https://www.middlewareinventory.com/blog/tomcat-modjk-configuration-installation/

<https://www.middlewareinventory.com/blog/docker-tomcat-example-dockerfile-sample/>

<https://dzone.com/articles/learn-how-to-setup-a-cicd-pipeline-from-scratch> (CI/CD pipe line tutorials)

<http://www.inanzzz.com/index.php/post/qw41/deploying-a-dockerised-application-with-ansible-as-part-of-ci-cd-jenkins-pipeline> ( ci/cd code based pipeline )

mavan life cycle

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

validate - validate the project is correct and all necessary information is available

compile - compile the source code of the project

test - test the compiled source code using a suitable unit testing framework.

These tests should not require the code be packaged or deployed

package - take the compiled code and package it in its distributable format, such as a JAR.

verify - run any checks on results of integration tests to ensure quality criteria are met

install - install the package into the local repository, for use as a dependency in other projects locally

deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

jenkins\_metrics\_and\_trends

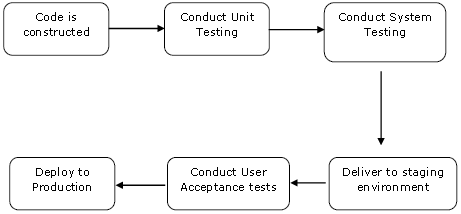
There are various plugins which are available in Jenkins to showcase metrics for builds which are carried out over a period of time. These metrics are useful to understand your builds and how frequently they fail/pass over time. As an example, let’s look at the ‘Build History Metrics plugin’.

This plugin calculates the following metrics for all of the builds once installed

Mean Time To Failure (MTTF)

Mean Time To Recovery (MTTR)

Standard Deviation of Build Times



Step 1 − Go to the Jenkins dashboard and click on New Item. Choose a ‘Freestyle project’ and enter the project name as ‘QA’. Click on the Ok button to create the project.

Step 2 − In this example, we are keeping it simple and just using this project to execute a test program for the Helloworld application.

Select build option and select Command

Javac HelloWorldTest.java

Java HelloWorldTest

So our project QA is now setup. You can do a build to see if it builds properly.

Step 3 − Now go to you Helloworld project and click on the Configure option

Step 4 − In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’

Step 5 − In the ‘Project to build’ section, enter QA as the project name to build. You can leave the option as default of ‘Trigger only if build is stable’. Click on the Save button.

Step 6 − Build the Helloworld project. Now if you see the Console output, you will also see that after the Helloworld project is successfully built, the build of the QA project will also happen.

Step 7 − Let now install the Delivery pipeline plugin. Go to Manage Jenkins → Manage Plugin’s. In the available tab, search for ‘Delivery Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.

Step 8 − To see the Delivery pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

Step 9 − Enter any name for the View name and choose the option ‘Delivery Pipeline View’.

Step 10 − In the next screen, you can leave the default options. One can change the following settings −

Ensure the option ‘Show static analysis results’ is checked.

Ensure the option ‘Show total build time’ is checked.

For the Initial job – Enter the Helloworld project as the first job which should build.

Enter any name for the Pipeline

Click the OK button.

You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.

Another famous plugin is the build pipeline plugin. Let’s take a look at this.

Step 1 − Go to Manage Jenkins → Manage Plugin’s. In the available tab, search for ‘Build Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.

Step 2 − To see the Build pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

Step 3 − Enter any name for the View name and choose the option ‘Build Pipeline View’.

Step 4 − Accept the default settings, just in the Selected Initial job, ensure to enter the name of the Helloworld project. Click on the Ok button.

You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.

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"]

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

root@ubuntu-xenial:~# cat FileforSupervisor/Dockerfile

FROM ubuntu

MAINTAINER <anand@brainstack.in>

RUN apt-get update

RUN apt-get install supervisor apache2 openssh-server -y

RUN mkdir /var/lock/apache2 /var/run/apache2 /var/run/sshd

RUN sed -i '/\[supervisord\]/a nodaemon=true' /etc/supervisor/supervisord.conf

RUN echo 'root:redhat123' | chpasswd

RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config

COPY myservice.conf /etc/supervisor/conf.d/

EXPOSE 80 22

CMD ["supervisord"]

root@ubuntu-xenial:~# cat FileforSupervisor/myservice.conf

[program:sshd]

command=/usr/sbin/sshd -D

[program:apache2]

command=/bin/bash -c "source /etc/apache2/envvars && exec /usr/sbin/apache2 -DFOREGROUND"

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

root@ubuntu-xenial:~# cat FileforEntrypoint/Dockerfile

FROM ubuntu

MAINTAINER <anand@brainstack.in>

RUN apt-get update

RUN apt-get install apache2 -y

RUN mkdir /var/lock/apache2 /var/run/apache2

ENV APACHE\_RUN\_USER www-data

ENV APACHE\_RUN\_GROUP www-data

ENV APACHE\_LOCK\_DIR /var/lock/apache2

ENV APACHE\_PID\_FILE /var/run/apache2/apache2.pid

ENV APACHE\_LOG\_DIR /var/log/apache2

ENV APACHE\_RUN\_DIR /var/run/apache2

ENV LANG C

WORKDIR /var/www/html

COPY ./entrypoint.sh /root/entrypoint.sh

ENTRYPOINT /root/entrypoint.sh

EXPOSE 80

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

root@ubuntu-xenial:~# cat FileforEntrypoint/entrypoint.sh

#!/bin/bash

rm -f /var/run/apache2/apache2.pid

/usr/sbin/apache2 -DFOREGROUND

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

FROM ubuntu

MAINTAINER <anand@brainstack.in>

RUN apt-get update

RUN apt-get install apache2 -y

RUN mkdir /var/lock/apache2 /var/run/apache2

ENV APACHE\_RUN\_USER www-data

ENV APACHE\_RUN\_GROUP www-data

ENV APACHE\_LOCK\_DIR /var/lock/apache2

ENV APACHE\_PID\_FILE /var/run/apache2/apache2.pid

ENV APACHE\_LOG\_DIR /var/log/apache2

ENV APACHE\_RUN\_DIR /var/run/apache2

ENV LANG C

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]

EXPOSE 80

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

FROM ubuntu

MAINTAINER anand@brainstack.in

RUN apt-get update

RUN apt-get install openssh-server -y

RUN cp /etc/ssh/sshd\_config /root/sshd\_config.org

RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config

RUN echo 'root:redhat123' | chpasswd

RUN mkdir /var/run/sshd

CMD ["/usr/sbin/sshd", "-D"]

EXPOSE 22

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

FROM ubuntu

MAINTAINER anand@brainstack.in

RUN apt-get update

RUN apt-get install openssh-server -y

RUN cp /etc/ssh/sshd\_config /root/sshd\_config.org

RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config

RUN echo 'root:redhat123' | chpasswd

RUN mkdir /var/run/sshd

CMD ['/usr/sbin/sshd', '-D']

EXPOSE 22

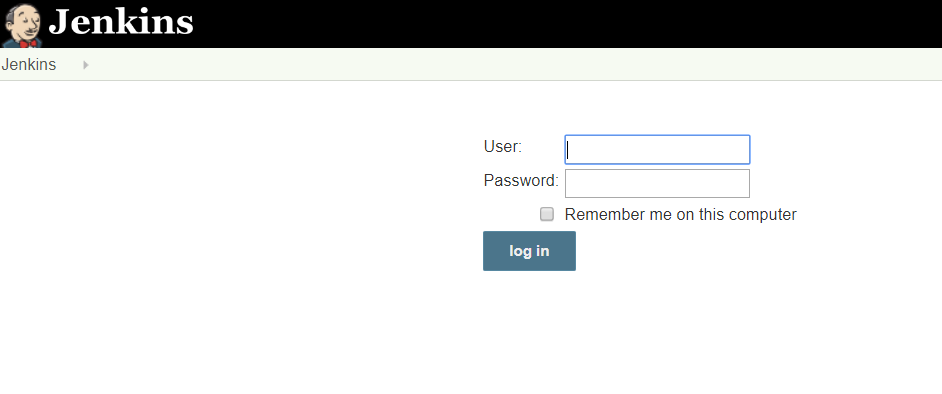
---------------------------------------------------------------------------------------------

## 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.

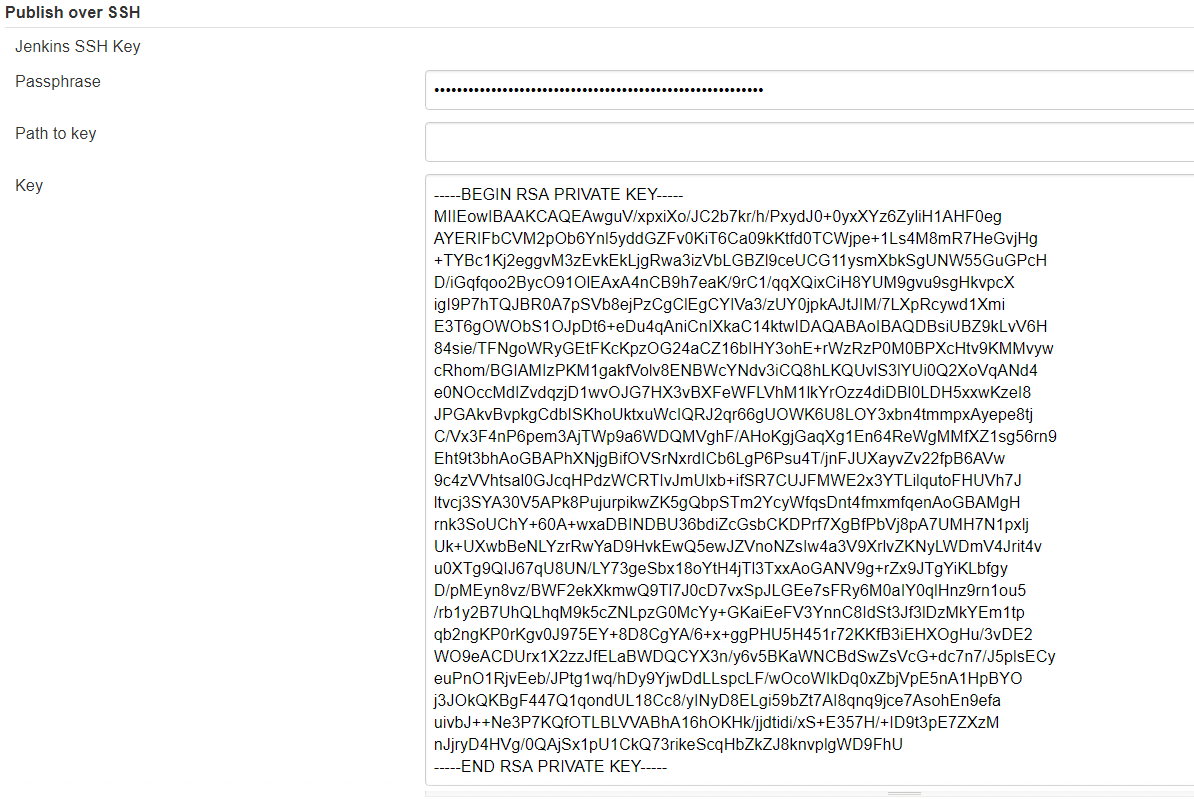
## https://miro.medium.com/max/646/1*8IeSB0_e4ZQmXJMskHp9Cw.png

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.

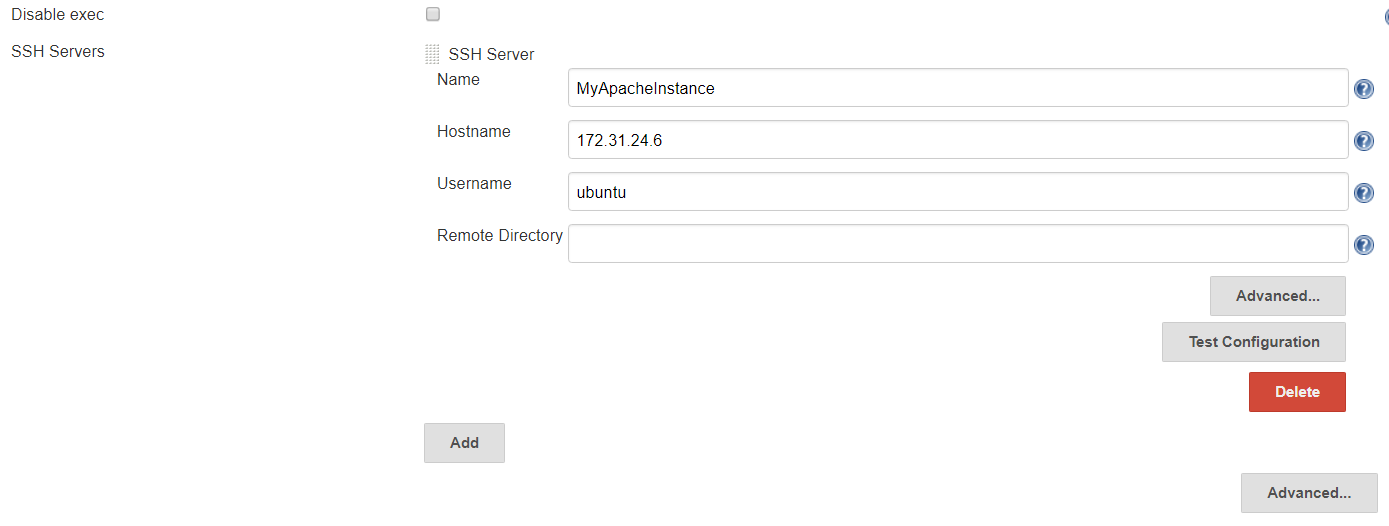
Step 3. Go to Manage Jenkins → Configure System → Publish over SSH



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.

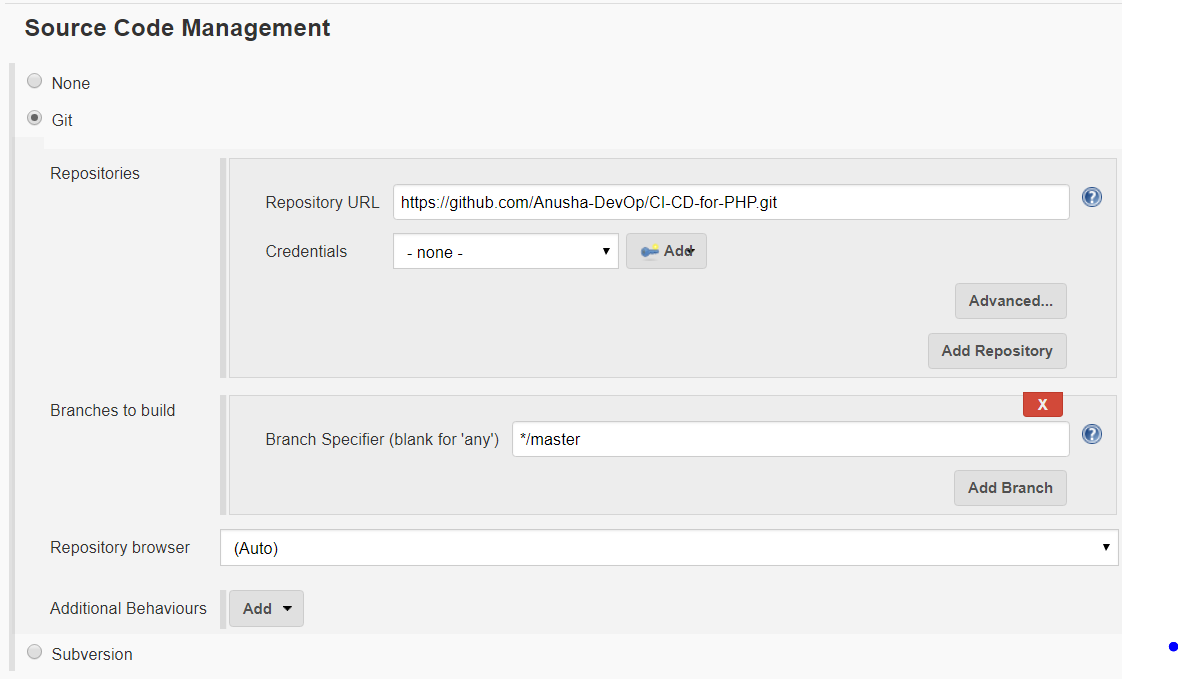
Note: Both servers must be up and running to test the configuration.

**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:

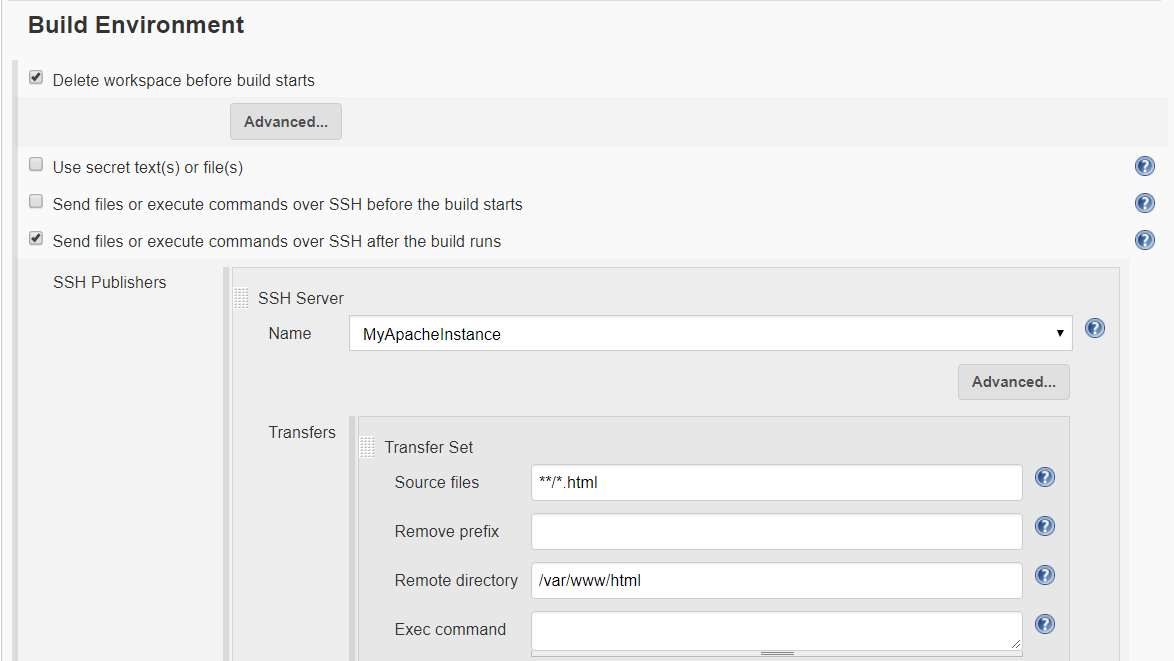
Step 2. Provide the Git URL from where code has to be pulled from.



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.