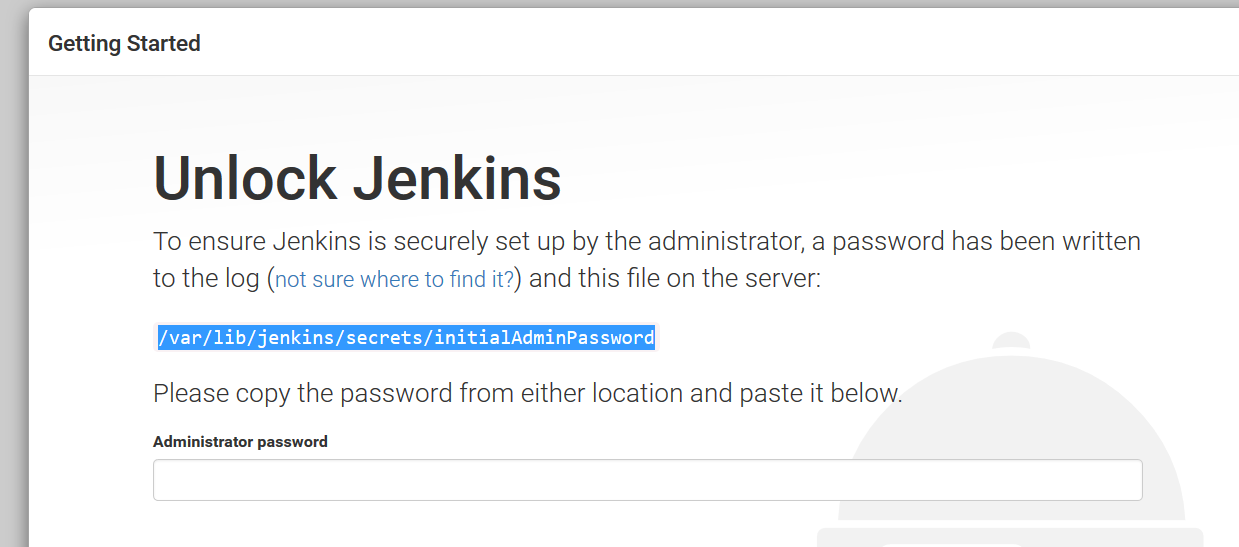
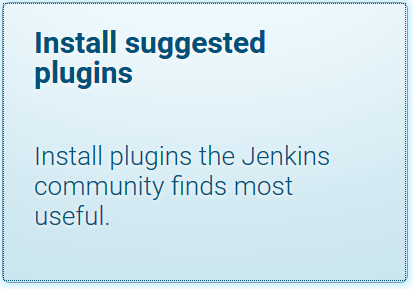
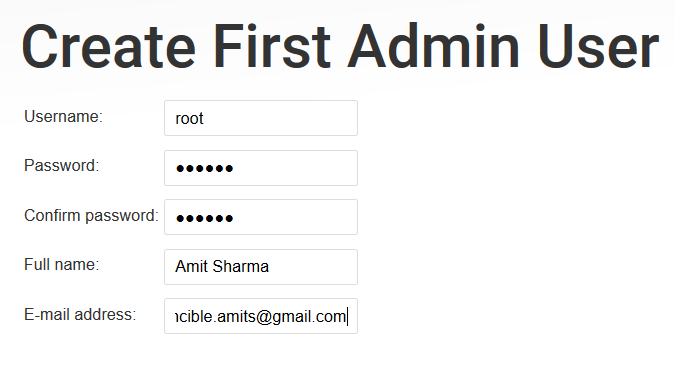
Ci-Cd

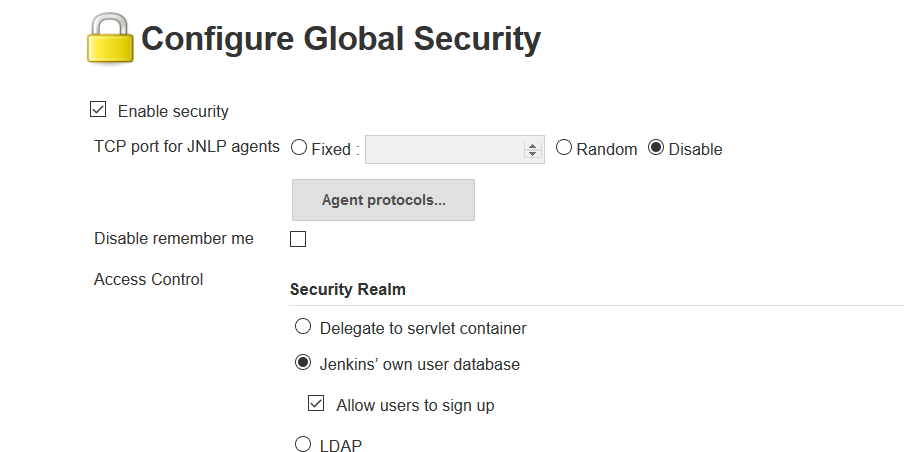
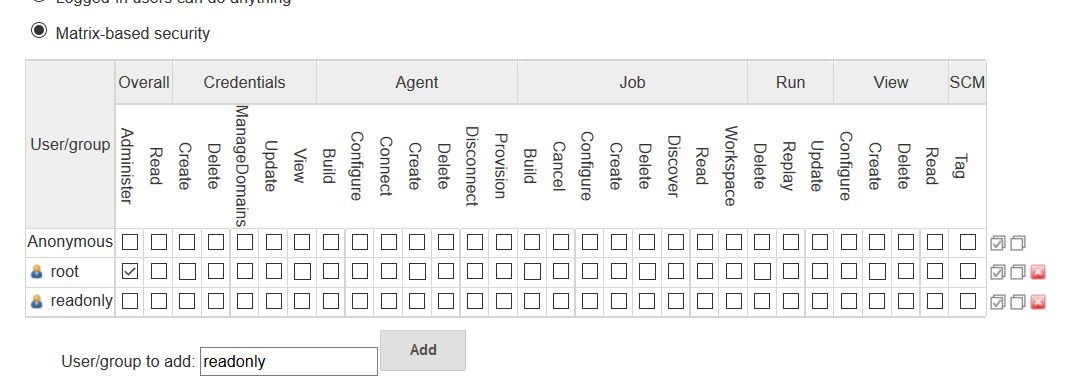
## Jenkins

Jenkins is the most popular **ci-cd (continues integration and continues deployment**) tool (written in java) and its open source.

### Jenkins installation steps on Ubuntu

1. Download the Jenkins key on the newly build Ubuntu server.
2. ***Wget*** [***https://pkg.jenkins.io/debian/jenkins-ci.org.key***](https://pkg.jenkins.io/debian/jenkins-ci.org.key)
3. This will download a Jenkins key to the apt cache which will be required for security purpose.
4. ***Apt-key add jenkins-ci.org.key (this will import the key certificate to jenkins server)***
5. Now list all the sources from where Jenkins should be downloaded.
6. ***Vim /etc/apt/sources.list.d/jenkins.list (and put the below text in this file)***
7. Deb https:pkg.jenkins.io/debian-stable binary/ (and save this file)
8. Now update the apt cache with command ***“apt-get update -y”***
9. **Now install the Jenkins** as below
10. ***Apt-get install Jenkins –y***
11. Once installed, check the ip with ifconfig and listening port with “netstat –nautpl | grep LISTEN”
12. For future upgrade do “apt-get upgrade Jenkins -y”
13. Now go to the server\_ip:port and launch Jenkins
14. Now it will ask for the one time password from below location /var/lib/jenkins/secrets/initialAdminPassword
15. 
16. Copy from above location and paste in the text bar and continue.
17. Now install the suggested plugin
18. 
19. Once the plugins are installed , create admin user (root/redhat)
20. 
21. Save and finish.
22. systemctl status Jenkins ( to check the Jenkins status manually on server)

### Jenkins best practice after setup

1. Got to Manage Jenkins🡪 configure global security and tick the checkbox to allow users to signup.
2. 
3. Now to define the role and access level, go to configure global security🡪 matrix based security 

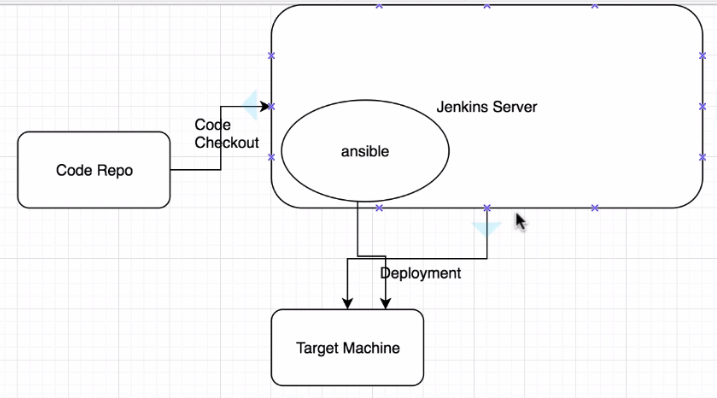
#### Below is the diagram that shows the relation between gitlab repo, Jenkins and nginx(web application server)

Here Code Repo – Gitlab server where our projects resides

Ansible – ansible small module is setup inside the Jenkins server to do the configuration management ( also to do the code checkout from code repo to Jenkins server)

Jenkins – This is the Ci-CD tool which will do the project build.

Target machine – this is nginx web application server where the code will be deployed from Jenkins server.



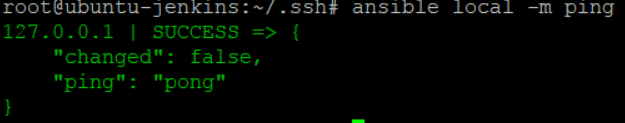
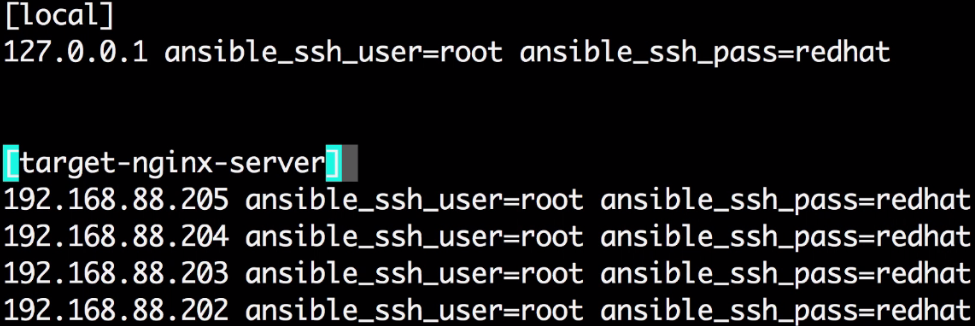
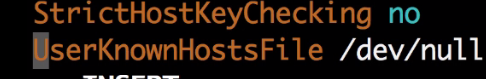
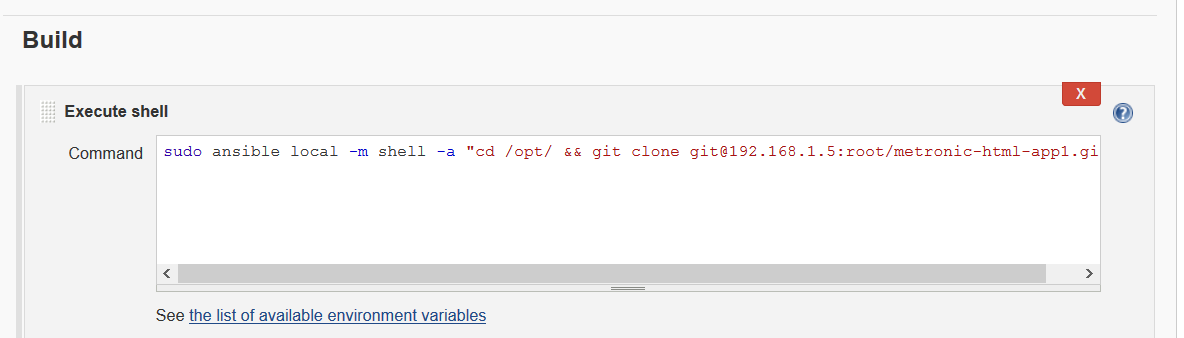
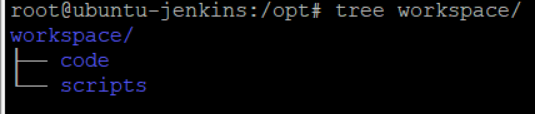
### Install nginx on new Ubuntu machine

1. Create a Ubuntu VM from scratch and update the server as usual (refer gitlab document on what update command to perform as best practice like apt-get update –y then apt-get dist-upgrade –y and then apt-get autoremove -y)
2. Run apt-get install nginx
3. Now check status sudo /etc/init.d/nginx status
4. The above can be used to start and stop nginx

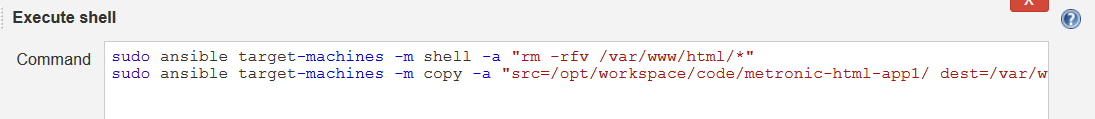
### How to install ansible on Jenkins server

1. First install the ansible dependency with below command(for centos u might need to install daily as it might go away with the session but for ubantu its automatic and always there)
2. apt-get install software-properties-common –y
3. now add the ansible repository
4. apt-add-repository ppa:ansible/ansible
5. now update the apt cache with apt-get update –y
6. now install with apt-get install ansible –y
7. once installation done, configure ansbile to make it work on Jenkins machine
8. vim /etc/ansible/ansible.cfg
9. Now disable the host key checking as below
10. 
11. Now tell ansible which servers it needs to connect to like localhost Jenkins server, nginx server etc.
12. Vim /etc/ansible/hosts (clear all the previous defaults contents)
13. Put the below text in above file
14. [local]

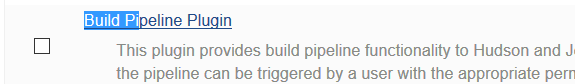
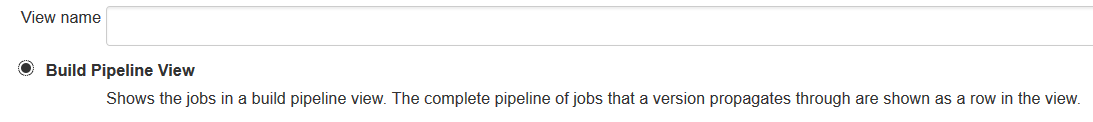
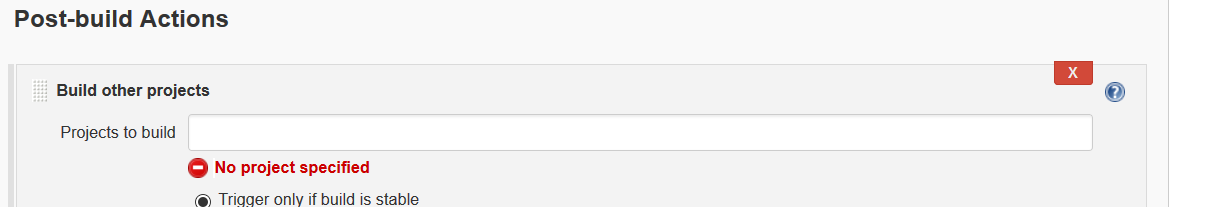
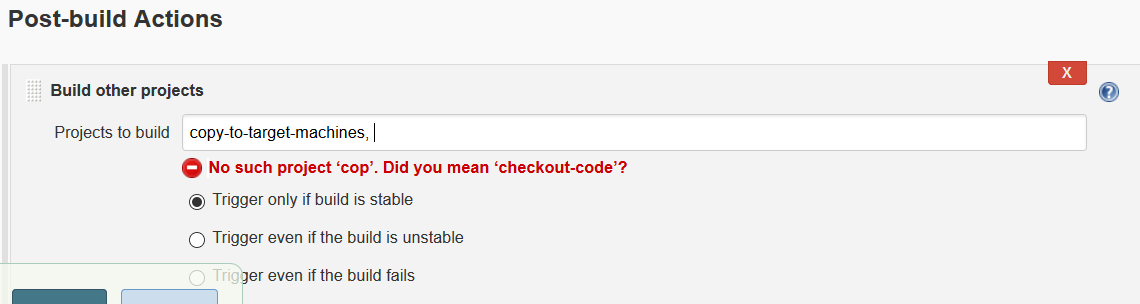
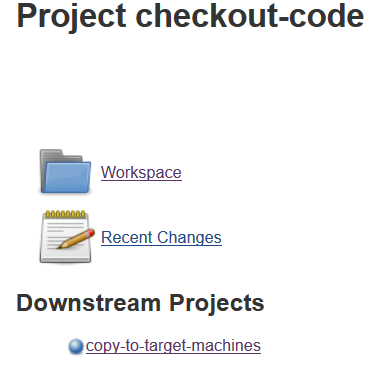
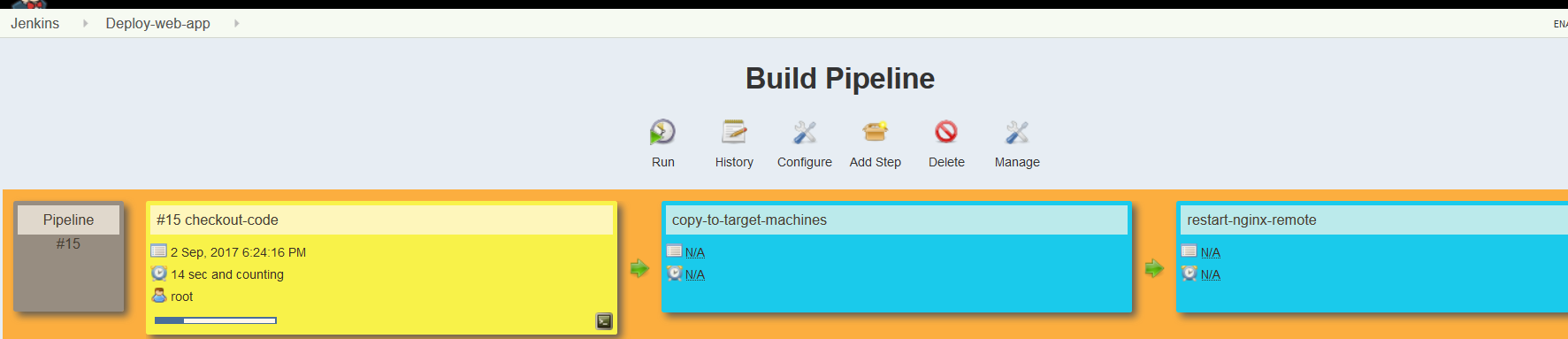
127.0.0.1 ansible\_ssh\_user=root ansible\_ssh\_pass=redhat

1. First check the connectivity with the local host like
2. ansible local -m ping
3. 
4. Now give all the nginx server as below(these are userid and password of the nginx server)
5. 
6. Make sure that the jenkins server ssh key is added on gitlab server from Jenkins server.
7. If build fails saying unable to connect, then try to clone the project from Jenkins server using normal git clone command. Also pay attention to the git remote –v output to check the target url.
8. Set *StrictHostKeyChecking no* in the cat /etc/ssh/ssh\_config by uncommenting these two lines
9. 
10. Now set the following to checkout the code on jenkin server from gitlab server
11. Go to project -> config 🡪 build 🡪 execute shell
12. 
13. And hit the build and check the console.
14. As best practice on the Jenkins server under /opt directory create a workspace and other directories like below
15. 

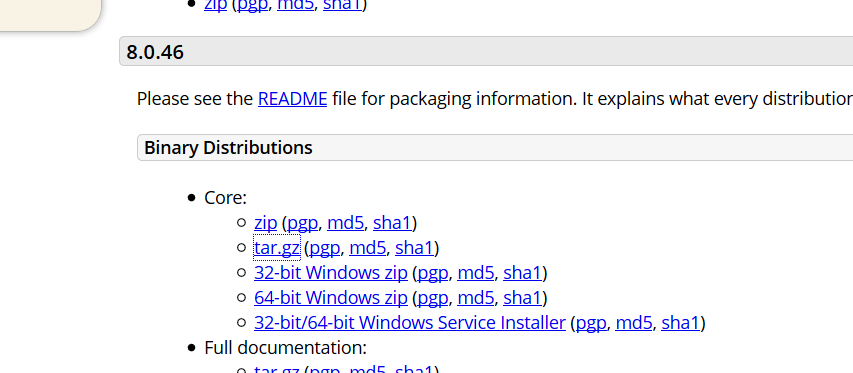
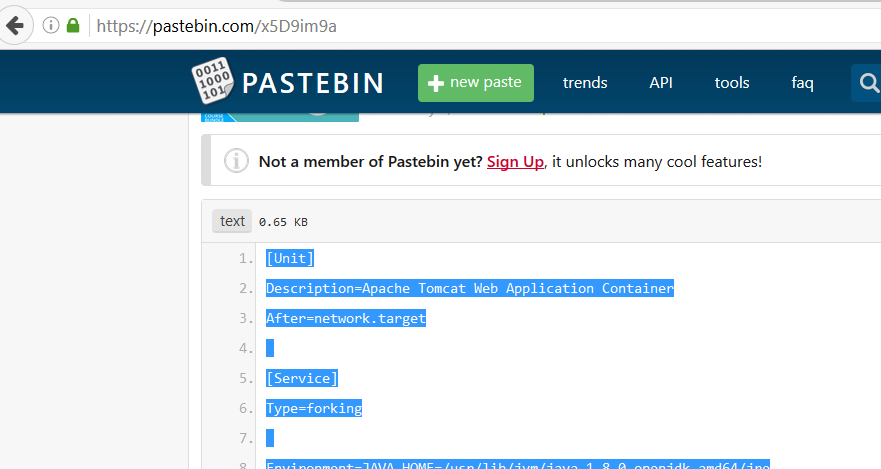
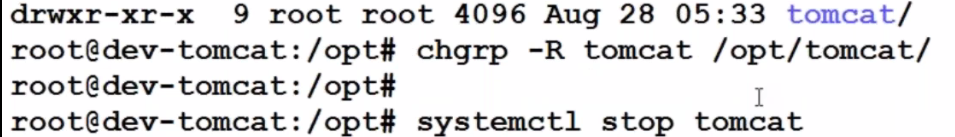
#### Now Deploy the app on the target machines(here njinx server)

1. Go to the target machines app server web directory /var/www/html
2. Now go to the project [copy-to-target-machines](http://192.168.1.11:8080/job/copy-to-target-machines/) configuration
3. Build 🡪 Execute shell and set the following
4. sudo ansible target-machines -m shell -a "rm -rfv /var/www/html/\*"
5. sudo ansible target-machines -m copy -a "src=/opt/workspace/code/metronic-html-app1/ dest=/var/www/html"
6. 
7. Now build it

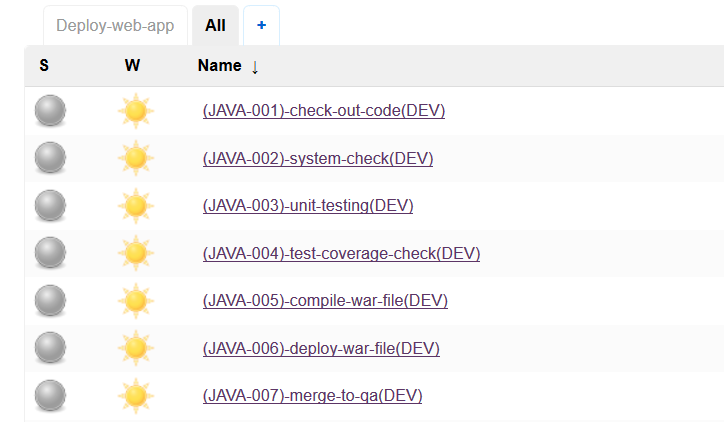
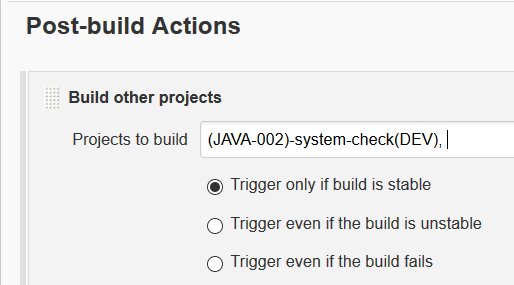
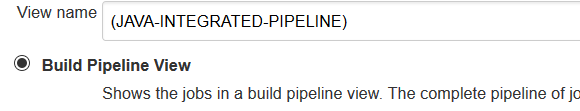
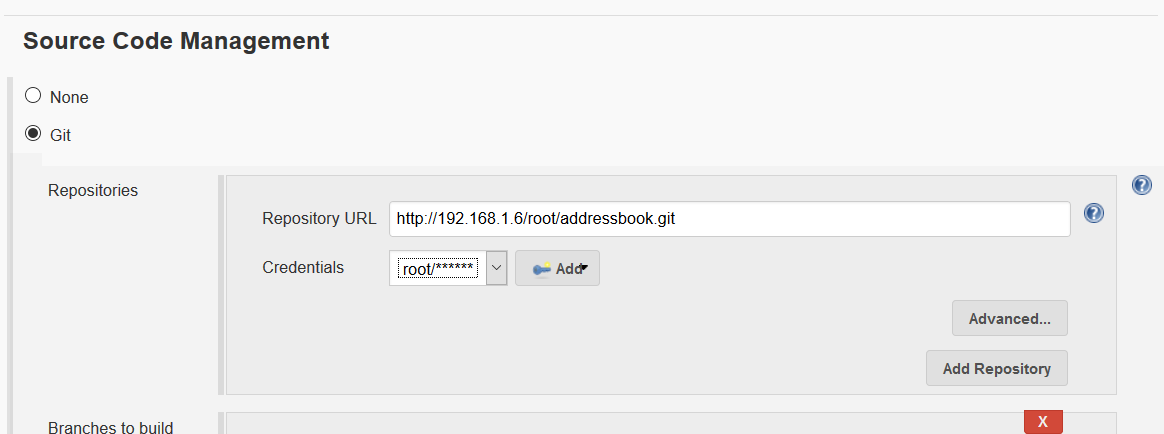
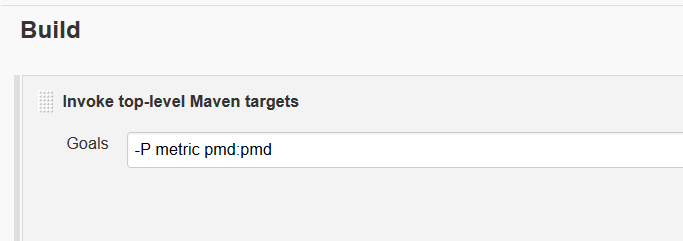
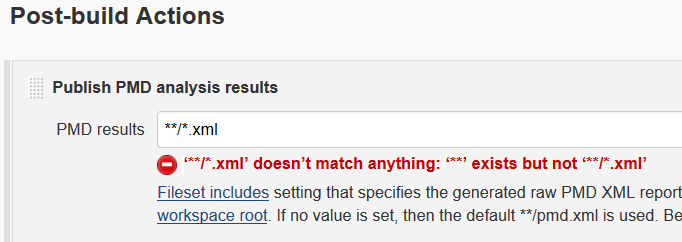
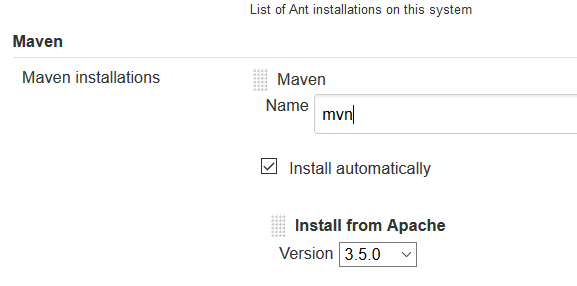
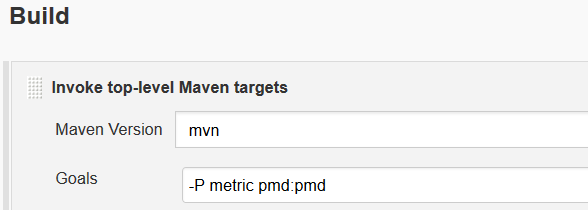
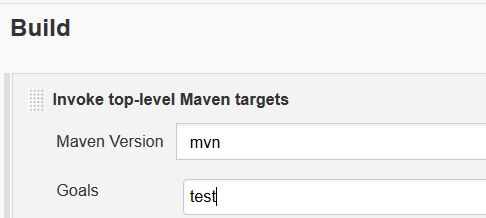
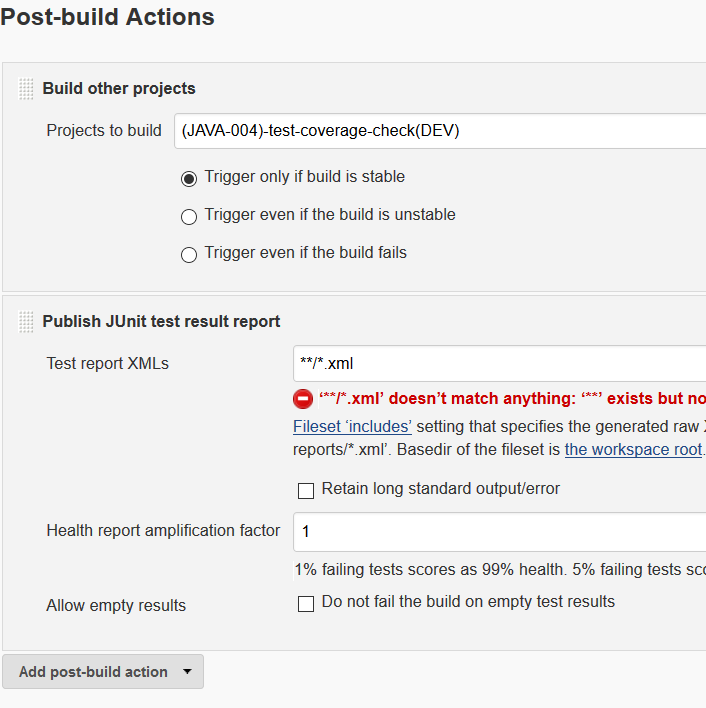
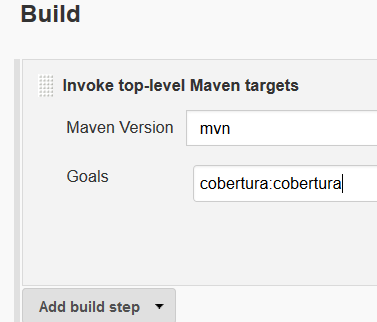
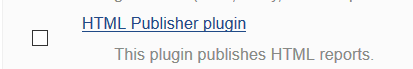
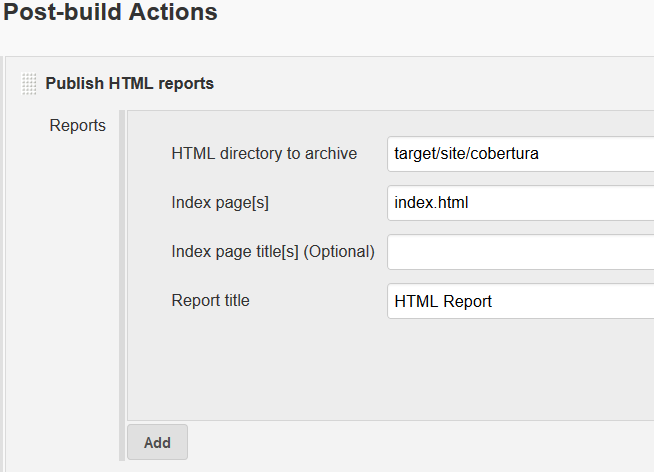
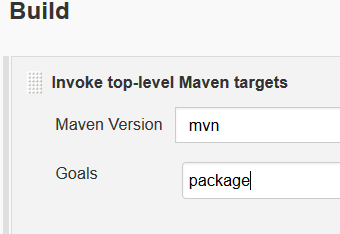
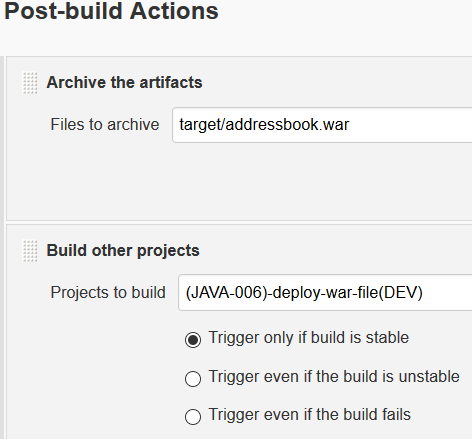
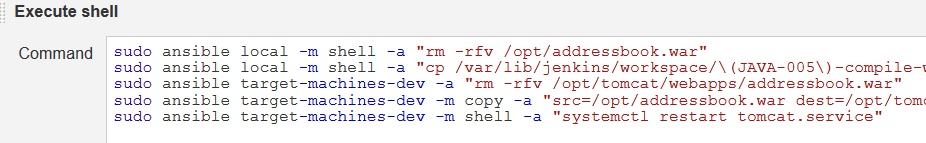
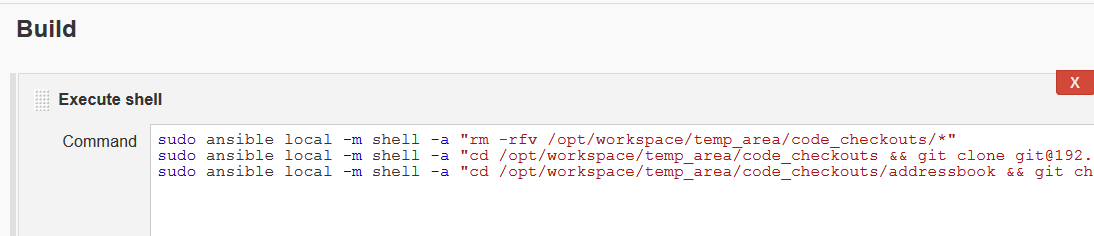
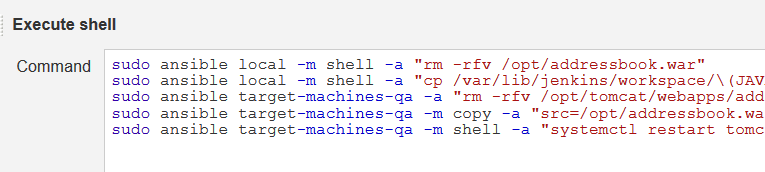
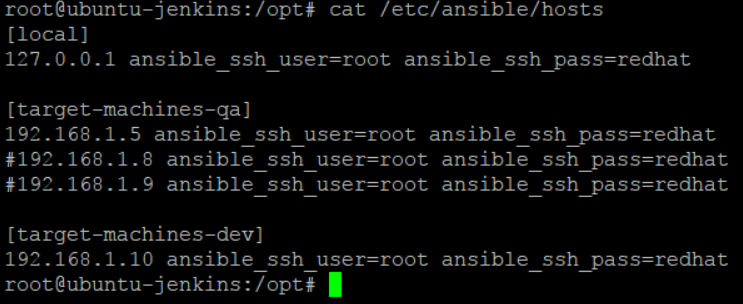
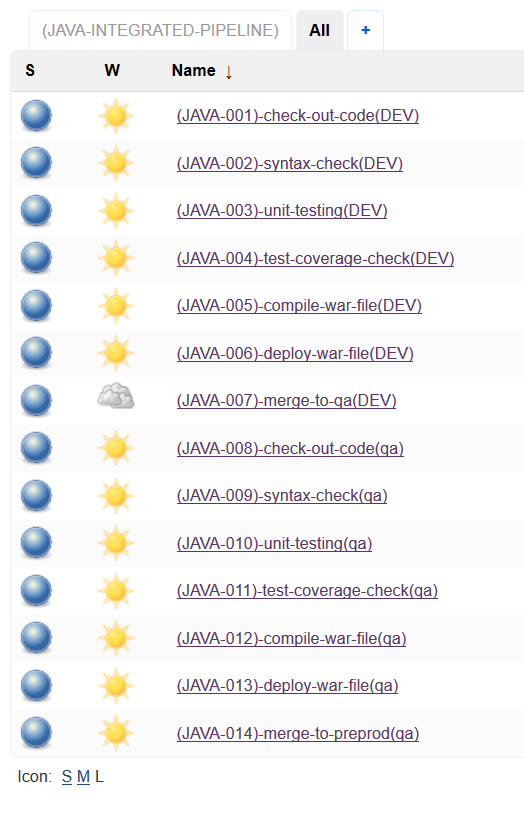
#### Building the project pipeline

1. Go to Manage Jenkins🡪Manage plugins🡪 advance and search for build pipeline plugins
2. 
3. Click checkbox and download and install after restart.
4. 
5. Once downloaded , check the box restart Jenkins and Jenkins will restart
6. Now build the pipeline after clicking on + button
7. 
8. Now go to the first build (check-out-code) and click configure.
9. Now go to end to make a upstream-downstream build connectivity
10. 
11. Now give next build name and save
12. 
13. Now this will list as downstream projects
14. 
15. Do the same with remain build to create a build chain.
16. Now run the build pipeline
17. 

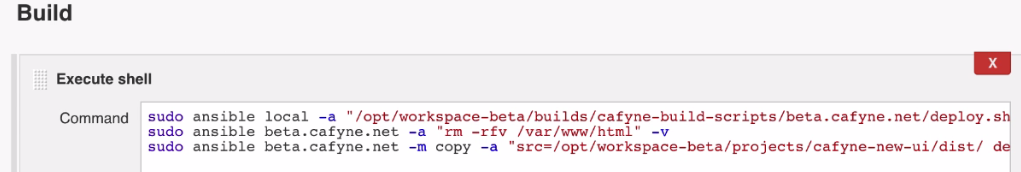
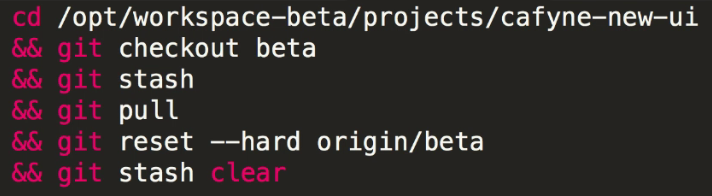
#### Install Tomcat on target servers

1. The same njinx machines can be used to install tomcat (just stop the tomcat service)
2. First install the latest jdk on machine (apt-get install default-jdk –y)
3. Now as best practice before installing tomcat, create tomcat group by groupadd tomcat
4. Now create a tomcat user( with false shell means it cant login and opt/tomcat home directory)
5. useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
6. install tomcat in /opt/tomcat
7. install it with the help of binary file available on tomcat site
8. 
9. Copy the link for this tar.gz file and do below.
10. Wget <http://www-eu.apache.org/dist/tomcat/tomcat-8/v8.0.46/bin/apache-tomcat-8.0.46.tar.gz>
11. Unzip it tar -xvzf apache-tomcat-8.0.46.tar.gz -C /opt/tomcat/ --strip-components=1
12. The above command will force it to unzip it in single /opt/tomcat dir without creating subfolders.
13. Change few permissions like below.
14. chmod -R g+r conf/
15. chmod g+x conf/
16. chown -R tomcat webapps/ work/ temp/ logs/
17. Now copy the tomcat startup script from internet (<https://pastebin.com/x5D9im9a>)
18. Paste this script in file vim /etc/systemd/system/tomcat.service
19. 
20. Now reload the system deamon with command systemctl daemon-reload
21. Now start tomcat and check the status systemctl start tomcat.
22. Now take the ip check on browser on port 8080.
23. If does not work then try following
24. systemctl enable tomcat
25. check log /opt/tomcat/logs/ catalina.out
26. also group change in tomcat directory required. Do the following
27. 

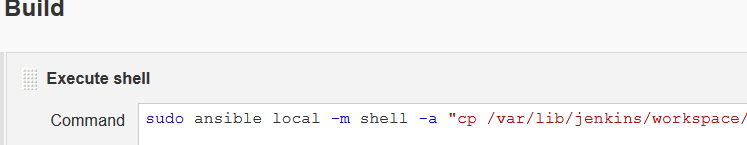
#### Creating project pipeline for multiple environment

1. Create free style new items in Jenkins like below
2. 
3. Now connect these builds to one another so that they can kick-off one another.
4. Click on the first build and configure like below
5. 
6. Above, build 1 will kick-off 2 and 2 will 3 and so on.
7. Do the same build for other build as well
8. Now Create a pipeline for these interconnected build
9. 
10. Now go to first code-check-out build and give the git repo http URL
11. 
12. Now build this project to check if the code checkout is successful or not.
13. Now move to next build which is code syntax check. For this pmd plugin will be required.
14. Go to manage Jenkins🡪 manage plugins🡪 Available🡪 PMD plugin
15. Now install it without restart.
16. Now go to syntax check project🡪 config and do like below
17. 
18. Now go to post build action🡪 publish PMD analysis and save it.
19. 
20. Now install the maven binary
21. Go Manage Jenkins🡪 global tool configuration
22. Add java jdk with credentials like below
23. 
24. Also add maven like below
25. 
26. Make sure maven is set like below before building the project
27. 
28. Now build the project.
29. In case of any failure make sure the git http url is specified in project config.
30. Now proceed for the Junit testing project
31. Go to Unit-testing🡪 config and give git url
32. 
33. Here test plugin is also required so go to manage Jenkins🡪 manage plugins
34. Search for below plugin and install it
35. 
36. Go to config and
37. 
38. Now build the project and check the unit test results.
39. Now move to test coverage check build and go to config and provide the git url with dev branch
40. 
41. Mention cobertura like above.
42. Now install the coberture plugin. Manage Jenkins🡪 Manage plugin🡪Available🡪Cobertura plugin and install.
43. 
44. Also install html publisher plugin
45. 
46. Now configure the test coverage project as below (this html report gives the details how much code is accessible by the cobertura)
47. 
48. Build other project should always come last.
49. Now restart Jenkins as best practice.
50. Now move to compile war file build and do the basic configuration of git and maven as above and specify as above
51. 
52. 
53. This Archive the artefact will show the war file on the project after the build as below.
54. 
55. Build this project to check if it works (after building it will show the above war file)
56. Now deploy the war project build. Set up the ansible script like below in config
57. *sudo ansible local -m shell -a "cp /var/lib/jenkins/workspace/\(JAVA-005\)-compile-war-file\(DEV\)/target/addressbook.war /opt/"*
58. *sudo ansible target-machines -a "rm -rfv /opt/tomcat/webapps/addressbook.war"*
59. *sudo ansible target-machines -m copy -a "src=/opt/addressbook.war dest=/opt/tomcat/webapps/"*
60. *sudo ansible target-machines -m shell -a "systemctl restart tomcat.service"*
61. 
62. Go to the Jenkins and tomcat server and check if above paths are correct or not.
63. Now make the config change in the last build of dev branch which is merge-to-qa branch.
64. *sudo ansible local -m shell -a "rm -rfv /opt/workspace/temp\_area/code\_checkouts/\*"*
65. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts && git clone git@192.168.1.6:root/addressbook.git"*
66. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts/addressbook && git checkout dev"*
67. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts/addressbook && git pull"*
68. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts/addressbook && git checkout qa"*
69. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts/addressbook && git pull"*
70. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts/addressbook && git merge dev"*
71. *sudo ansible local -m shell -a "cd /opt/workspace/temp\_area/code\_checkouts/addressbook && git push"*
72. The above commands will checkout and merge the dev changes in qa branch.
73. 
74. Now build this project to make sure it working fine.
75. Continue with the same pipeline for qa as well and merge it to the preprod and follow the same for preprod🡪 prod.
76. Just make sure in the deploy war file build, build config is set as below
77. 
78. For deploy war file **dev build, target machine should be dev and map the same in ansible host file as below.**
79. 
80. 
81. While make the new project copy it from the same old dev branch counterpart project and just change the branch and other setting.Follow the same logic as we did for dev branch pipeline(this time just for different branch and different target machines and make sure these target machines are added in ansible host file)

**How to refresh code with changes only rather than going for cloning of full project**

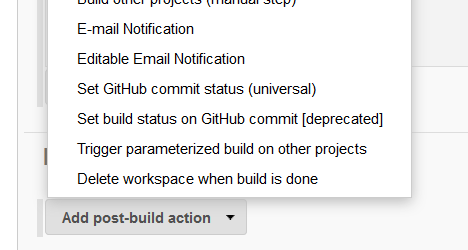
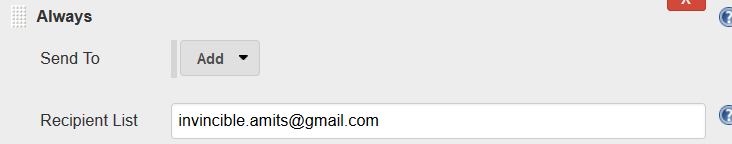
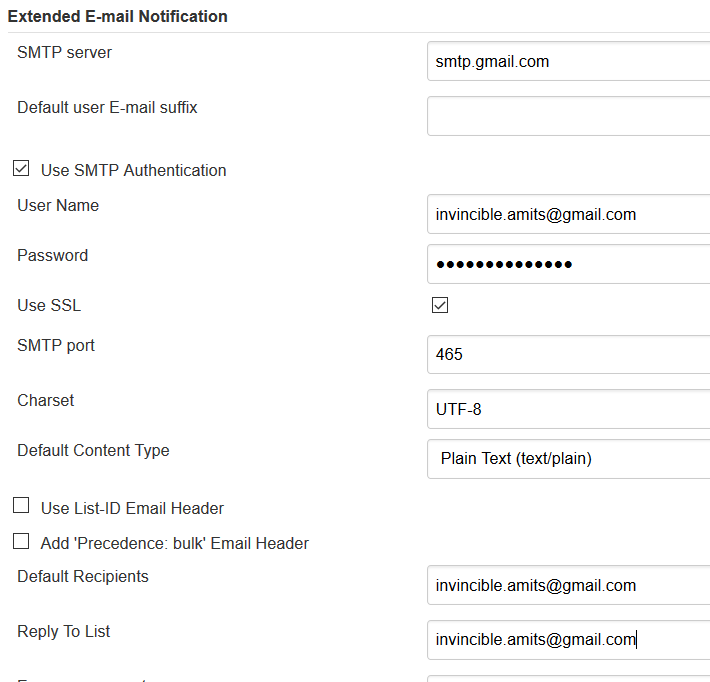
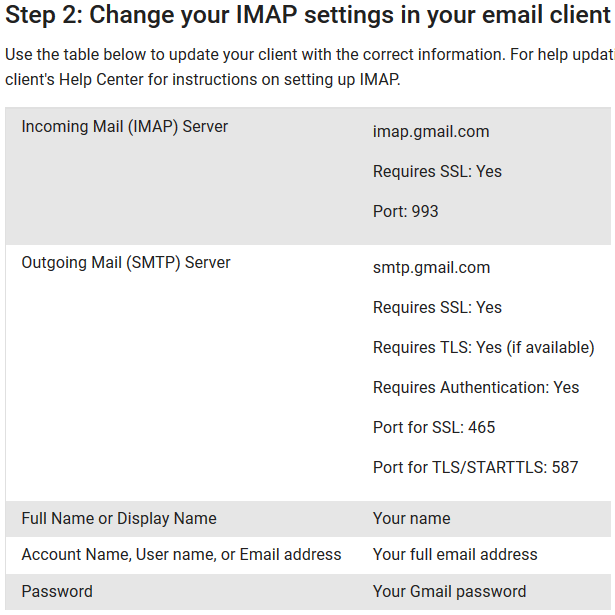
1. **Go to the config of the build and create a shell script like this(rather then do the git clone)**
2. 
3. **Now go to the above path on Jenkins servers and create a deploy.sh script like below.(also put –v at the end of script above)**
4. 
5. **Put the above test after replacing the branch name in deploy.sh script**

Create backup of the war file

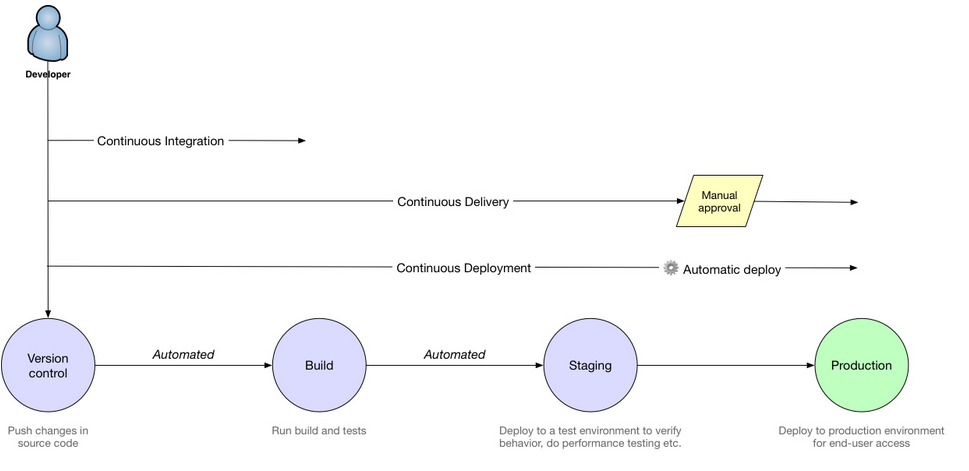
1. Create a separate job and place it before last build where we deploy the war to the target servers
2. 
3. Put the below text in this script
4. *sudo ansible local -m shell -a "cp /var/lib/jenkins/workspace/\(JAVA-005\)-compile-war-file\(DEV\)/target/addressbook.war /opt/addressbook-`date +%s`.war"*
5. Now bring this job in pipeline by setting the upstream and downstream projects.

## Email Integration with Jenkins

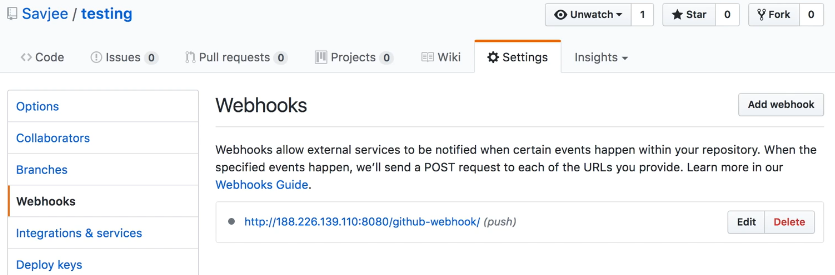
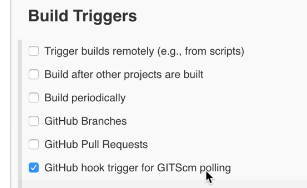
First install the below plugins in Jenkins

1. Manage Jenkins🡪 Manage plugins and search for email extension plugin and extended email notification plugin and install them (if not installed already)
2. Go to any build project and select editable email notification
3. 
4. 
5. Now we need to set the SMTP server details which Jenkins will use to send mail( we are using gmails smtp server here to send mails from Jenkins)
6. Manage Jenkins🡪 configure system
7. 
8. Used the below gmail settings
9. 

**Difference Between Continuous Integration, Continuous Deployment and Continuous Delivery**



**Polling GIT hub for every code commit and starting Jenkins build automatically**

1. Make sure webhook is configured inside the github project repository. This is Jenkins server URL/github-webhook
2. 
3. This is to tell t Jenkins that there is a change in github repository.
4. At the Jenkins side, install the github integration plugin and configure the job with github ULL details and check the below setting which will poll the web hook URL and trigger the build.
5. 

List down the most important plugins of Jenkins – very important

***Important***- In all the build projects, the post build action(build another project) should always be the last task to execute as next build should only be triggered once all other tasks are completed for a project ( if it is not last, then drag it to last position)

***Important*** *- The advantage of using ansible in Jenkins is because you can disable the host key checking in ansible which would help in automating the build process (as no prompt or password will be asked during check/build and deployment process)*

***Important point to learn*** *– host key checking can also be disabled in linux cat /etc/ssh/ssh\_config But it is not recommended as it a security compromise*