1. **Overview of Jenkins**

Jenkins generate war file which is called as Artifact.

If source is DevopsBasics/webapp/src/main/webapp/index.jsp then

Location of war file is **workspace/webapp/targer/**[**webapp.war**](http://20.55.101.111:8080/job/DevopsBasics/ws/webapp/target/webapp.war)

/var/lib/jenkins/workspace is workspace directory.(when you configure build you have option called as **Delete workspace before build starts**)

/var/lib/jenkins/config.xml config file for jenkins.

1. **Installation of Jenkins on Jenkins Server.**

wget -O <http://pkg.jenkins.io/redhat/jenkins.repo>

rpm --import <http://pkg.jenkins.io/redhat/jenkins.io.key>

yum install jenkins

Start and enable Jenkins( http:// JenkinsMaster:8080)

1. **Install java java-1.8.0-openjdk-devel**

Remember, it must be an SDK (Software Developer Kit), not just a JRE (Java Runtime Environment). The JRE does not contain a Java compiler. Only the SDK does.

[amitadmin@JenkinsMaster ~]$ rpm -qa | grep -i java

tzdata-java-2020d-1.el8.noarch

javapackages-filesystem-5.3.0-1.module\_el8.0.0+11+5b8c10bd.noarch

java-1.8.0-openjdk-1.8.0.272.b10-1.el8\_2.x86\_64

java-1.8.0-openjdk-headless-1.8.0.272.b10-1.el8\_2.x86\_64

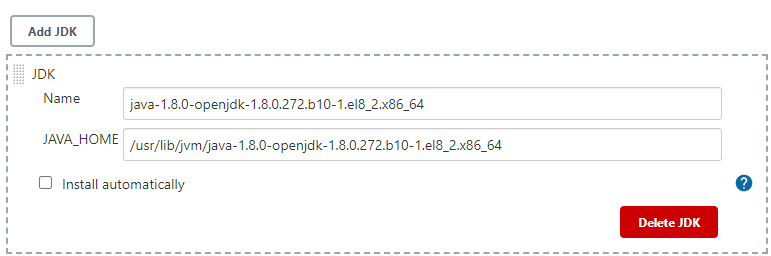
java-1.8.0-openjdk-devel-1.8.0.272.b10-1.el8\_2.x86\_64

[root@JenkinsMaster jenkins]# ls -l /usr/bin/java -> /etc/alternatives/java

[root@JenkinsMaster jenkins]# ls -l /etc/alternatives/java

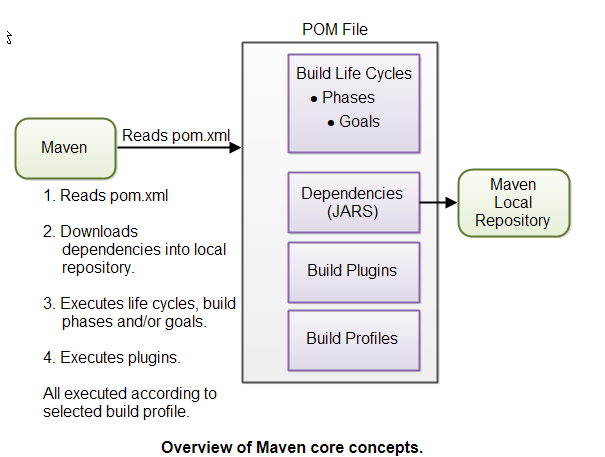
lrwxrwxrwx. 1 root root 73 Nov 17 15:09 /etc/alternatives/java -> /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.el8\_2.x86\_64/jre/bin/java

**All these paths are set in Global Tool Configuration.**



1. **Install Maven**

[http://maven.apache.org](http://maven.apache.org/)



**POM Files**  
When you execute a Maven command you give Maven a POM file to execute the commands on. Maven will then execute the command on the resources described in the POM.

The POM file describes what to build, but most often not how to build it. How to build it is up to the Maven build phases and goals. You can insert custom actions (goals) into the Maven build phase if you need to, though.

Each project has a POM file. The POM file is named pom.xml and should be located in the root directory of your project. A project divided into subprojects will typically have one POM file for the parent project, and one POM file for each subproject. This structure allows both the total project to be built in one step, or any of the subprojects to be built separately.

<modelVersion>4.0.0</modelVersion> The modelVersion element sets what version of the POM model you are using.

<groupId>com.example.maven-project</groupId> The groupId element is a unique ID for an organization, or a project (an open source project, for instance). Most often you will use a group ID which is similar to the root Java package name of the project.

The group ID does not have to be a Java package name and does not need to use the . notation (dot notation) for separating words in the ID. But, if you do, the project will be located in the Maven repository under a directory structure matching the group ID. Each . is replaced with a directory separator, and each word thus represents a directory. The group ID com.jenkov would then be located in a directory called MAVEN\_REPO/com/jenkov. The MAVEN\_REPO part of the directory name will be replaced with the directory path of the Maven repository.

<artifactId>maven-project</artifactId> The artifactId element contains the name of the project you are building. In the case of my Java Web Crawler project, the artifact ID would be java-web-crawler. The artifact ID is used as name for a subdirectory under the group ID directory in the Maven repository. The artifact ID is also used as part of the name of the JAR file produced when building the project. The output of the build process, the build result that is, is called an artifact in Maven

<version>1.0-SNAPSHOT</version> The versionId element contains the version number of the project. If your project has been released in different versions, for instance an open source API, then it is useful to version the builds.

**Build Life Cycles, Phases and Goals**The build process in Maven is split up into build life cycles, phases and goals. A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals. When you run Maven you pass a command to Maven. This command is the name of a build life cycle, phase or goal. If a life cycle is requested executed, all build phases in that life cycle are executed. If a build phase is requested executed, all build phases before it in the pre-defined sequence of build phases are executed too.

**Dependencies and Repositories**

One of the first goals Maven executes is to check the dependencies needed by your project. Dependencies are external JAR files (Java libraries) that your project uses. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts

them in your local repository.

**Maven vs. Ant**

Ant uses an imperative approach, meaning you specify in the Ant build file what actions Ant should take. You can specify low level actions like copying files, compiling code etc. You specify the actions, and you also specify the sequence in which they are carried out. Ant has no default directory layout.

Maven uses a more declarative approach, meaning that you specify in the Maven POM file what to build, but not how to build it. The POM file describes your project resources - not how to build it. Contrarily, an Ant file describes how to build your project. In Maven, how to build your project is predefined in the [Maven Build Life Cycles, Phases and Goals](http://tutorials.jenkov.com/maven/maven-tutorial.html#maven-build-life-cycles-phases-and-goals).

**mvn install**

This command executes the build phase called install (part of the default build life cycle), which builds the project and copies the packaged JAR file into the local Maven repository. Actually, this command executes all build phases before install in the build phase sequence, before executing the install build phase.

**mvn clean install**

This command first executes the clean build life cycle, which removes compiled classes from the Maven output directory, and then it executes the install build phase.

**mvn dependency:copy-dependencies**

You can also execute a Maven goal (a subpart of a build phase) by passing the build phase and goal name concatenated with a : in between, as parameter to the Maven command.

[amitadmin@JenkinsMaster bin]$ pwd

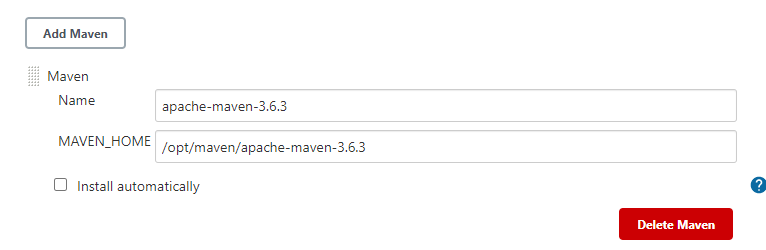
/opt/maven/apache-maven-3.6.3/bin

[amitadmin@JenkinsMaster bin]$

Note - command is mvn clean install, but in jenkins you just put clean install while configuring job.

Note - ng build command - Compiles an Angular app into an output directory named dist/at the given output path. Must be executed from within a workspace directory.

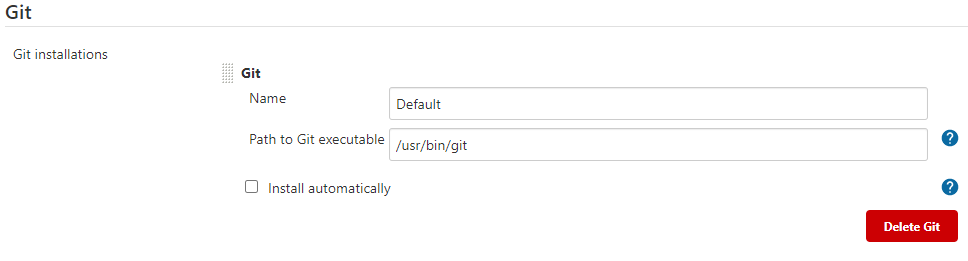
ng build <project> [options]https://stackoverflow.com/questions/52170298/jenkins-angular-how-to-execute-ng-build-after-git-push)



1. **Install git**

[root@JenkinsMaster jenkins]# /usr/bin/git --version

git version 2.18.4



1. **Install plugin** [**Publish Over SSH**](https://plugins.jenkins.io/publish-over-ssh)

This will send build artifact over ssh , Login with jenkins user using command

**su - -s /bin/bash jenkins**

this command is required as /bin/false shell is assigned to jenkins user in password file and generate ssh key for jenkins user. Note the PEM file normal key does not work with jenkins.

**ssh-keygen -t rsa -b 4096 -m PEM**

**Copy public key to docker01 and ansible server in /home/amitadmin/.ssh/** **authorized\_keys (400 permission of this file and 600 for .ssh folder)**

# Now install plugin from dashboard > Manage Jenkins > Manage plugins > Publish Over SSH

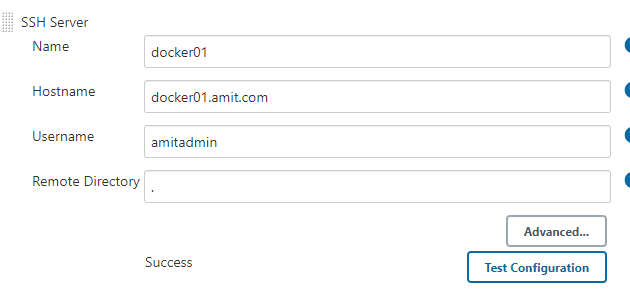
It will give 2 options

1. Send files or execute commands over SSH before the build starts
2. Send files or execute commands over SSH after the build runs.

Now from jenkins dashboard add ansible and docker01 system using the key generated above.(make sure to copy public key in authorized\_keys file on remote host)

dashboard > Manage Jenkins > Configure System > Publish over SSH





1. **Create repository in github.com for your project.**

Use folder name as repository name in github.com example DevopsBasics.

cd DevopsBasics (this folder contain project all files and directories)

git init

git add README.md

git commit -m "first commit"

git branch -M main ( we generally run git branch -M master)

git remote add origin git@github.com:amitsuneja/DevopsBasics.git

git push

1. **Test your repository**

cd /tmp

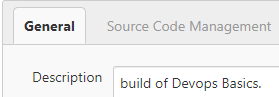
git clone <https://github.com/amitsuneja/DevopsBasics.git>.

Rm -rf DevopsBasics

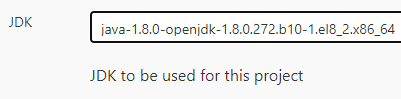
1. **Create build job in jenkins.**

DashBoard > New Item > BuildOfDevopsBasics > Freestyle project > ok.

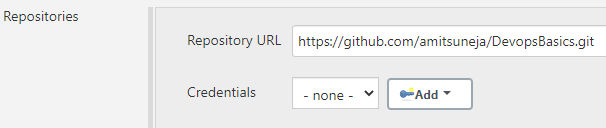
Description > build of Devops Basics.

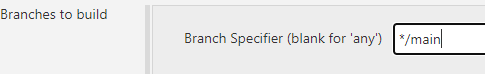


Select your JDK version



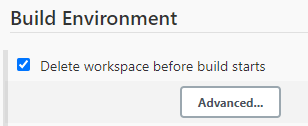
Source Code Management:



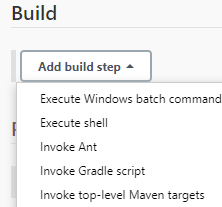


Build Triggers > GitHub hook trigger for GITScm polling (We will use it later)

Build Triggers > Poll SCM (check git after some specific time)



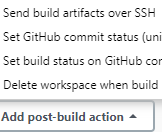


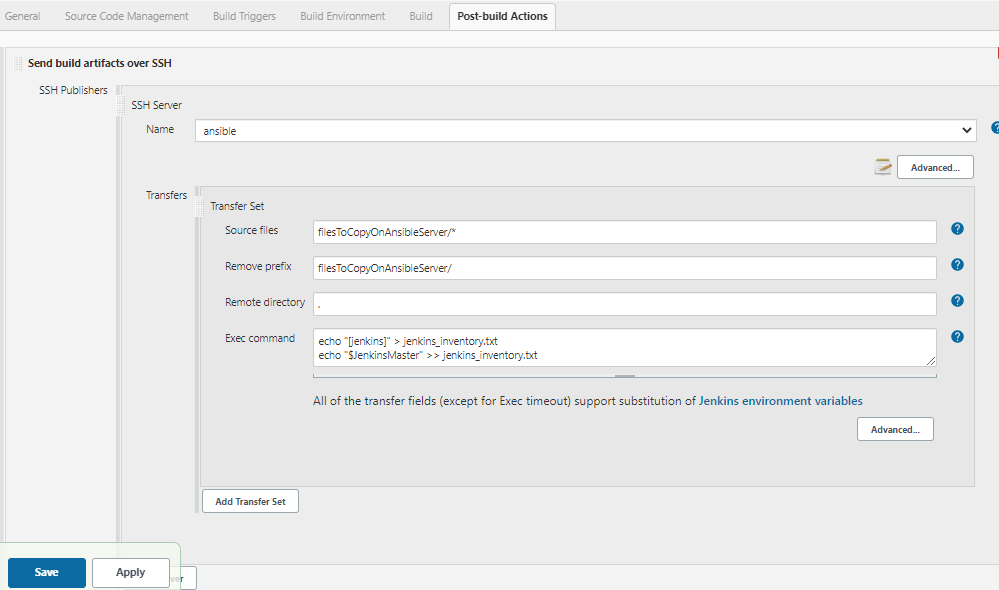




Add post build Action to copy Dockerfile and cpWarAndDockerFile2Ansible.yml, createWebServer.yml file to Ansible server using send build artifact over ssh.

Also, we created file jenkins\_inventory.txt on ansible server in this step





Note – Now we will run cpWarAndDockerFile2Ansible.yml from ansible server which will copy Dockerfile and createWebServer.yml to ansible server. This step is not required . But I am adding it just as an example. As its Test environment.

1. **Make changes in docker server (This step is wrongly mentioned here and is not required, you can skip it but we may use on some other project)**

so that docker API can be called from ansible server

Edit file : /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

TO

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:4243

systemctl daemon-reload

systemctl restart docker.service;systemctl restart containerd.service

systemctl status -l containerd.service; systemctl status -l docker.service

[root@docker01 ~]# curl http://localhost:4243/version

{"Platform":{"Name":"Docker Engine - Community"},"Components":[{"Name":"Engine","Version":"19.03.13","Details":{"ApiVersion":"1.40","Arch":"amd64","BuildTime":"2020-09-16T17:02:21.000000000+00:00","Experimental":"false","GitCommit":"4484c46d9d","GoVersion":"go1.13.15","KernelVersion":"3.10.0-1127.19.1.el7.x86\_64","MinAPIVersion":"1.12","Os":"linux"}},{"Name":"containerd","Version":"1.3.7","Details":{"GitCommit":"8fba4e9a7d01810a393d5d25a3621dc101981175"}},{"Name":"runc","Version":"1.0.0-rc10","Details":{"GitCommit":"dc9208a3303feef5b3839f4323d9beb36df0a9dd"}},{"Name":"docker-init","Version":"0.18.0","Details":{"GitCommit":"fec3683"}}],"Version":"19.03.13","ApiVersion":"1.40","MinAPIVersion":"1.12","GitCommit":"4484c46d9d","GoVersion":"go1.13.15","Os":"linux","Arch":"amd64","KernelVersion":"3.10.0-1127.19.1.el7.x86\_64","BuildTime":"2020-09-16T17:02:21.000000000+00:00"}

[root@docker01 ~]#

1. **Make Lets create a DockerFile in Docker Server and test that local docker commands are working then copy it in git repo.**

[amitadmin@docker01 ~]$ pwd

/home/amitadmin

[amitadmin@docker01 ~]$ cat Dockerfile

FROM tomcat:latest

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

[amitadmin@docker01 ~]$ ls -ltr

total 4

-rw-rw-r--. 1 amitadmin amitadmin 133 Jan 4 10:35 Dockerfile

-rw-rw-r--. 1 amitadmin amitadmin 0 Jan 4 10:35 webapp.war

[amitadmin@docker01 ~]$

[root@docker01 ~]# cat /etc/group | grep docker

docker:x:992:

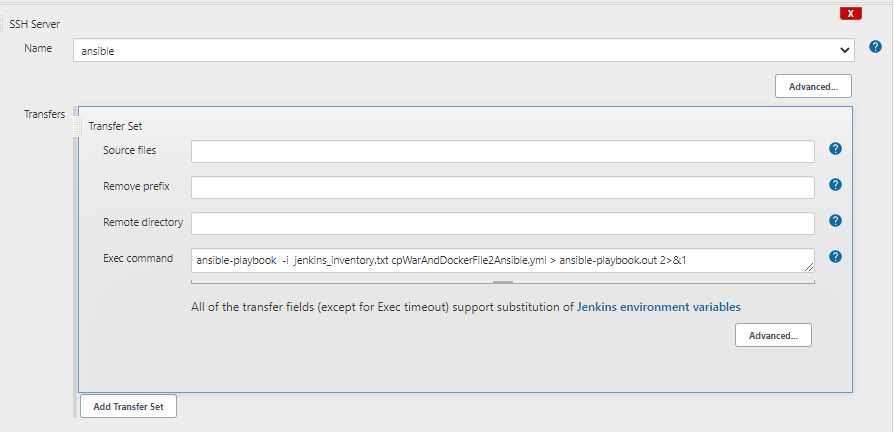
[root@docker01 ~]# usermod -aG docker amitadmin **(this is v important step)**

[root@docker01 ~]# cat /etc/group | grep docker

docker:x:992:amitadmin

[root@docker01 ~]#

In below screenshots in advance > check to enable detailed logging





**Installing and Enabling and disabling tomcat into Linux Server centos 7**

[root@docker01 ~]# rpm -qa | grep -i tomcat

tomcat-lib-7.0.76-16.el7\_9.noarch

tomcat-webapps-7.0.76-16.el7\_9.noarch

tomcat-el-2.2-api-7.0.76-16.el7\_9.noarch

tomcat-admin-webapps-7.0.76-16.el7\_9.noarch

tomcat-javadoc-7.0.76-16.el7\_9.noarch

tomcat-servlet-3.0-api-7.0.76-16.el7\_9.noarch

tomcat-jsp-2.2-api-7.0.76-16.el7\_9.noarch

tomcat-7.0.76-16.el7\_9.noarch

tomcat-docs-webapp-7.0.76-16.el7\_9.noarch

[root@docker01 ~]#

[root@docker01 ~]# systemctl status tomcat

● tomcat.service - Apache Tomcat Web Application Container

Loaded: loaded (/usr/lib/systemd/system/tomcat.service; disabled; vendor preset: disabled)

Active: active (running) since Tue 2021-01-05 13:50:50 UTC; 9s ago

Main PID: 22123 (java)

Tasks: 38

Memory: 109.9M

CGroup: /system.slice/tomcat.service

└─22123 /usr/lib/jvm/jre/bin/java -Djavax.sql.DataSource.Factory=org.apache.commons.dbcp.BasicDataSourceFact...

Jan 05 13:50:55 docker01 server[22123]: Jan 05, 2021 1:50:55 PM org.apache.catalina.startup.TldConfig execute

Jan 05 13:50:55 docker01 server[22123]: INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable... time.

Jan 05 13:50:55 docker01 server[22123]: Jan 05, 2021 1:50:55 PM org.apache.catalina.startup.HostConfig deployDirectory

Jan 05 13:50:55 docker01 server[22123]: INFO: Deployment of web application directory /var/lib/tomcat/webapps/doc...278 ms

Jan 05 13:50:55 docker01 server[22123]: Jan 05, 2021 1:50:55 PM org.apache.coyote.AbstractProtocol start

Jan 05 13:50:55 docker01 server[22123]: INFO: Starting ProtocolHandler ["http-bio-8080"]

Jan 05 13:50:55 docker01 server[22123]: Jan 05, 2021 1:50:55 PM org.apache.coyote.AbstractProtocol start

Jan 05 13:50:55 docker01 server[22123]: INFO: Starting ProtocolHandler ["ajp-bio-8009"]

Jan 05 13:50:55 docker01 server[22123]: Jan 05, 2021 1:50:55 PM org.apache.catalina.startup.Catalina start

Jan 05 13:50:55 docker01 server[22123]: INFO: Server startup in 4025 ms

Hint: Some lines were ellipsized, use -l to show in full.

[root@docker01 ~]#

[amitadmin@docker01 webapps]$ ls -ltr

total 12

drwxr-xr-x. 3 tomcat tomcat 4096 Jan 5 13:21 ROOT

drwxr-xr-x. 8 tomcat tomcat 127 Jan 5 13:21 examples

drwxr-xr-x. 5 tomcat tomcat 86 Jan 5 13:21 sample

drwxr-xr-x. 5 root tomcat 87 Jan 5 13:21 host-manager

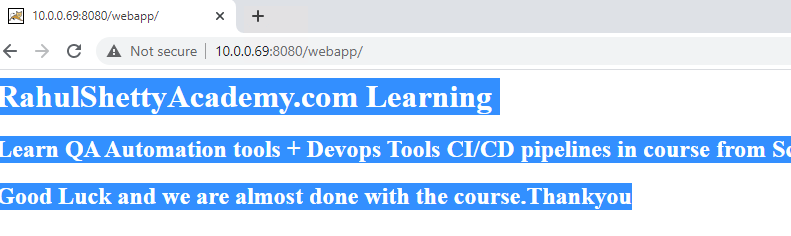
drwxr-xr-x. 5 root tomcat 103 Jan 5 13:21 manager

drwxr-xr-x. 14 root root 4096 Jan 5 13:22 docs

**-rwxrwxr-x. 1 tomcat tomcat 2554 Jan 5 13:43 webapp.war**

drwxr-xr-x. 4 tomcat tomcat 54 Jan 5 13:45 webapp

[amitadmin@docker01 webapps]$



[root@docker01 ~]# curl http://10.0.0.69:8080/webapp/

<h1> RahulShettyAcademy.com Learning</h1>

<h2> Learn QA Automation tools + Devops Tools CI/CD pipelines in course from Scratch</h2>

<h2> Good Luck and we are almost done with the course.Thankyou</h2>

[root@docker01 ~]#

[root@docker01 ~]# systemctl stop tomcat

[root@docker01 ~]# systemctl disable tomcat