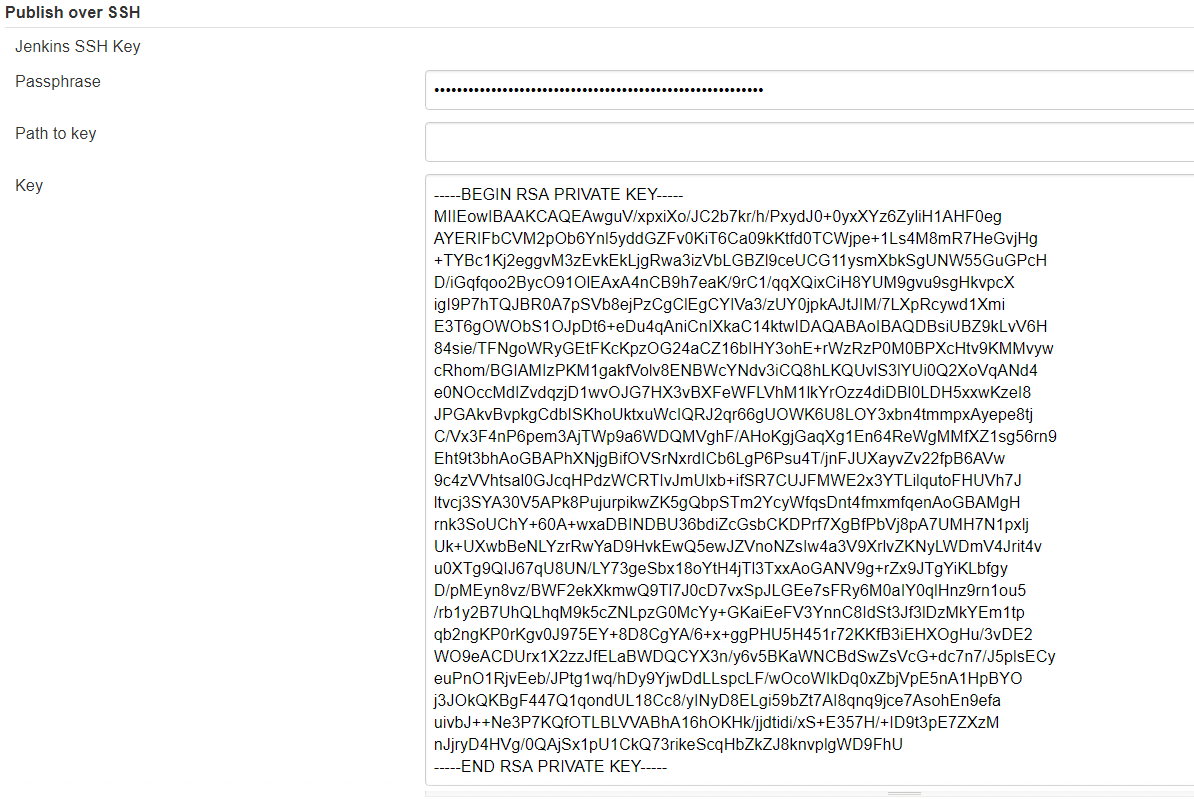
## https://miro.medium.com/max/646/1*8IeSB0_e4ZQmXJMskHp9Cw.png

Step 1. Go to browser and start Jenkins server at default port 8080



Step 2. After login, Go to Manage Jenkins → Manage Plugins → Available → search for “Publish over SSH” plugin → install without restart.

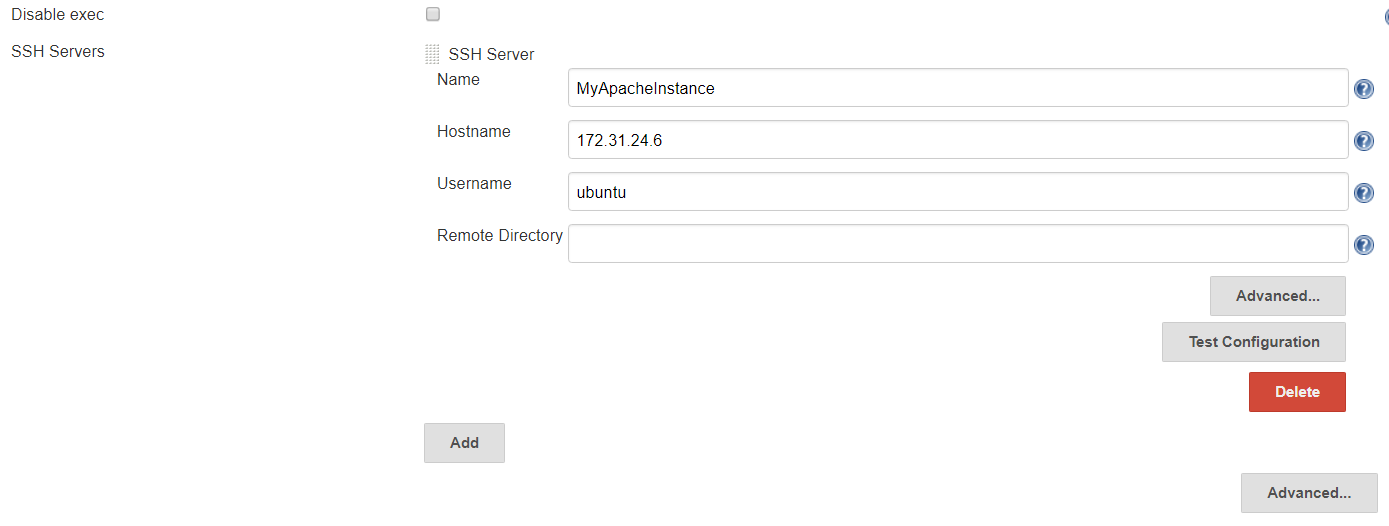
Step 3. Go to Manage Jenkins → Configure System → Publish over SSH



Either provide the path to the generated ssh key or paste it directly. It is important to paste everything including header and footer as shown in the above snip from my lab experiment.

If wondering how to generate a SSH key to establish connection between two servers, click [**here**](https://devops4solutions.com/ssh-aws-ec2-instances/) to get more details and steps.

Then click on ADD button in order to add a server to SSH with/ con

nect with.

Fill in the details, like;

* Name: Provide any logical name which can be used later on to configure job.
* Hostname: ip address of the server to connect to (in this case it is the instance second hosting Apache2)
* Username: name of the user to login to (in this case it is ubuntu)
* Remote Directory: path of any directory you want to deploy to (or can be leaved blank for later configuration within the job)

Step 4. Click ‘Test Configuration’ to confirm the connection and ‘Save’ at the bottom of the page.

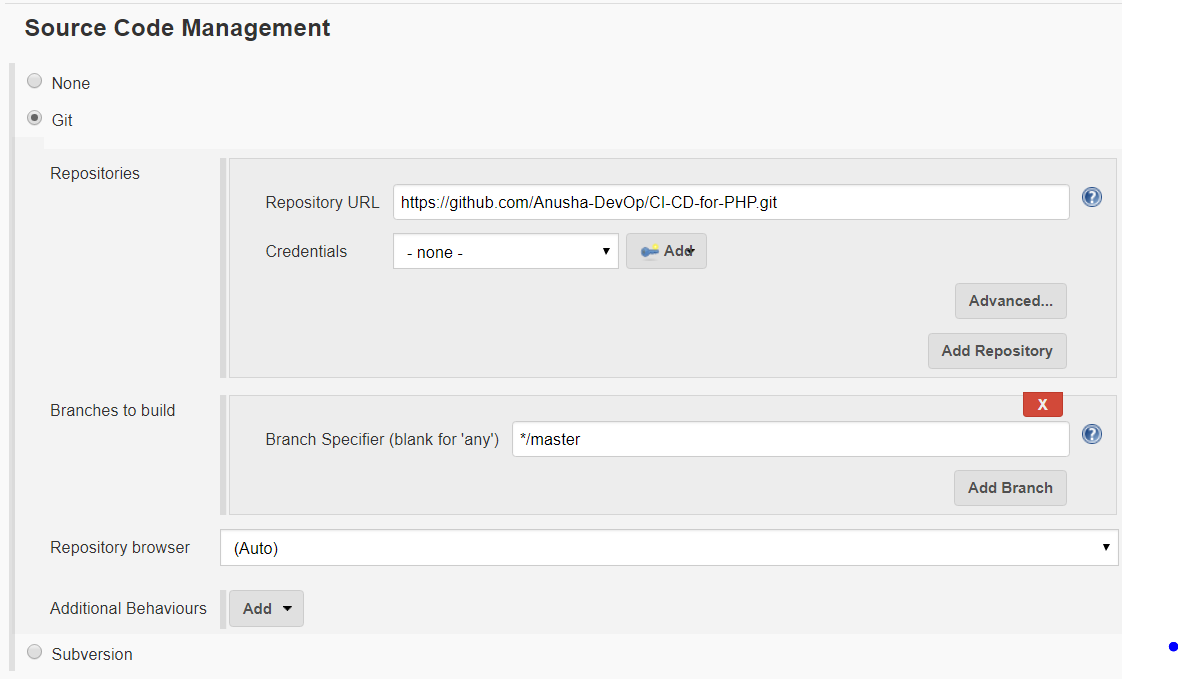
Note: Both servers must be up and running to test the configuration.

**Create a Jenkins Job:**

Step 1. Go to Jenkins Dashboard → Click on ‘New Item’ → Provide name of the project (e.g. CI-CD-PHP) → choose ‘Freestyle Job’ → Click ‘OK’.

In the configuration window of the job:

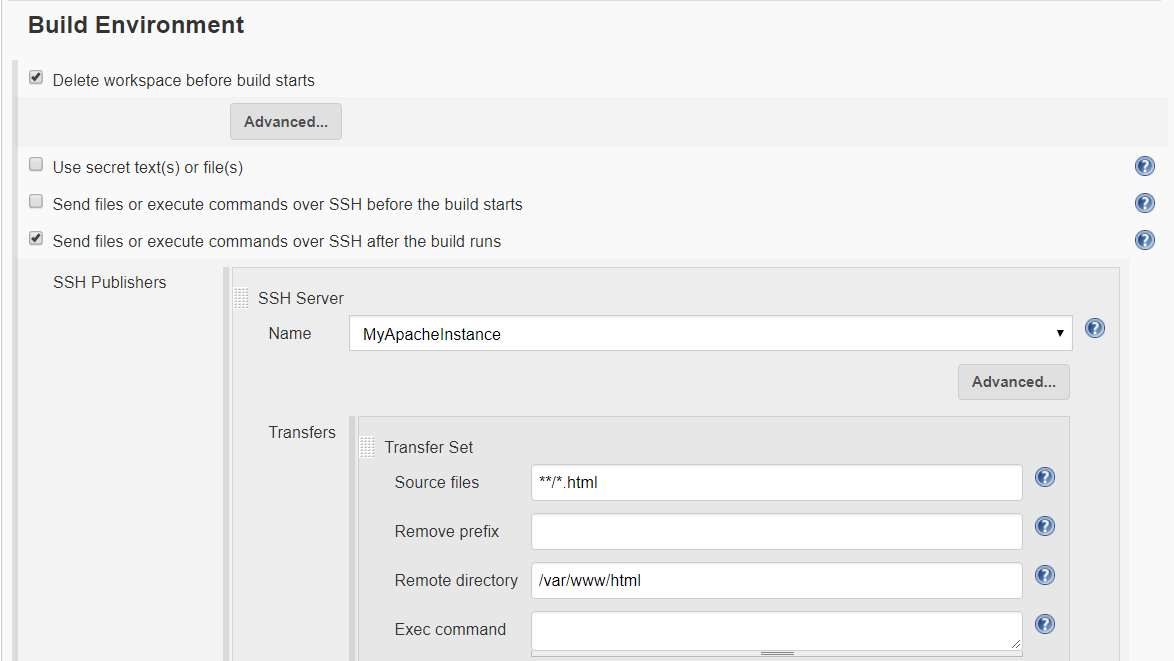
Step 2. Provide the Git URL from where code has to be pulled from.



Step 3. In the Build Environment section choose:

a). Delete workspace before build starts.

b). Send files or execute commands over SSH after the build runs



Provide the Name of the server, source files and remote directory.

https://github.com/ValaxyTech/Hello-world

https://github.com/ValaxyTech/hello-world

create project --- project name -- source code management -- git ---

build -- goal -- clean install package --

manage plugin -- deploy to container

set the creditial

tomcat creditial - jenkins - global creditials -

-add credti -username and passwd

select project - congigure - postbuild action

- deploy war to a container -

war/ear files

container - tomcat 8

creditaial - tomcat url -

build triggers

poll scm \*/2 \* \* \* \*

---------------------------------------

project 2

remove postbuild

login into ansible server

useradd ansadmin

vi sudo

ansadmin ALL=(ALL) NOPASSWD:ALL

passwd ansadmin

login to tomcat server

----------------------

useradd ansadmin

visudo

ansiadmin ALL = (ALL) NOPASSED:ALL

passwd ansadmin

ansible server

ssh-keygen

ssh-copy-id targetsystem(tomcat server)

login jenkins

goto plugin - add - publishoverssh

in ansible playbook

copyfile.yml

---

- hosts: web-servr

become: true

tasks:

- name: copy war onto tomcat server

copy:

src: /opt/playbooks/webapp/target/webapp.war

dest: /opt/apache-tomcat-8.5.32/webapp

systemconfiguration - ssh server - add - Name-

ansible server -- hostname - ip

seelect passwordless auth

select job -- configure --

addpost build step send or create configuration over ssh

target -- sourece file (ansible server) webapp/target/\*.war

remote dirctory -- /opt/playbooks

entercommand - ansible-playbook .yml

---------------------

simple devops project 3

git jenkins docker

1. launch ec2 instance for docker host

install docker on ec2 instance and start services

create new user for docker management and ad him to docker default group

ueradd dockeradmin

paswd

useradd -aG docker dockeradmin

mkdir /opt/DockerFile

FROM tomct:8-jred

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