**DLP Product build and Deployment:**

a) DLP Personal Loan:

Dev

Mysql- 192.168.0.241

Postgres - 192.168.0.198

QA

Mysql- 192.168.0.242

Postgres- 192.168.0.152

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

b) DLP Business Loan:

Dev

Mysql: 192.168.0.244

postgres: 192.168.0.243

QA

Mysql:

Postgres

For above environments Jenkins is configured to do the deployments, development team can trigger the build.

We are using docker compose in all the Dev and QA deployments, the minimum requirement to know the docker-compose file location, just in case to see the logs and check the issues manually

**See Docker Compose document information about the docker compose**

Docker compose contains the docker-compose.yml file which has definitions of all the services

All the deployments are happening via Jenkins console (we are not encouraging the users about the manual deployments

Jenkins deployment strategy:

In Jenkins we deploying to the servers based on the following steps.

1.Build the git project with maven tool which is done by Jenkins maven plugin

2.once build is done, copy the war/jar to the server with ssh plugin in Jenkins

3.Trigger the docker-compose build command so that it will build the image and start the services ssh plugin in Jenkins

**a) DLP Personal Loan:**

Dev

Mysql- 192.168.0.241

Postgres - 192.168.0.198

Jenkins Jobs Location: <http://192.168.0.174:8080/view/DLP-Dev-0.198/>

Docker-compsose file location: /home/deployer/web/UAT

These Jenkin jobs deploy to 192.168.0.198 and 0.241 servers simultaneously, Please check the each job for more information.

QA

Postgres- 192.168.0.152

Docker-compsose file location: /home/deployer/web/UAT-mysql

Jenkins Jobs Location: <http://192.168.0.174:8080/view/DLP-QA-152/>

These Jenkin jobs deploy to 192.168.0.152 , Please check the each job for more information

b) DLP Business Loan:

Dev

Mysql: 192.168.0.244

postgres: 192.168.0.243

Docker-compsose file location: /home/deployer/web/UAT

Docker-compsose file location: /home/deployer/web/UAT-mysql

Jenkins Jobs Location: <http://192.168.0.174:8080/view/DLP-BL-Dev-0.243%20and%200.244/>

These Jenkin jobs deploy to 192.168.0.243 and 0.244 servers simultaneously, Please check the each job for more information.

**Sonar:**

Jekins jobs location: <http://192.168.0.174:8080/view/Sonar/>

In these jobs, each project is build, run the sonar scan and share the reports to developers

Please check the each job for more information.

SonarQube server: <http://192.168.0.175/>

Location on the server: /opt/sonarqube/

Start the server:

su - sonar

(Sonar)$ cd /opt/sonarqube/bin/linux-x86-64

(Sonar)$./sonar start

To stop the server:

(Sonar)$./sonar stop

**Vulnerabilities scan for all the open source java dependencies:**

We are checking the opensource jar dependencies vulnerabilities with help of the OWASP depency check and sending the report to the developer,

Jenkins Jobs location:

<http://192.168.0.174:8080/view/Weekly_Reports_DLP/job/DLP_develop_Sofware_Volunerabilities/>

, Please check the each job for more information.

**DLP release and snapshot:**

View location on the Jenkins:

<http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/>

**1.Bre core tag creation:**

Following Jenkins job creates the tag by taking the developer input as the tag name, this will done when there is release

Please check the job for more information.

Jenkins job location: <http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/job/BRE_Core_develop_Tag/>

**2.DLP snapshot push:**

The following Jenkins job will push the maven snapshots and their dependencies to the apache archiva server to resolve the issues if some jars not found on the apache archiva server as the developer may face the issue while building the project on the local,

this will done when there is release

Please check the job for more information.

Jenkins job location: <http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/job/DLP-Snapshot/>

**3.Creating UI(frontend tag):**

Following Jenkins job creates the tag by taking the developer input as the tag name, this will done when there is release

Please check the job for more information.

Jenkins job location:<http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/job/DLP_UI_Tag/>

**4.DLP backend release:**

The following job will automate the process of maven release process for DLP backend project , which includes creating the release version, creating the tag, creating the next release version and increasing the SQL versions in the git as per the release version

this will be done when there is release

Please check the job for more information.

Jenkins job location :<http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/job/DLP-Release/>

**5.Generating the product release notes:**

The following Jenkins job automate the process of the Generating the release notes for both frontend and Backend jobs, once the specific version released

And also it updates the YouTrack as per the YouTrack ids in the release notes

Jenkins Jobs location:

<http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/job/DLP-Product_Relese_notes/>

Please check the job for more information.

**6. Gerrit sonar:**

The following Jenkins job will run every Gerrit commit to build and scans for the SonarQube quality checks, if any violations found it will commit back to Gerrit UI

We are doing this with help of SonarGerrit plugin in the Jekins, as per documentation it requires SonarQube server version < 7.7, that’s we are running SonarQube server 7.6 on 192.168.0.110 server

SonarQube server: <http://192.168.0.119:9000/sonar/>

Location on the server: /opt/sonarqube/

su - sonar

(Sonar)$ cd /opt/sonarqube/bin/linux-x86-64

(Sonar)$./sonar start

To stop the server:

(Sonar)$./sonar stop

Jenkins Jobs location:

<http://192.168.0.174:8080/view/DLP-Release%20and%20Snapshot/job/Gerrit_Sonar/>

Please check the job for more information.

Note:

In the DLP product, we are controlling the user commit message ie user has follow the specifi pattern in the git commit message.

Pattern: <YouTrack id>:commit message

For multiple YouTrack ids:

<YouTrack id1>,<YouTrack id2>:commit message

This is achieved by the git hook scripts in DLP, DLP\_UI projects

When user commits, there is gerrit hook commit-received will execute, it will further execute the project level hook script if present

These hook scripts check the YouTrack id exist or not from the commit message, it will validate and allow the commit

Once the reviewer reviews the commit,after merging the commit it going to call the Gerrit change-merge hook script, it will further excute project level git hook script if present , this hook update the specific YouTrack id from the commit

Gerrit hooks location on the server:

/home/gerrit2/review\_site/hooks

Git hooks location on the server

/home/gerrit2/review\_site/git/hooks/