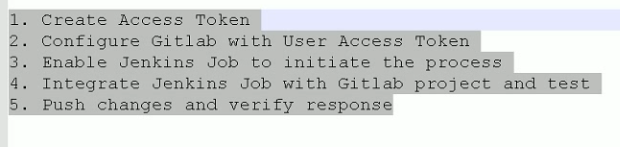
**Webhook:**

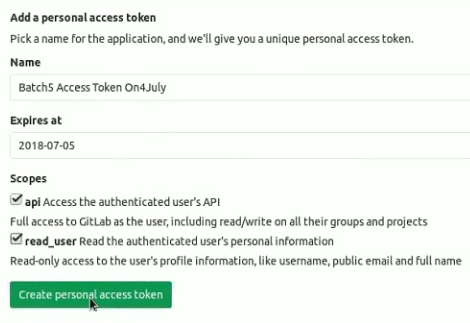
Whenever code is pushed to remote repo. We use webhook to build

Steps:



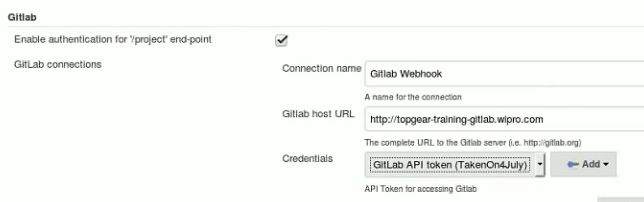
In GitLab, click on the right-side mark and open settings, go to access tokens

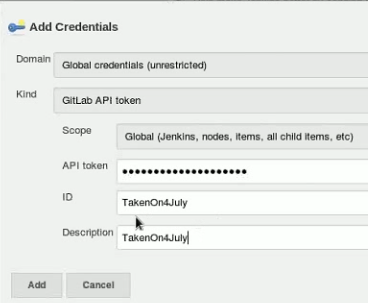
We can give any name and expires date and create personal access token



After this, it will generate a token. Copy that and save it

Then go to system settings, search for GitLab and configure that as below with API token





We can test the connection whether Jenkins is able to communicate with the GitLab

Commit any java file and create a job with git local repo

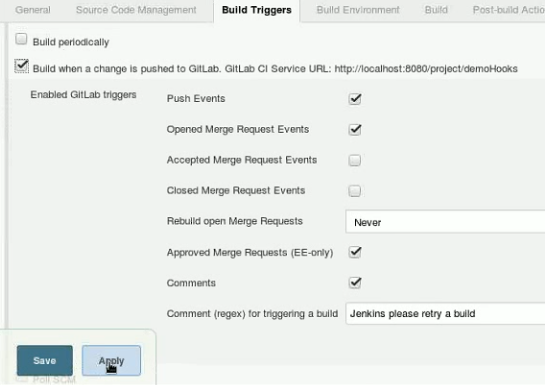
Under build step, go for shell and type javac welcome.java and java welcome

Build the job to ensure everything is fine with the job

Then create a new public project and push to remote repo

Select below under build triggers section

Then create a new public project and push to remote repo

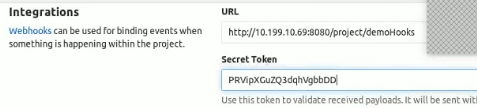


Now, in GitLab, select the setting options in left hand side and go to integration settings

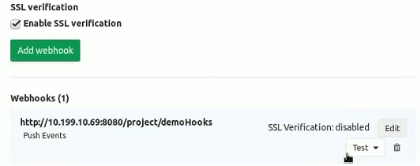
And we must give the URL of Jenkins job. In the link, modify it as project if it is mentioned as job

Better we copy it from build triggers settings in job as above

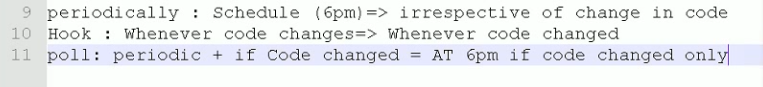
And enter the token id here. And under trigger options, let it be only push, uncheck the enable SSL and click on add webhook

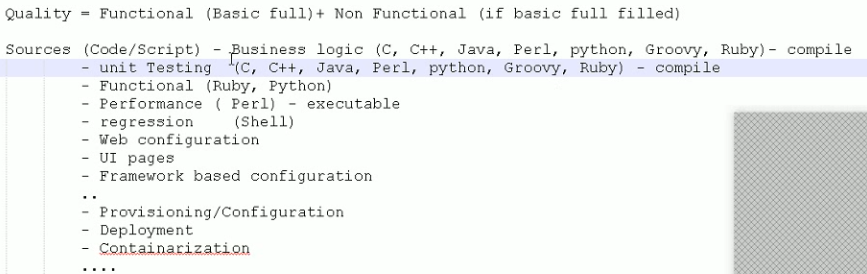


After adding, we can see the link below. If we want to change the IP address, we can change it by using edit option



Click on test and push the event. After this we can be able to see the HTTP 200 response on the top

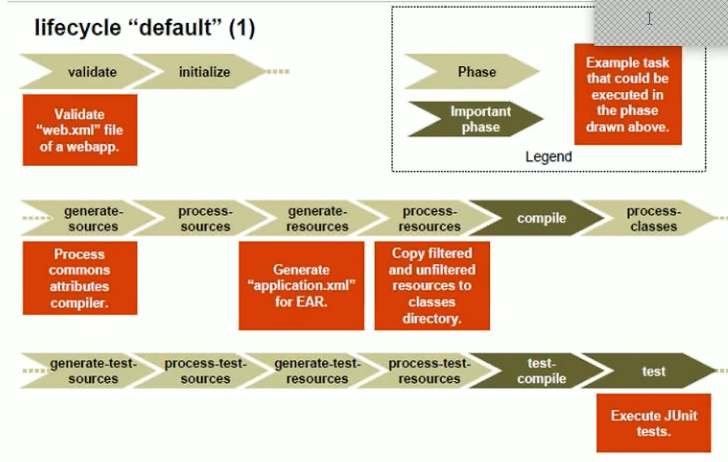


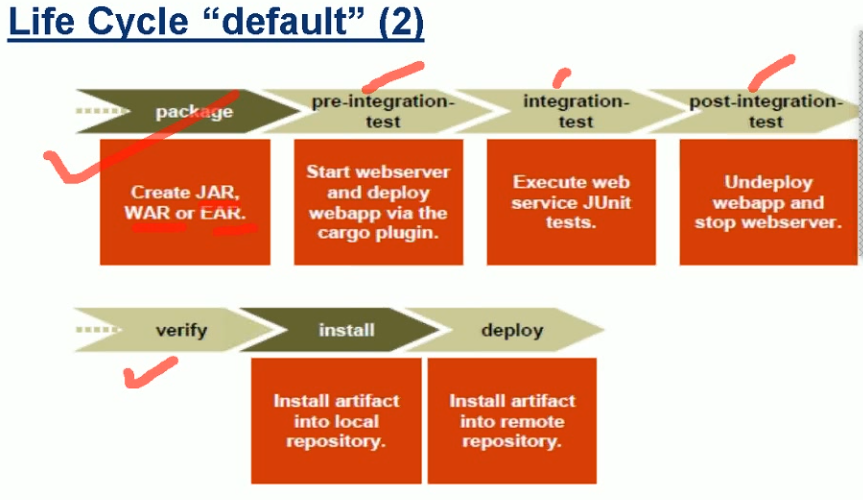


Source may come from any interest. Sometimes we may not need to compile

If compilation is success, then only we can go for testing

Maven is not enforced to follow the project structure. It is suggested to follow



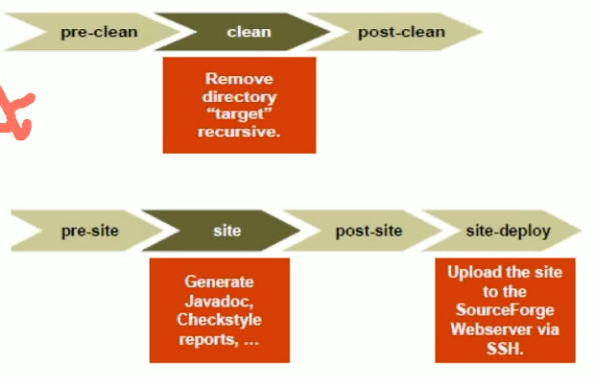


If we are going to perform functional testing, then the war file needs to be deployed in test server. This is called pre-integration testing

Then we can do testing with junit

After testing, we can stop web server. Then verify means reporting

If we are not following standard structure, we need to specify that in pom.xml, maven will take care of all these things



It is always good to perform clean build

we can configure pom.xml file to take backup before clean build and we can configure something else after the clean also

we can configure pre-site and post site also

SonarQube integration will be used in site task