AWS GeoManager Build/Deployment

1. Introduction

This document outlines the system requirements, the tools and configuration details for the GeoManager build and deployment infrastructure as well as the procedure for geomanager build, deployment and test automation for AWS environment.

2. Build and dependent server System Requirements

1. **SVN Server**
    a. Instance Type : M4 Large
    b. Root Volume : 50 GB
    c. EBS Volume : 250 GB (Mount Point : ***/usr/local/csvn***)
    d. Port 443 should open via NAC and stacks able to access the Server

2. **Jenkins Server - Master**
    a. Instance Type : M4 Large
    b. Root Volume : 50 GB
    c. Additional --- EBS Volume : 300 GB (Mount Point : **/usr/local/jenkinslave**)
    d. Port 8080 should open via NAC

3. **Artifactory Server**
    a. Instance Type : M4 Large
    b. Root Volume : 50 GB
    c. Port 8080 should open via NAC and This port should be accessible to jenkins build (Will provide the server details once created)

3. Tools Requirements

1. Jenkins (Version 2.79) - Build/Deployment
2. JFrog Artifactory (5.4.6) - Artifactory storage
3. AWS-S3 CLI - Package Storage
4. Shell/ANT/Maven/JDK 1.6/JDK1.8 - Compilation and restructure
5. FPM - Create RPM's
6. Ansible (Version 2.2) - Deployment package
7. SVN Edge (Version 5.2.2) - Source Code Maintenance

4. GeoManager Build/Deployment Tools

The following are the primary tools we are using for build and deployment

a. ***SVN***
b. ***Jenkins***
c. ***Package and Storage***

4a. SVN

Subversion which is often abbreviated as SVN, is a software versioning and revision control system distributed under an open source license. GeoManager source code is maintain on SVN. Infra team is managing and control this tool.

- **SVN Authentication**
- **SVN SourceCode**
- **SVN BackUp**

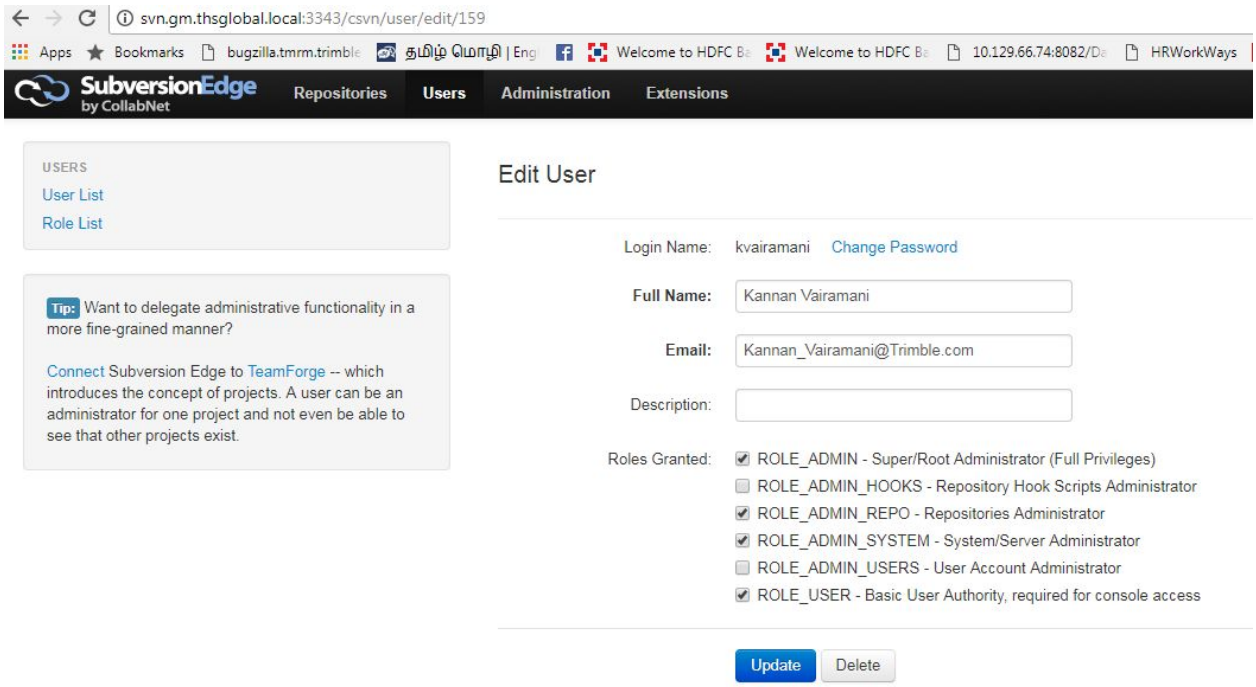
SVN Authentication

1. User Creation

SVN users are created via SVN Edge UI. Login name should be unique. Password must be a minimum of 8 character and must contain at least one capital letter, one small letter, number and special character. For Infrastructure team members (Kannan, Arun and Vijay) will be added in ROLE_ADMIN. For all other users need to be added in ROLE_USER.

Infra-Operation team have admin privilege to grant access to others,

URL : svn.gm.thsglobal.local:3343/csvn/login/auth (URL Will change after moved to AWS)



2. SVN access (cli, tortoisesvn)

SVN can be accessed via two ways Command Line Interface and Tortoise SVN (UI). Users will access SVN server via HTTPS port 443(SVN handled https certification internally). Users needs to use their respective username and password to commit their changes to SVN. All users have

read access . Incase they get 403 Forbidden error while commit their change, they need to contact Infrastructure team.

SVN Source Code

We are using the below two strategy to maintain the GeoManager source code in SVN

1. Trunk
2. Branch

1. Trunk :

The Production level code is maintained in Trunk. The Geomanager deliverable will use this source code for Release/Production deployments.
http://svn.gm.thsglobal.local/svn/geomanager/trunk/

Infrastructure Team will provide access to Trunk for the developers.Trunk access will be revoked when it is delivered to Production stack

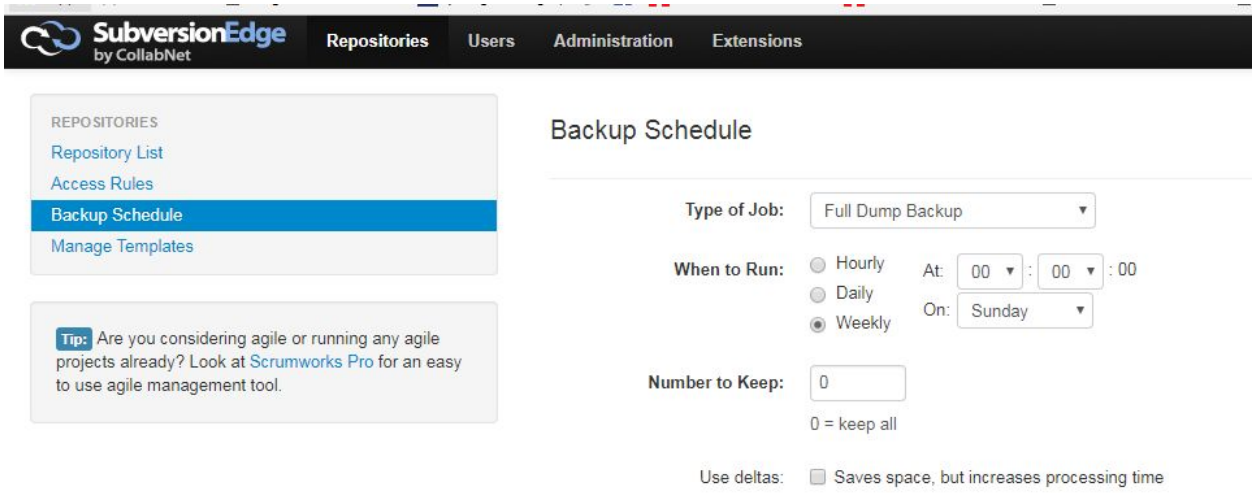
2. Branch :

Branch operation is used to create another line of development. It is useful when you want your development process to fork off into two different directions. For example, when you release version 1.0, you might want to create a branch so that development of 2.0 features can be kept separate from 1.0 bug-fixes.

The development team will raise branch request for their new development or any bug fixes to Infra team. Infra team will create branch for their development. Dev team use this branch until QA validation complete on QA stack. After that Dev team will send request to Infra team to open Trunk to check in their changes. Infra team will check the release tracker for any conflict between releases. Later open the trunk for their code check in on Trunk.

SVN Backup

We should take AWS snapshots volume daily in SVN server and 15 days retention period for the snapshots.
We can take automated backup using SVN edge and SVN dump will be stored in SVN server. We can use this dump to restore the SVN



4b. Jenkins

Jenkins is the build tool using to do build and deployment. Jenkins is a software that allows **continuous integration**. Jenkins will be installed on a server where the central build will take place.

Jenkins will checkout source code from SVN repository and do compilation, restructure and package in rpm format.

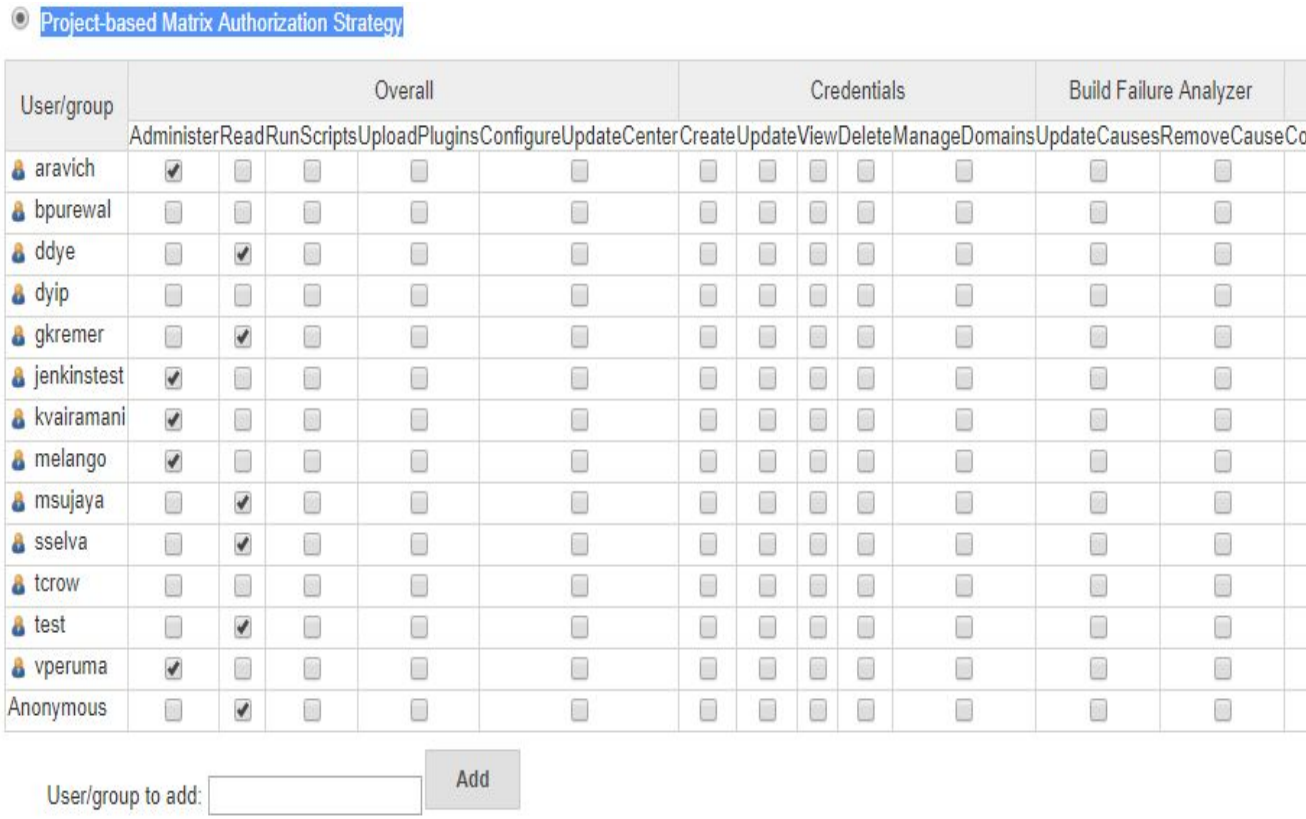
After packaging copy the package to AWS S3 repository. Infra team is managing and control this tool.

- Jenkins Authentication
- Jenkins Jobs
- Jenkins Backup

Jenkins Authentication

User Creation and Access

a. User creation is done via Jenkins UI or CLI. Restriction for each user is controlled by Project-based Matrix Authorization Strategy. Using this we can restrict a user by giving least privilege. Infra-Operation team have admin privilege to grant access to others



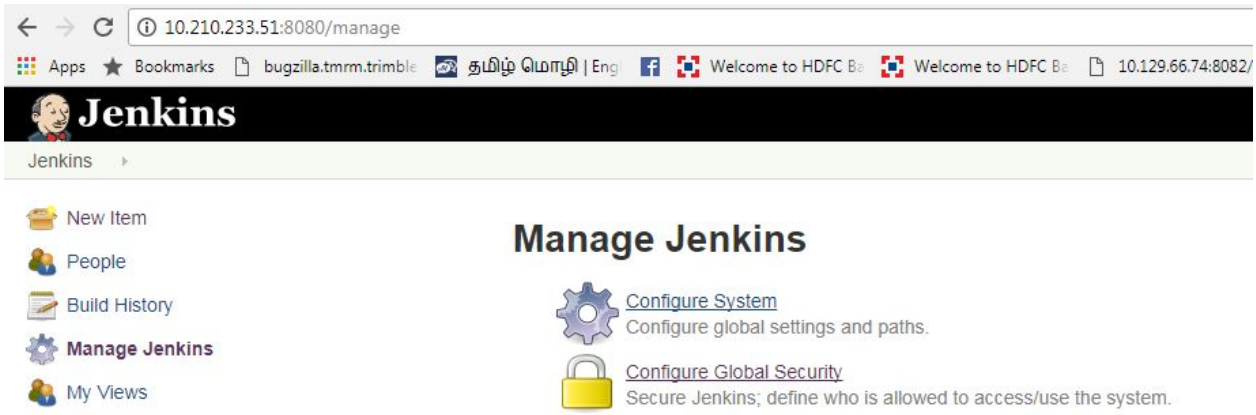
b. User account will be mainiting in AD. We will control/store the user account in AD server and we need to create 3 groups in AD server for Jenkins authentication(*read,readwrite,admin*.)

AD Group Name	Privilege
Jenkins-read (gm-jenkins-read)	Trigger Build Job
Jenkins-readwrite (gm-jenkins-readwrite)	Trigger Build Job/Create & Modify Job
Jenkins-admin (gm-jenkins-admin)	Full access

By default, the latest Jenkins (2.79) provides **Active Directory** authentication and not required to install any plugins.

Follow the below steps to enable the AD for jenkins

Step 1:
Click “Manage Jenkins” → Configure Global Security →



Step 2:

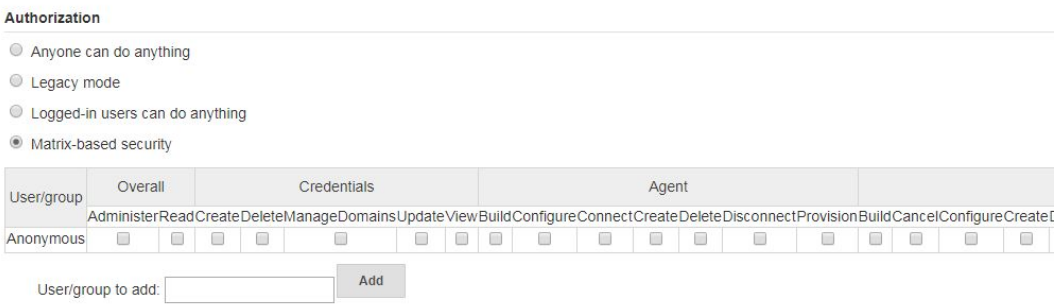
It will list the available authentication options, select the **Active Directory** option and provide the below details

Step 3:

After provide the details, click Test Domain to check the access

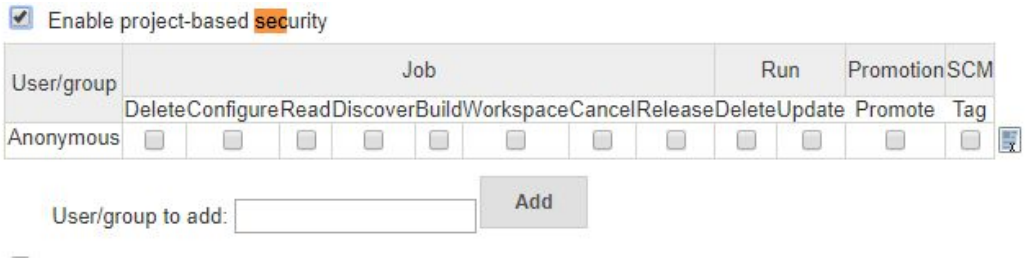
Step 4:

After test the Domain, we can control AD group permissions using **Matrix based security**.



Jenkins Job level authentication

We can control user access via job level too. We need to enable project-based security in Jenkins in each job.



Jenkins Jobs

We create a individual build job for each project. Whenever the development team will raise new request for branch for their new development or any bug fixes, we also create jenkins build job for that project. Infra team will also grant access to them to trigger and deploy build on QA environment.

Sample Build and Deploy Job

The below jenkins build job is for WM4.5 project,
http://sjc3-rl1-cimaster1.gm.thsglobal.local:8080/job/WM4_5_Linux/

This job will checkout the source code from below WM4.5 branch
http://10.122.147.20/svn/geomanager/branches/WM4.5
And do compilation, restructure and do required RPM package.

The below jenkins deploy job is for deploy the package to VG4 stacks
http://sjc3-rl1-cimaster1.gm.thsglobal.local:8080/job/VG4_Deploy_job/

We are using the **dashboard** account to do the deployment on the servers through jenkins.

Dev team use this branch until QA validation complete on QA stack. After that Dev team will send request to Infra team to open Trunk to check in their changes. Infra team will check the release tracker for any conflict between releases. Later open the trunk for their code check in on Trunk. The below job used to build the trunk code and copy the packages to respective AWS S3 repository

http://sjc3-rl1-cimaster1.gm.thsglobal.local:8080/job/Geomanager_Trunk_Linux/

Jenkins Backup

To take the backup of the Jenkin server using AWS snapshots. We should take AWS snapshots the ***root*** volume and 15 days retention period for the snapshots. Jenkins configuration file will take a backup /var/lib/jenkins and upload those files to AWS S3

4c. Package and Storage

Jenkins build and make package in RPM format

RPM Package Format :

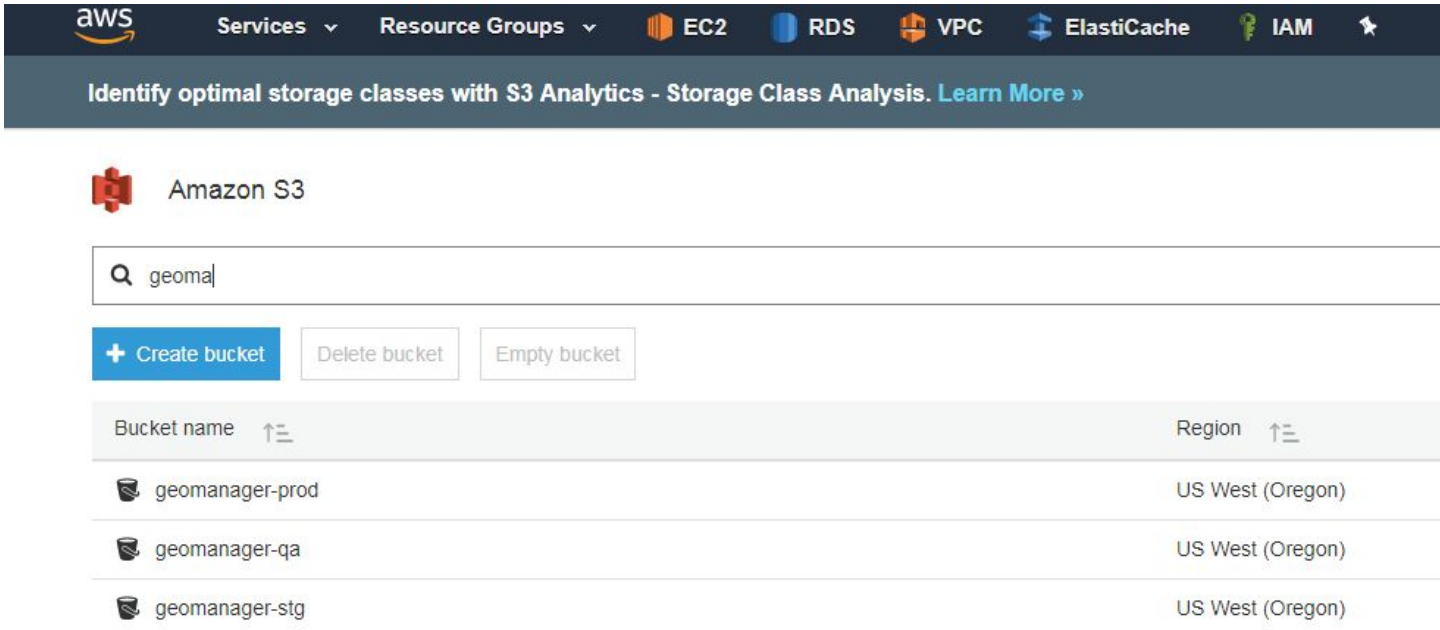
RPM Standard : TRMB-<componentname>-<SVNVersion>-<BuildNumber>.rpm

Ex. TRMB-quartzap-50192-4955.rpm

The created RPM package will be stored in AWS S3 repository. We are store the packages in three different repository which depends on the build we used.

Following are the three package stored location in AWS s3

- geomanager-qa (It contains the branch/trunk packages rpm's)
- geomanager-stg (It contains the trunk package's rpm)
- geomanager-prod (It contains the final trunk package rpm's for Production)



The project build job will copy the packages to ***Geomanager-qa*** AWS S3 repo. After QA validation completed on QA stack, the developer merge the code to trunk. The trunk build job will build package and copy the packages to ***Geomanager-qa , Geomanager-stg and Geomanager-prod*** AWS s3 repo using package copy job***.*** We maintain same packages on the QA, staging and production stacks(During Production deployment).

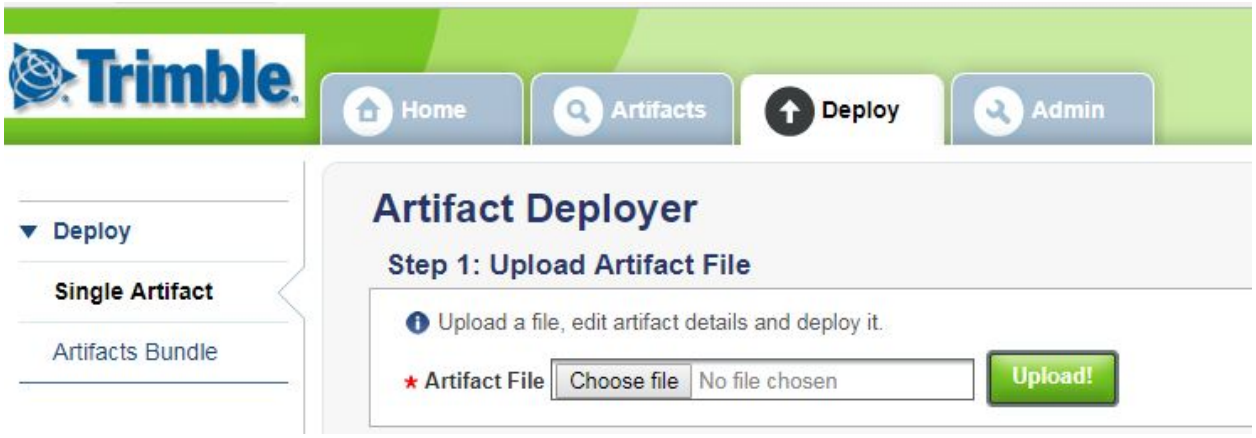
5.0 Artifactory Servers

This server is used to store all the internal and third party libraries. Infrastructure team maintain this server. Dev team will send request to infra team to upload required artifact files. These files may required either for compilation or we need to bundle it to packages.

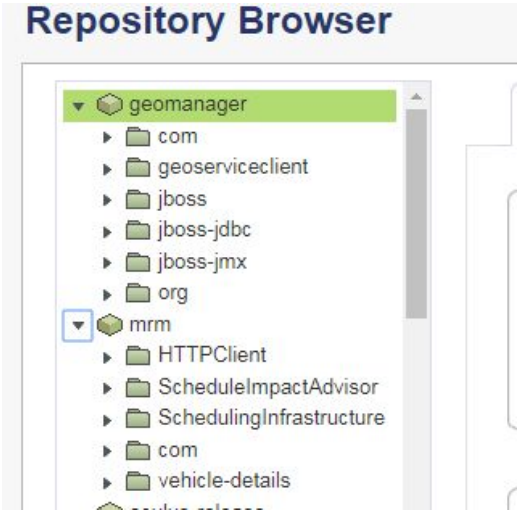
Existing Artifactory server
http://unixbuild.tmrm.trimble.com:8080/artifactory

Infra team will do the upload using below option

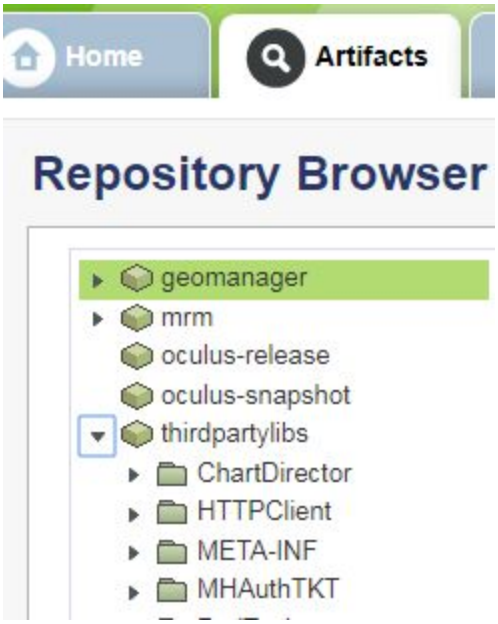
Click Deploy → Choose File → Click Upload



Internal libraries available in below path



Third Party libraries available in below path,



Artifactory Backup

To take the backup of the Artifacotry server using AWS snapshots. We should take AWS snapshots the ***root*** volume and 15 days retention period for the snapshots.

6.0 GeoManager Build/Deployment Workflow

The complete geomanager build/deployment workflow from development to Production.

https://drive.google.com/open?id=0BwGlBhsGfBdINGFXZnlFa0RhUGM

7.0 GeoManager Build/Deployment Categories

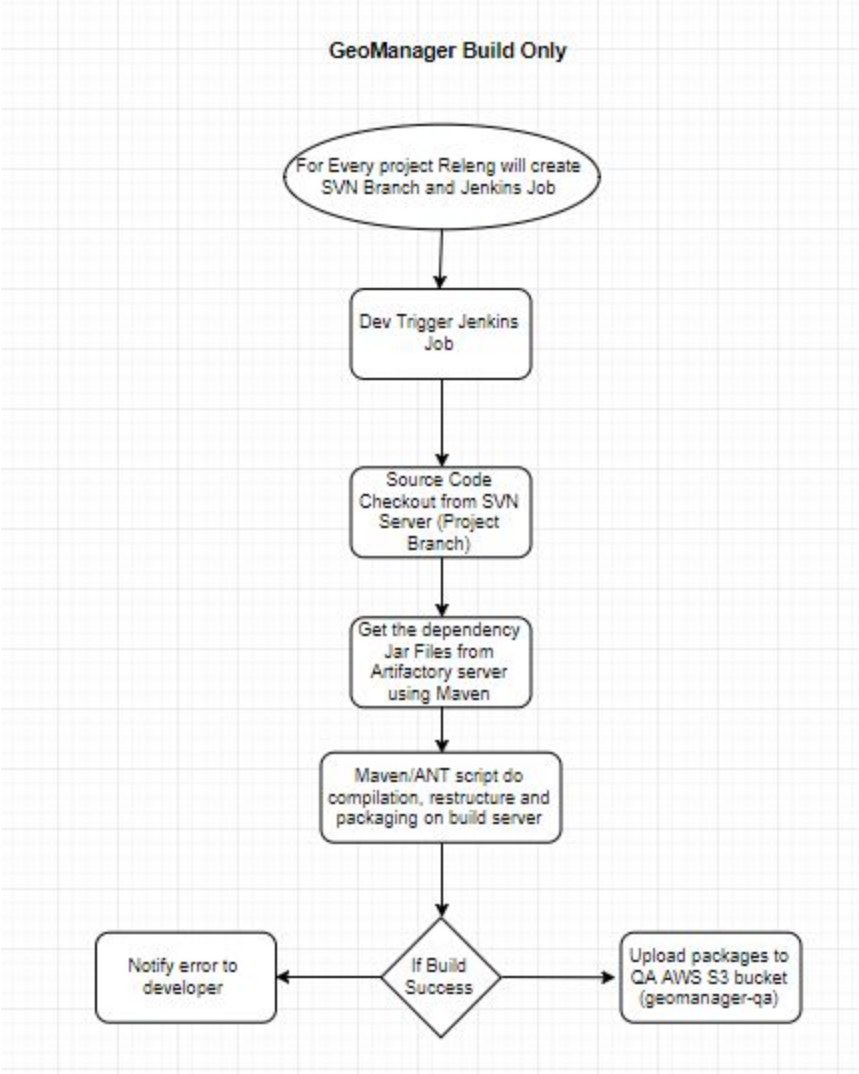
Following are different categories we are doing build and deployment for GeoManager.

A. GeoManager Build Only
B. GeoManager Deployment and Rollback - Need basis - QA
C. GeoManager Build and Deployment for QA and Dev (*CI/CD for QA*)
D. GeoManager Deployment and Rollback for Release and Production
E. GeoManager Test Automation

A. GeoManager Build

This process will checkout the source code from SVN and do compilation. After successful compilation, make a RPM packages and save it to AWS S3 repository. If any failures occur during the build, it will notify to the relevant developer.

Following flowchart gives workflow of GeoManager Build

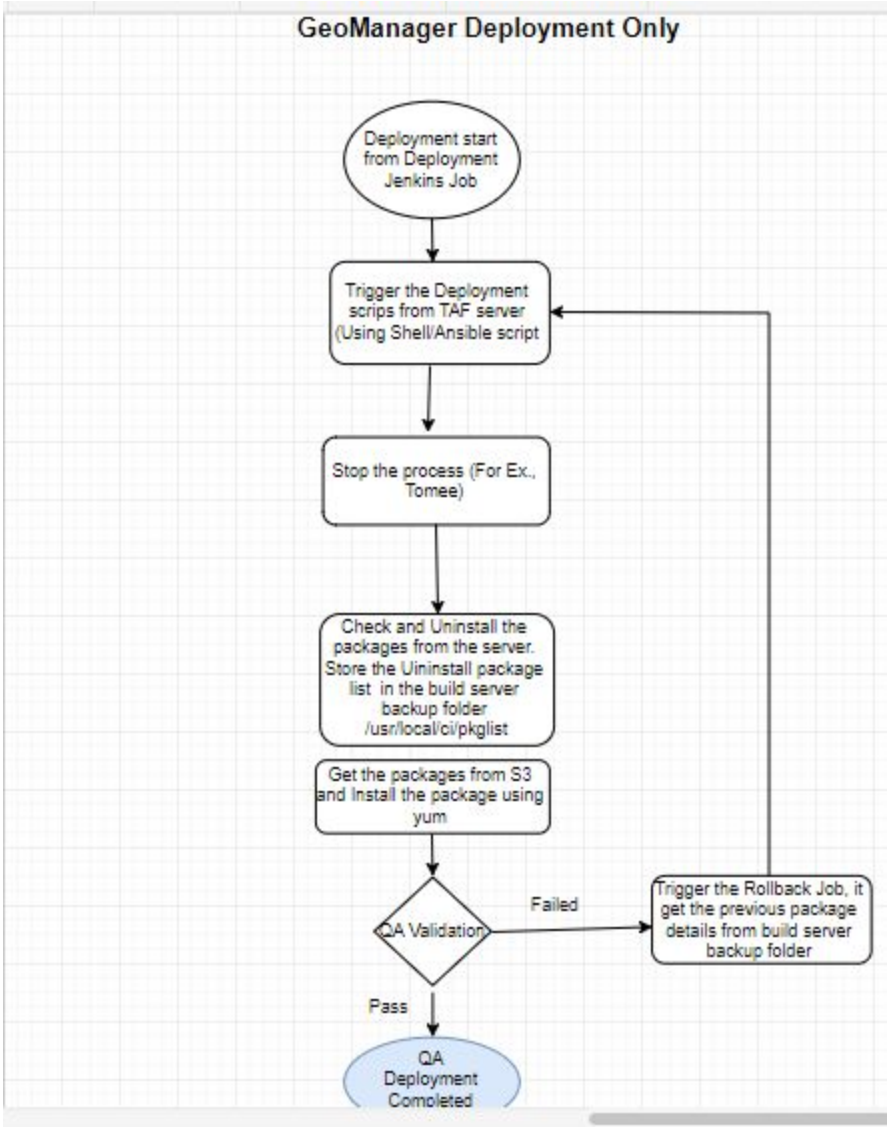


B. GeoManager Deployment and Rollback

This Jenkins job will trigger after successful completion of the build. Take all relevant packages from AWS-S3(*geomanager-qa*) and deploy it to respective stacks. **DEV/QA** stacks can do build/deployment by developer itself. If anything failed during QA validation, we can do the rollback immediately to previous version using the Rollback Job. We are maintaining each

deployment package details in ***/usr/local/ci/<stack>/<package>.list*** in jenkins server. We use this file to rollback whenever required.

Following flowchart gives workflow of Deployment and Rollback



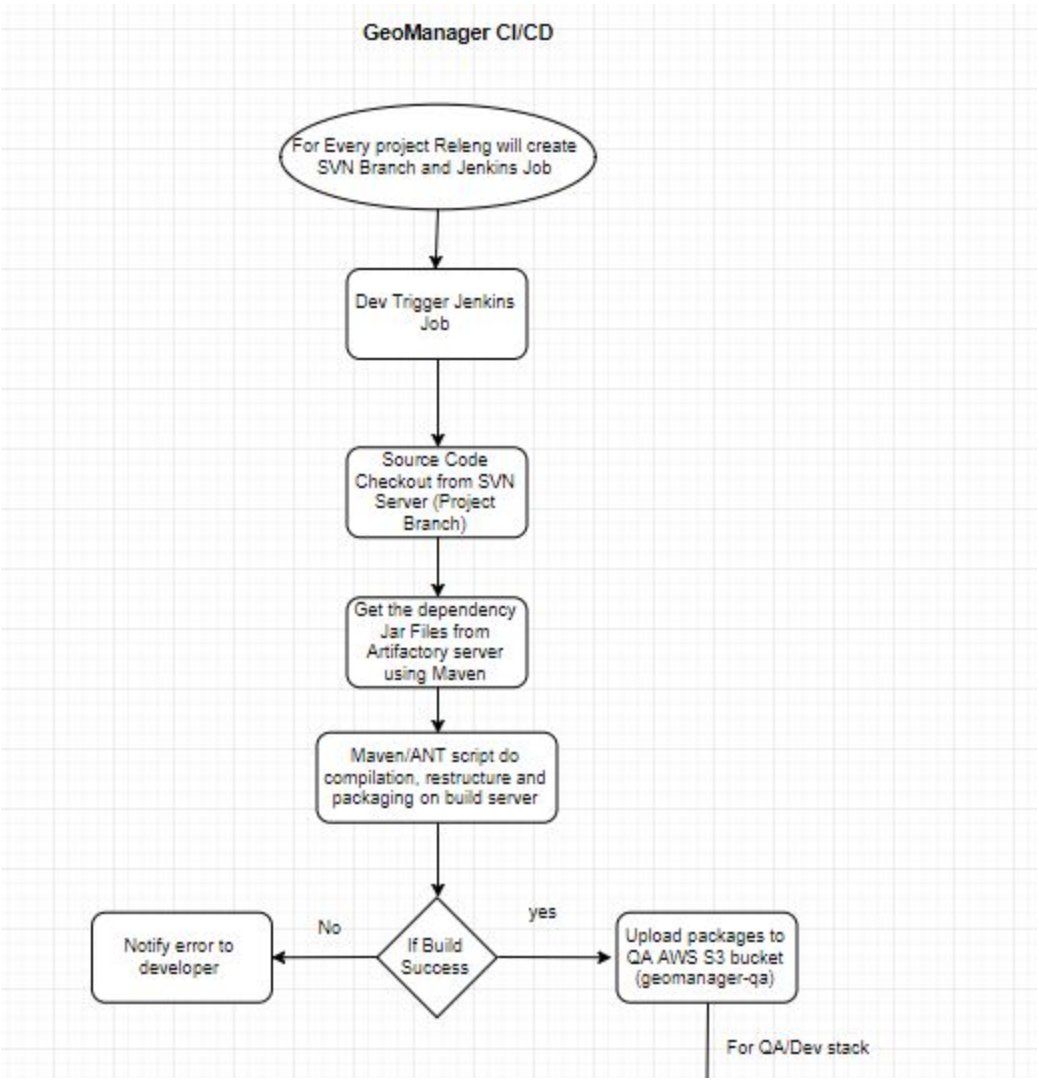
C. GeoManager Build and Deployment (CI/CD) for QA

This process will checkout the source code from SVN and do compilation. After successful compilation, make a RPM packages and save it to AWS S3 repository (geomanager-qa). If any failures occur during the build, it will notify to the relevant developer.

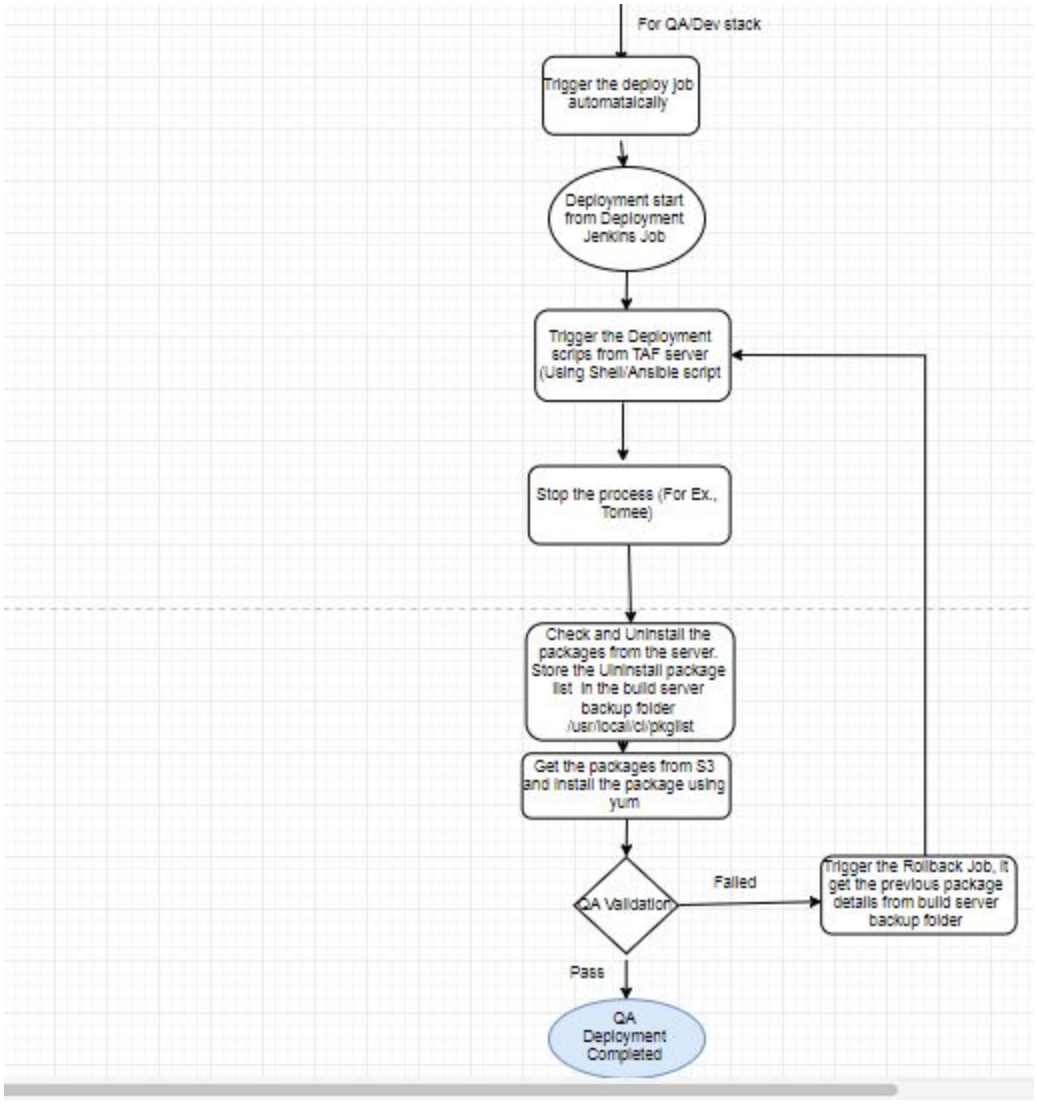
After build completion, it triggers deploy Jenkins job. Take all relevant packages from AWS-S3 and deploy it to respective Dev/QA stacks. We give privilege to Developer to do build and deployment on DEV/QA stacks.

The automation jenkins job will trigger after the successful completion of the deployment on respective stack. This job has categorised for QA, Release, Production stacks. It will execute the testcases specific to the stacks and send results to whoever we mentioned.

Following flowchart gives workflow of Build / Deployment / Rollback



Continue in next page…….

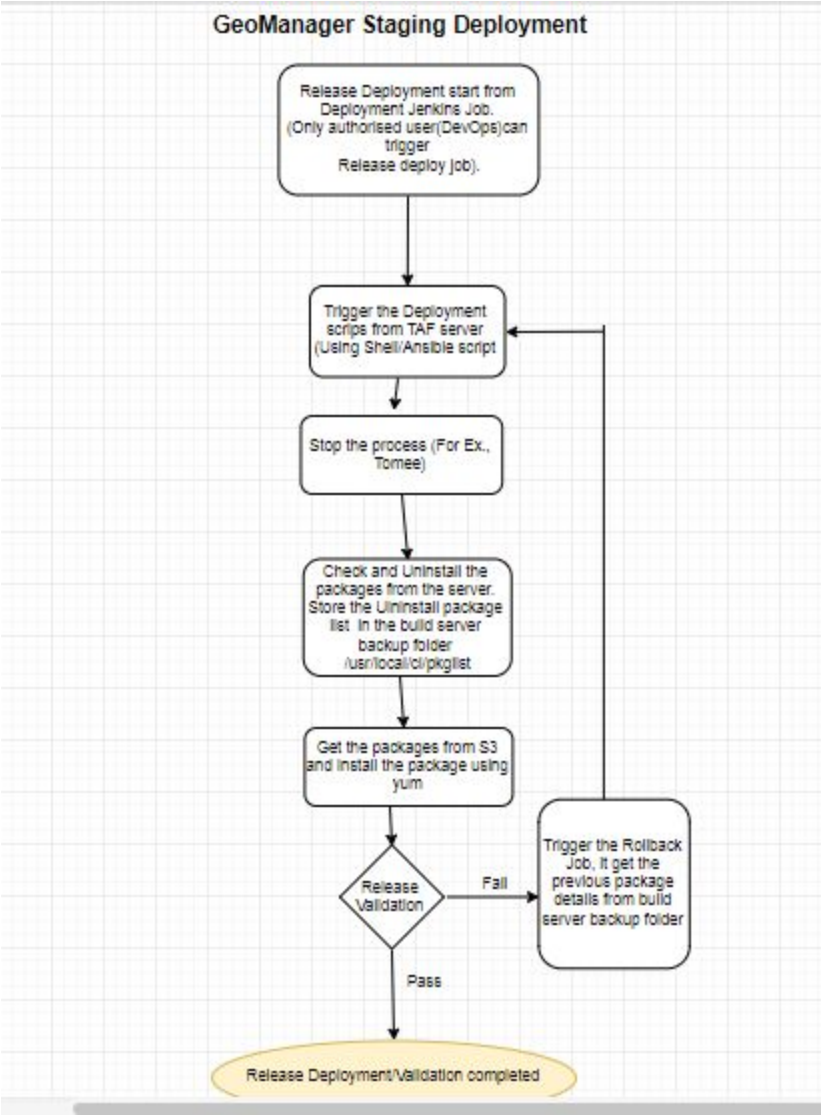

For QA/Dev stack
Trigger the deploy job automatalcally
Deployment start from Deployment Jenkins Job
Trigger the Deployment scrips from TAF server (Using Shell/Ansible script
Stop the process (For Ex., Tomee)
Check and Uninstall the packages from the server. Store the Uninstall package list In the build server backup folder /usr/local/ci/pkglist
Get the packages from S3 and install the package using yum
QA Validation
Failed
Trigger the Rollback Job, it get the previous package details from build server backup folder
Pass
QA Deployment Completed

D. GeoManager Deployment and Rollback for Release and Production

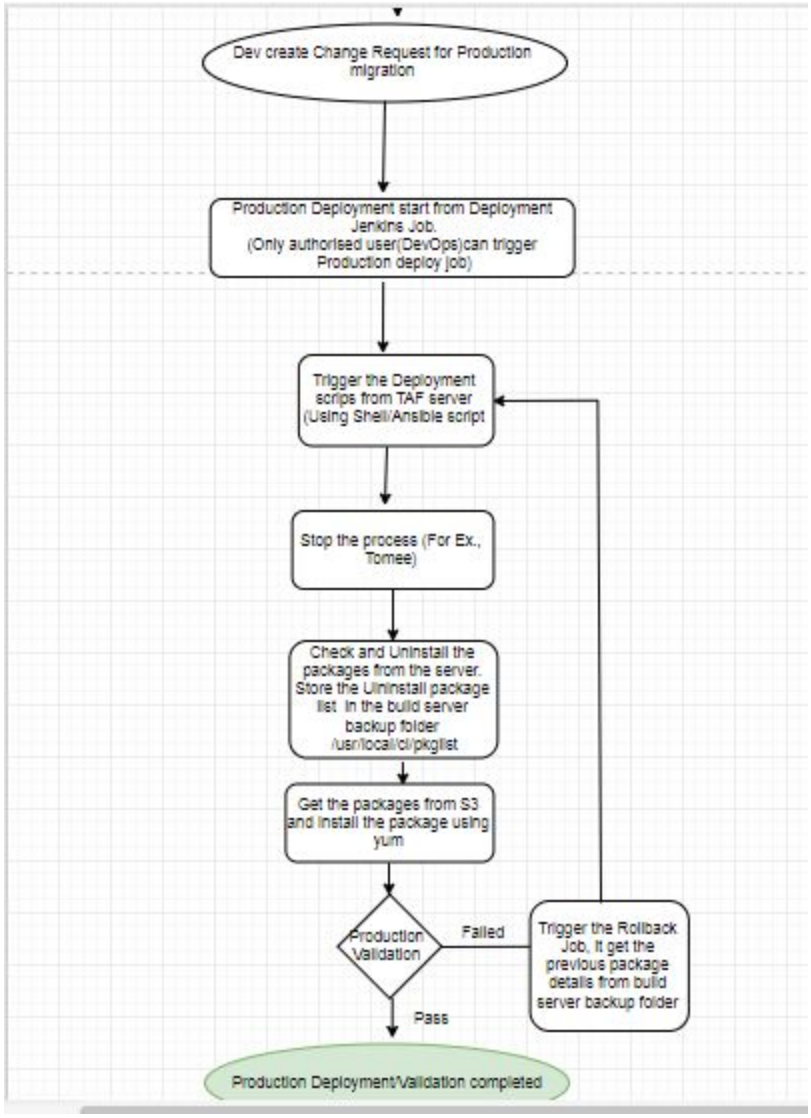
For Release and Production, Infra team will do the deployment. This job should take packages from AWS-S3 (geomanager-prod) repository. We enable approval features for Release and Production deployments.

After completion of deployment, automation jenkins job will trigger for respective stack(Release/Production). It will execute the testcases specific to the stacks and send results to whoever we mentioned. In case of QA failure, we will rollback to previous version using the Rollback jenkins job which use the

Following flowchart gives workflow of staging Build / Deployment / Rollback

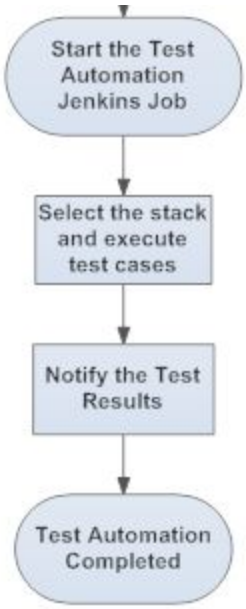


Following flowchart gives workflow of production Build / Deployment / Rollback



E. GeoManager Test Automation

The automation jenkins job will trigger for respective stack(Release/Production). It will execute the testcases specific to the stacks (QA/Release/Production) and send results to whoever we mentioned.



8.0 GeoManager Release Process

The below document provide the details about Release Process we are following in GeoManager.
https://docs.google.com/document/d/1BCFyr6gWJF3I2RAByfo2143tJp7a54gPQY0u27D_ss8/edit

9.0 Document History

Maintain a history of document revisions and versions that have been officially approved following a review.

Version	Modifier	Date	Reason for Change	Approved By
0.1	Kannan Vairamani	20-Nov-2017	Initial Draft	