Pipeline Builds for PLM Applications

GE Internal

Index

GE Internal

- Configure SonarQube
  - Overview
  - Install SonarQube Scanner in Jenkins (Optional)
  - Generate SonarQube user token and update it in Jenkins

- Configure pipeline Global library
  - Overview & Pipeline Syntax
  - Configure pipeline library

- Create Github organization and other util jobs
- How to add a new application and Environment to pipeline
  - Configure Application (ex:GTCC)
  - Configure Environment (DEV1/QA1)

GE Internal

Getting Started with pipeline

GE Internal

Getting Started with pipeline

- **Jenkins Pipeline** is a combination of plugins that support the integration and implementation of continuous delivery **pipelines** using **Jenkins**
- Pipeline feature is introduced to incorporate Continuous Delivery. Continuous delivery ensures that the software is built, tested and released more frequently. It reduces the cost, time and risk of the incremental software releases.
- for PLM applications, pipeline is being used for Continuous integration, Continuous testing and Deployment in DEV and QA environments. Business not yet taken decision on continuous delivery. We can expect this in future.

- *To use Jenkins Pipeline, you will need:*
  - Jenkins 2.x or later (older versions back to 1.642.3 may work but are not recommended)
  - Pipeline plugin

GE Internal

- What you need to configure in new Jenkins instance for pipeline..

1. **Install required Plugins and tools.**

Check the environment section in vars/<groovy files> in pipeline scripts and make sure that they are configured in jenkins. If tools are configured with different name in jenkins then update the same name here.

Ex: antTool name is given as ‘ant’ in env section in below screenshot, if tool name is given as ‘ant1.x’ in new jenkins then update the same name in script. But remember that change in script will impact other jenkins builds incase same scripts are using by other jenkins.

github.build.ge.com/PLM-ENOVIA/plm-secure-pipeline-scripts/blob/power_jenkins_branch/vars/plmSecurePipeline.groovy

```
22      }
23
24  environment {
25              current_ws = pwd()
26              branch = "${env.BRANCH_NAME}"
27
28              /**  POWER JENKINS: Jenkins tools and credential ids */
29              credentials_id='plmenovia_github_cred_id'
30              artifactory_repo="QQHDK"
31              sonarqube_server='plmenovia-propel-sonarqube'
32              sonarqube_scanner='sonarQube Runner'
33              artifactory_server='plmenovia_artifactory_server'
34              antTool='ant'
35              jdkTool='jdk1.8'
36              coverityTool='cov-analysis-2018.12'
37              coverity_server='plm-coverity'
```

GE Internal

2. **User credentials**

Need to configure credentials in order to connect different servers like GitHub, Artifactory, SonarQube and Coverity .

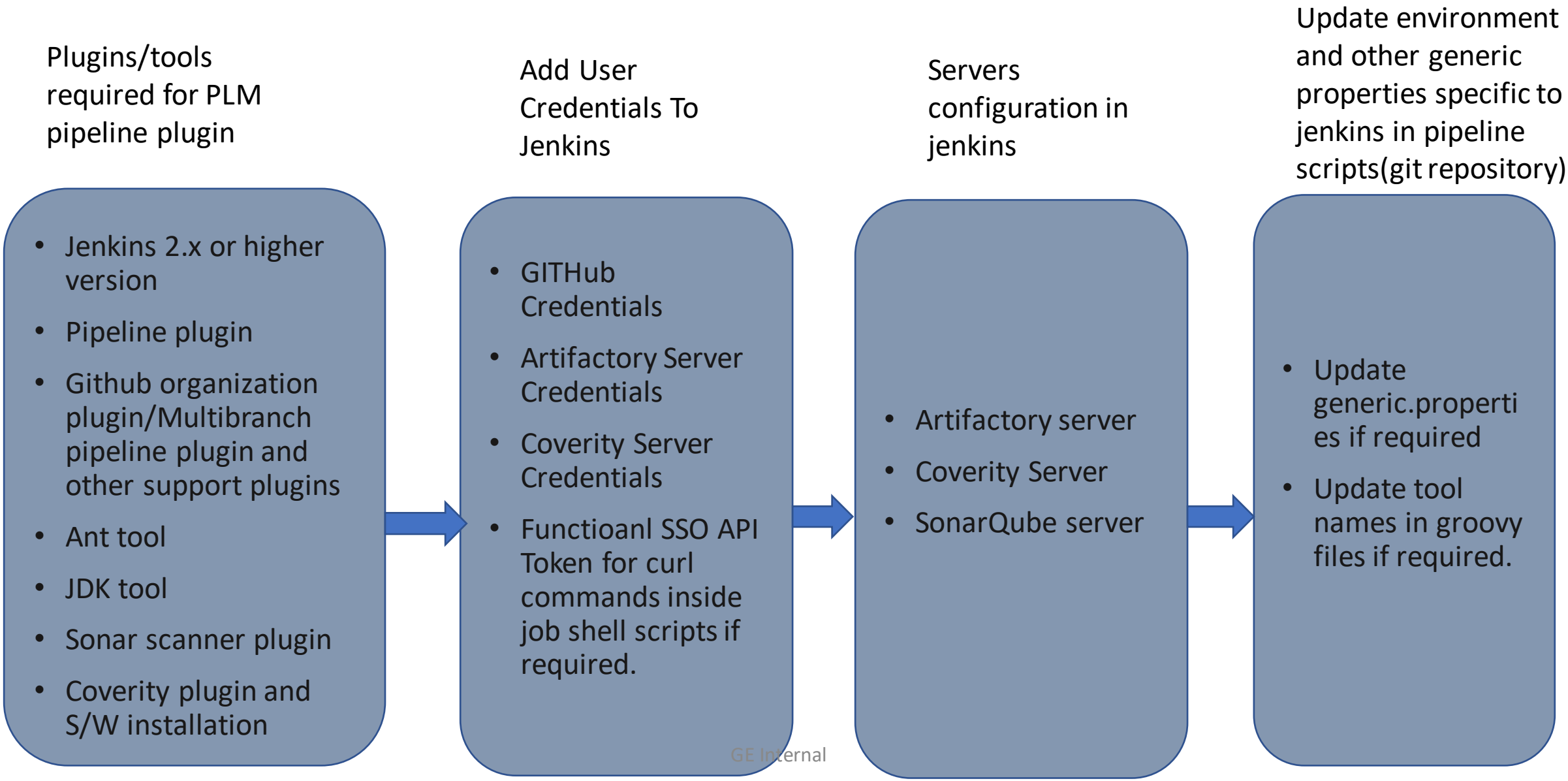
3. **Configure Servers** like artifactory, sonarqube and coverity.

4. **Update pipeline scripts** with jenkins tools, plugins if **required**.

If any tool is given with name “Z” in jenkins then update the same name in pipeline scripts groovy files other wise build will be stopped with errors.

GE Internal

Pre setup in Jenkins and pipeline scripts to get start with pipeline



GE Internal

Checklist

GE Internal

Click on below file to view Jenkins pre setup checklist and Application Environment setup checklist tabs.

Pipeline_powerJenkins_Checklist_V1

Jenkins configuration with tools and plugins

GE Internal

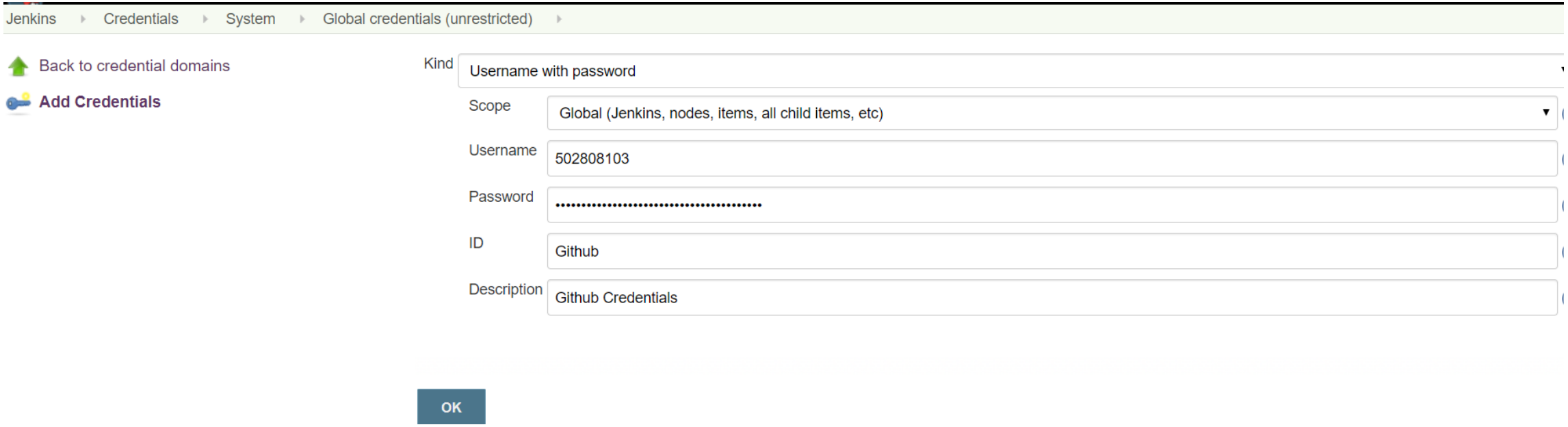
Add GITHUB User Credentials

(functional SSO)

GE Internal

Add user credentials Demonstration

- Go to credentials -> System -> Global Credentials
- Click Add Credentials
- Enter **username, personal access token**, ID and provide meaningful description and save them.



Note: Please refer following slide to create ***personal access token*** for github user

Generate Personal access token - Demonstration

1. Login into github.build.ge.com using **functional SSO.**
2. In the upper-right corner of any page, click your profile photo, then click **Settings**.
3. In the left sidebar, click **Developer settings**.
4. In the left sidebar, click **Personal access tokens**.
5. Click **Generate new token**.
6. Give your token a descriptive name.
7. Select scopes: *admin:org_hook, admin:read:repo_hook, repo* to grant this token. To use your token to access repositories from the command line, select **repo**.
8. Click **Generate token**.
9. Click "copy to clipboard icon" to copy the token to your clipboard. For security reasons, after you navigate off the page, you will not be able to see the token again.

Configure Artifactory

GE Internal

1. Generate API key in Artifactory

- If you already had functional SSO Artifactory API key then you can skip this step.

What is API Key

Artifactory allows authentication for REST API calls using your API key as an alternative to your username and password.

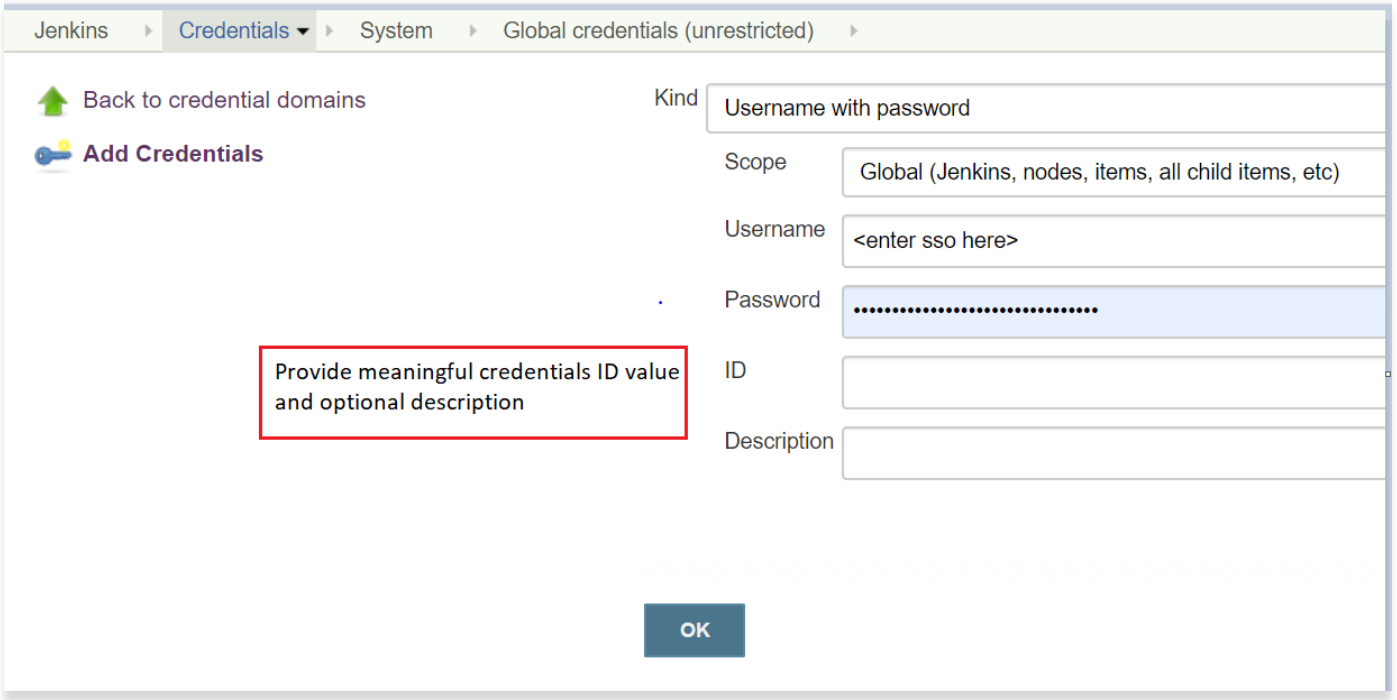
Creating an API Key

- Login with functional SSO and click on profile.
- To edit your profile, you first need to unlock it by entering your current password and clicking **Unlock**.
- To create an API Key, once you have unlocked your profile, click the "Generate" button next to the **API Key** field.

2. Add credentials to Jenkins

Steps to add credentials to jenkins

- Click the **Credentials** link in the sidebar
- Click on the **Global credentials** domain
- Click on **Add Credentials**
- User name is your functional user SSO.
- Password is functional user's Artifactory API token.
- Enter ID and Description and click on OK.

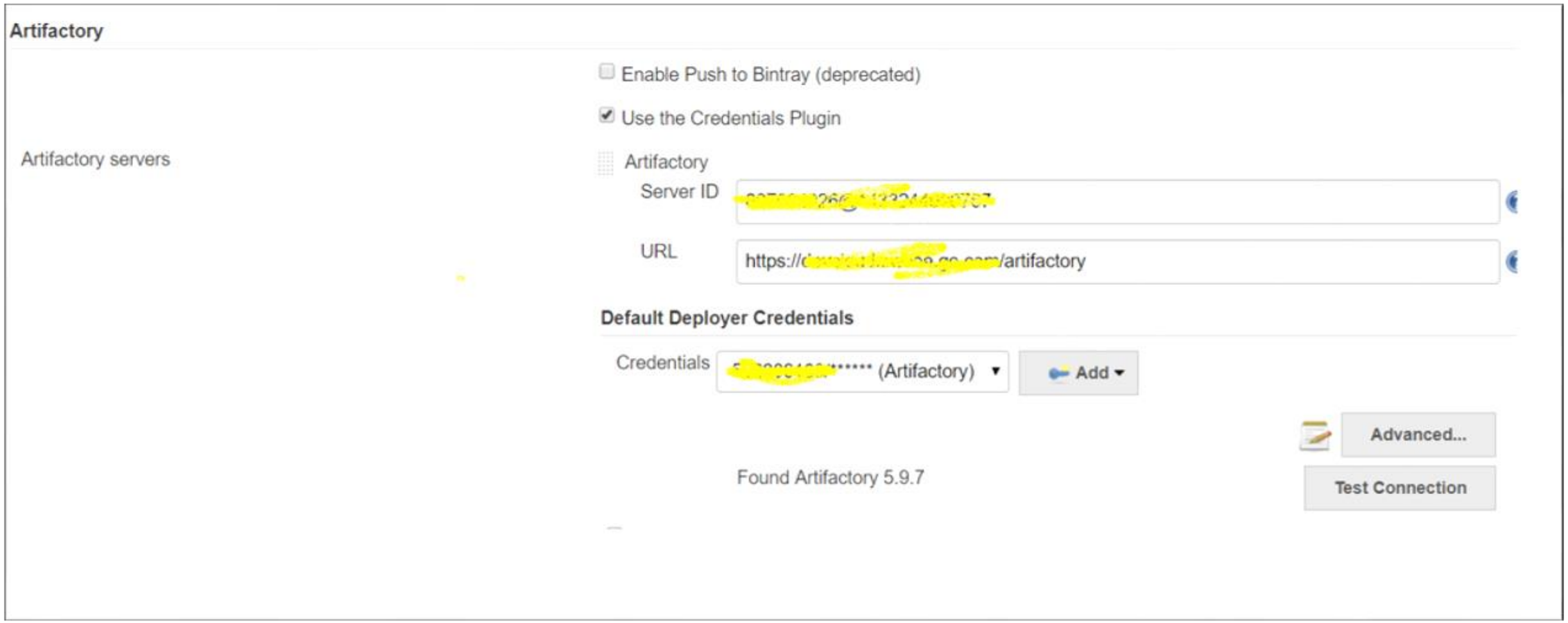


3. Configure Artifactory server in Jenkins

- **T**o configure your Artifactory server settings, go to the Jenkins System Configuration Page (***Manage Jenkins*** > ***Configure System***).
- Click the ***Add Artifactory Server*** button to create a new Artifactory server configuration.

Use the Credentials Plugin	Enable this checkbox to use the Jenkins Credentials Plugin to configure your Artifactory Servers credentials in your Jenkins job configuration pages.
Server ID	Configure the Artifactory server (or servers) that will be used for artifacts resolution and build info deployment.
URL	The Artifactory URL.
Username/Password	(Optional) Username and password that will be used for this Artifactory instance. You can also override these credentials from within the Jenkins Job. **Note**: the credentials are only required if Artifactory is configured not to allow anonymous access.
Use Different Resolver Credentials	(Optional) Enable this checkbox to set different credentials for the resolver.

GE Internal



Note

Server ID: you can give any unique id. Make sure to use the same id in pipeline script where we define tools, plugin and serverid's under environment section.

Configure Coverity

GE Internal

1. Overview

- The Synopsys Coverity for Jenkins plugin enables you to integrate Coverity static analysis tools in your Jenkins builds.

Requirements

- Jenkins versions 2.60.1 or higher. Non-LTS versions of Jenkins are not supported.
- Java 8
- Coverity 2018.03 or higher.

GE Internal

2. Register your application with Coverity

- Check with @POWER Security Coverity Analysis power_security_coverityanalysis@ge.com team to register your application with coverity.

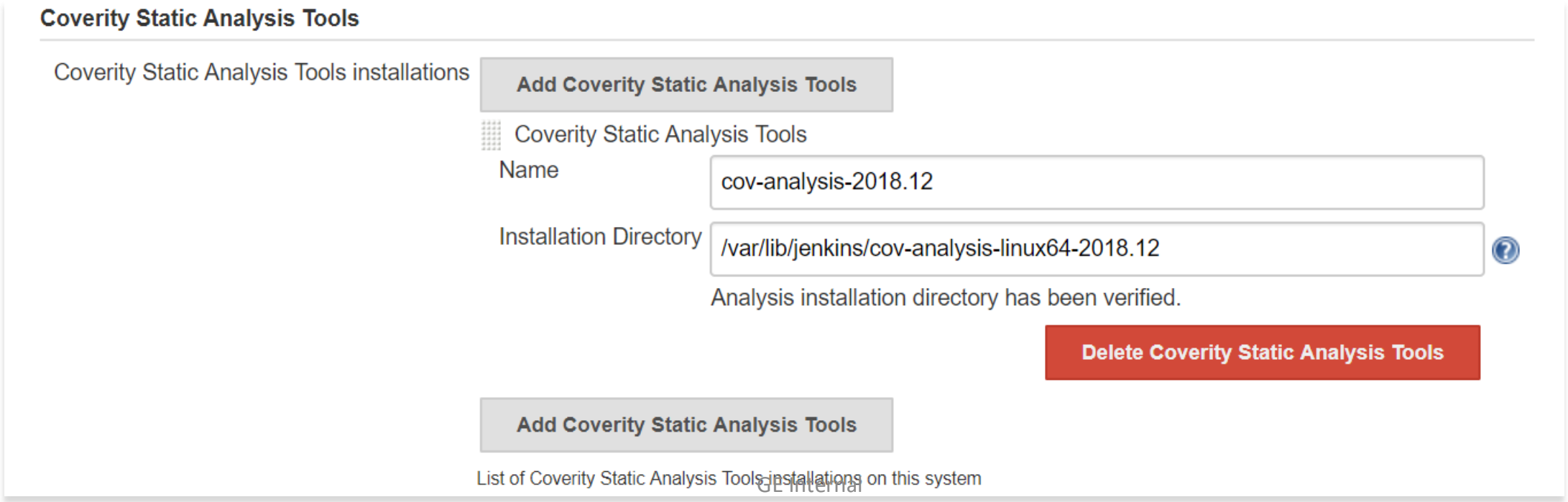
Coverity team will request following information. Share the required information with them.

- Application Config ID : (Go to service now https://geit.service-now.com/navpage.do > (From the left nav panel, under **Configuration**) Business Application.
- Application GIT Repository URL
- Git Committers list (To privilege developers to login and view the report)
- Coverity admin list (CCM team is the admin for PLM applications)

GE Internal

3. Install Coverity in Jenkins

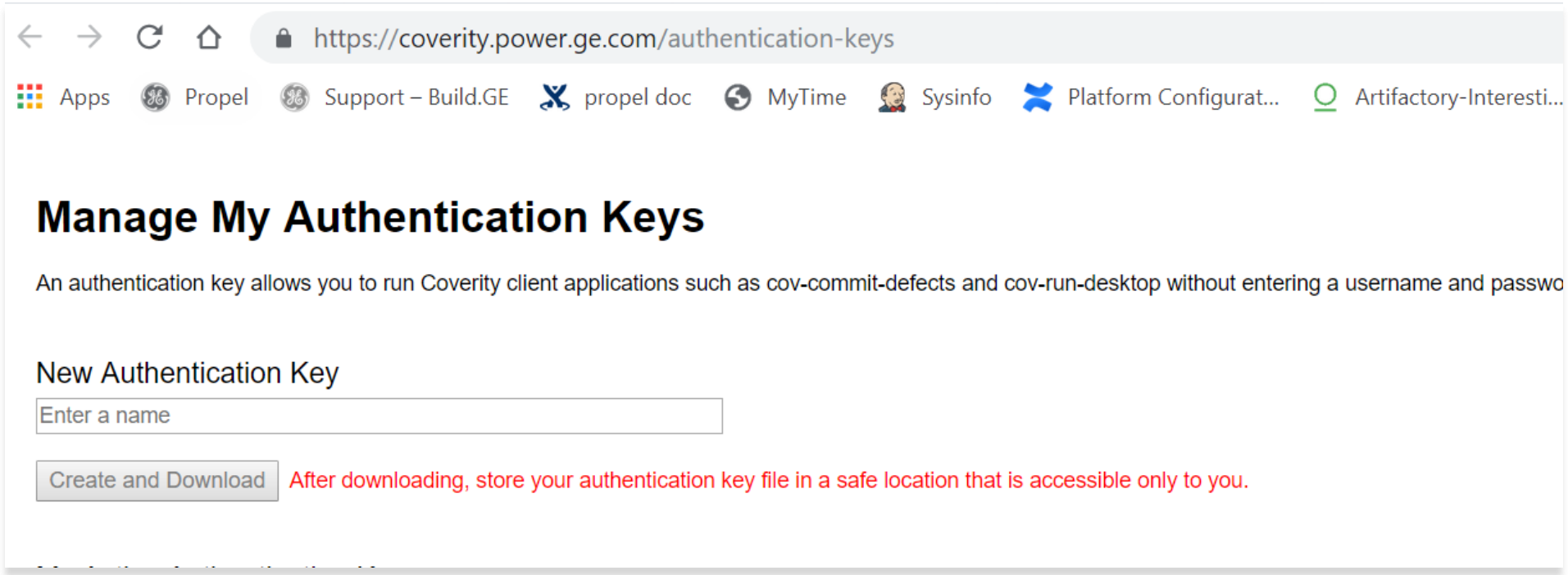
- Coverity team will share the coverity software, license and other details after application gets register with coverity. You can share those details with Nola team to install Coverity in Jenkins.
- Nola team shares installation directory after installation is completed on Jenkins. You need to update the directory path in Jenkins -> Global tool configuration->Coverity Static Analysis Tools



GE Internal

4. Get Coverity connect authentication key

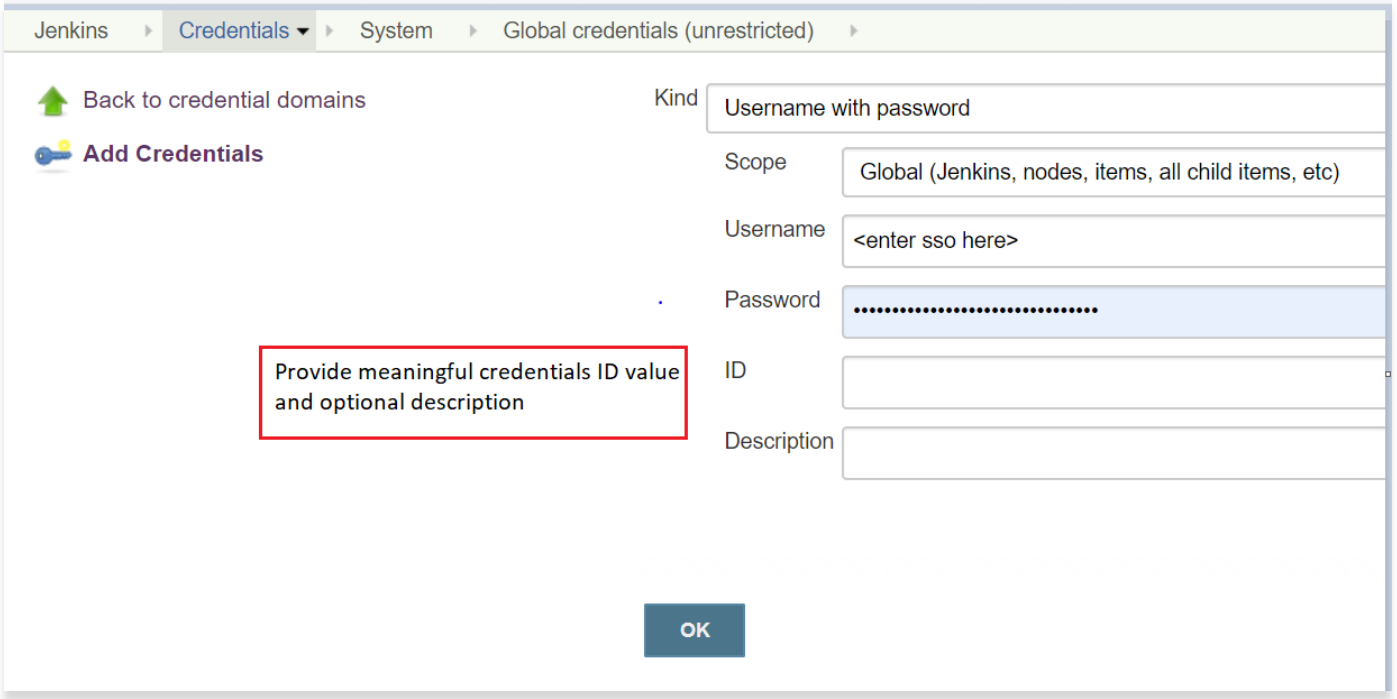
- Login into coverity server (coverity.power.ge.com) with functional SSO credentials.
- Click on user profile then “Authentication keys”
- Enter authentication key name and then click on Create and Download button. You can see encrypted key. copy and save it in txt file to use it as password in coverity credentials configuration described in next slide



5. Add credentials to Jenkins

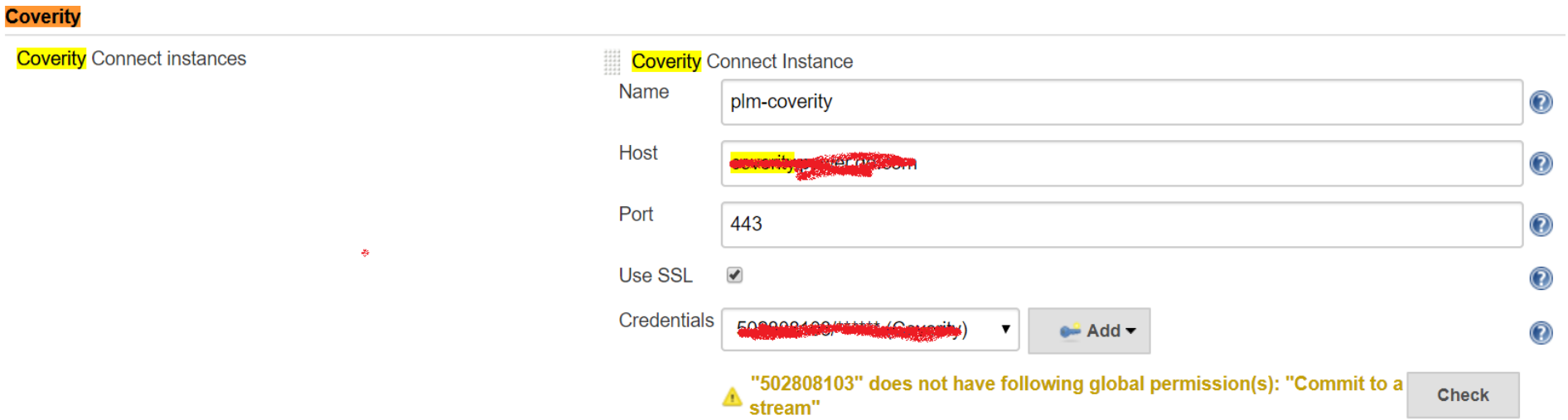
Steps to add credentials to jenkins

- Click the **Credentials** link in the sidebar
- Click on the **Global credentials** domain
- Click on **Add Credentials**
- User name is your functional user SSO.
- Password is functional user's coverity Authentication key.
- Enter ID and Description and click on OK.



6. Add Coverity Connect Instance

- Navigate through Manage Jenkins -> Configuration
- Search with “Coverity” and click on “Add” button
- Provide coverity server name (any name). It will be referred in pipeline scripts.
- Provide covertiy server host and port as 443 as details shared by the coverity server team.
- Select coverity credentials .



Configure Sonarqube

GE Internal

1.Overview

- **Sonar scan is for** static code analysis
- **SonarQube Scanner 3.3** is Compatible with SonarQube 5.6+ (LTS)

- NOLA team provides SonarQube server and SonarQube scanner and you are no need to configure them as they are part of the jenkins image.
- You need to generate SonarQube server user token and update it in SonarQube server configuration.

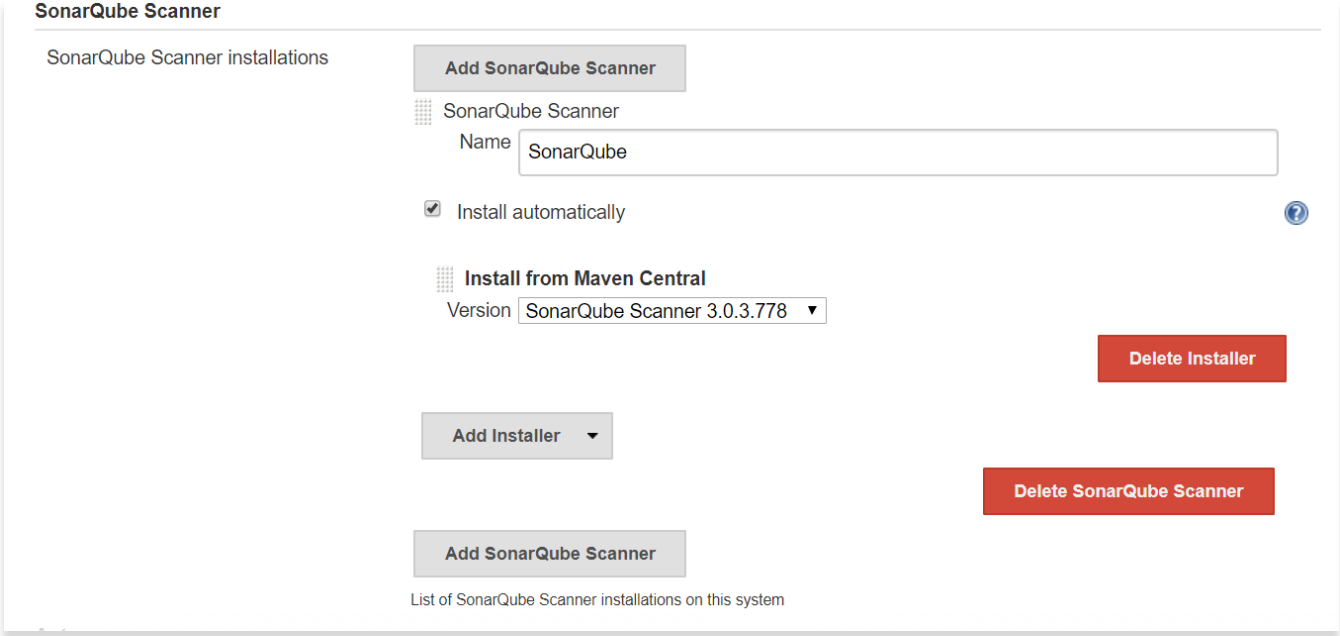
GE Internal

2.Install SonarQube scanner plugin (Optional)

- Nola team provides default SonarQube scanner which is compatible with SonarQube server.
- If you want to install specific version then follow below steps.

Steps to install specific version

- Go to Manage Jenkins -> Global Tool configuration -> Sonarqube Scanner
- Click on Add installer.
- Select version from the list.
- provide scanner name and select “install automatically” checkbox option.
- Save changes.

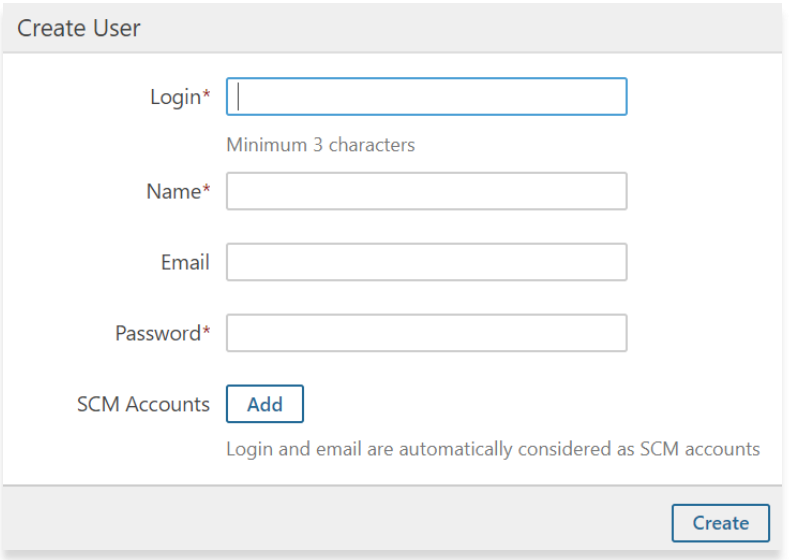


3. Generate SonarQube Server Authentication token and update it in Jenkins

Step#1 Create new user

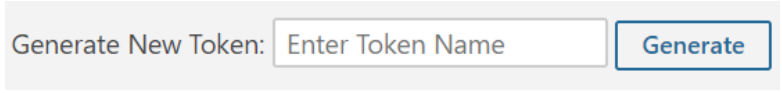
- Navigate through Manage Jenkins -> Configure System
- Find the SonarQube servers section and copy the server URL and open it in browser
- Login into server using admin credentials.
- Navigate through Administration -> Security -> Users -> Create User
- Create new user by providing user name, password and other details.

  Note: Use this user for SonarQube server authentication only.
- Logout



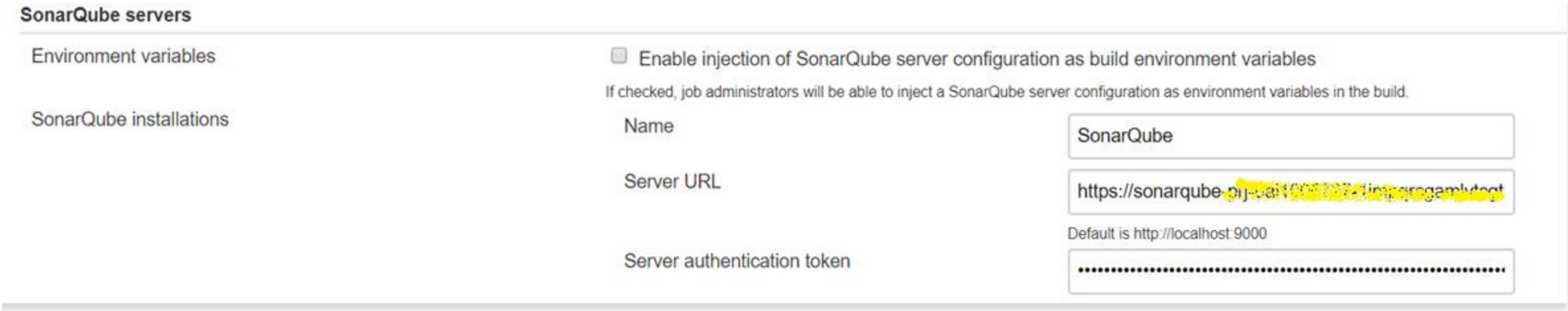
Step#2 Login with new user and generate token

- Login with new user
- Go to My Account -> Security
- Enter Token name and click on Generate
- Copy the generated token in text file.



Step#3 Update token in SonarQube server configuration in Jenkins

- Navigate through Manage Jenkins -> Configure System
- Find the SonarQube servers section
- Provide sonarqube name. It must be unique. If you are not using same sonarqube server for all PLM applications then make sure to add all the servers to configuration add update the pipeline scripts accordingly.
- Enter the token that is generated from previous step into field “Server authentication token”
- Save changes



Create GitHub organization and other util jobs in Jenkins

GE Internal

Create Jobs in Jenkins

- To use Pipeline as Code, projects must contain a file named ***Jenkinsfile*** in the repository root, which contains a "Pipeline script."

- Additionally, one of the enabling jobs needs to be configured in Jenkins:
  - *Multibranch Pipeline*: build multiple branches of a *single* repository automatically
  - *Organization Folders*: scan a **GitHub Organization** or **Bitbucket Team** to discover an organization's repositories, automatically creating managed *Multibranch Pipeline* jobs for them.

Fundamentally, an organization's repositories can be viewed as a hierarchy, where each repository may have child elements of branches and pull requests.

Note: *Multibranch Pipeline* and *Organization Folders* eliminate the manual process by detecting branches and repositories, respectively, and creating appropriate folders with jobs in Jenkins automatically.

GE Internal

The Jenkinsfile

- Presence of the Jenkinsfile in the root of a repository makes it eligible for Jenkins to automatically manage and execute jobs based on repository branches.

- The Jenkinsfile should contain a Pipeline script, specifying the steps to execute the job.

GE Internal

PLM Jobs hierarchy for pipeline

Job Name: PLM-Enovia

PLM-Enovia GITHUB organization job

1. Create organization job

gtccplm

Gtccplm_spinner

Powerplm_2016x

Branch 1 Jenkinsfile

Branch N

Branch 1

Branch N

Branch 1

Branch N Jenkinsfile

Gtccplm-app-db-job

Use app-db job for manual builds

powerplm-app-db-job

Gtccplm-dev1-trigger

Gtccplm-qa1-trigger

powerplm-dev1-trigger

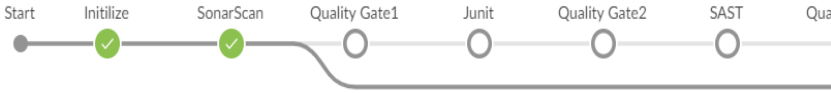
powerplm-dev2-trigger

powerplm-qa1-trigger

Crete trigger job for each environment

- Organization job creates a ***Multibranch Pipeline*** project for each repository.
- Repositories under organization will be shown in hierarchy.

- Repositories have child elements of branches which contains Jenkinsfile that runs pipeline



GE Internal

Create Organization folder Job

GE Internal

Create organization job

- To create an organization Folder in Jenkins, go to: **New Item → GitHub Organization** and follow the configuration steps for each item, making sure to specify appropriate *Scan Credentials* and a specific **owner** for the GitHub Organization, respectively.

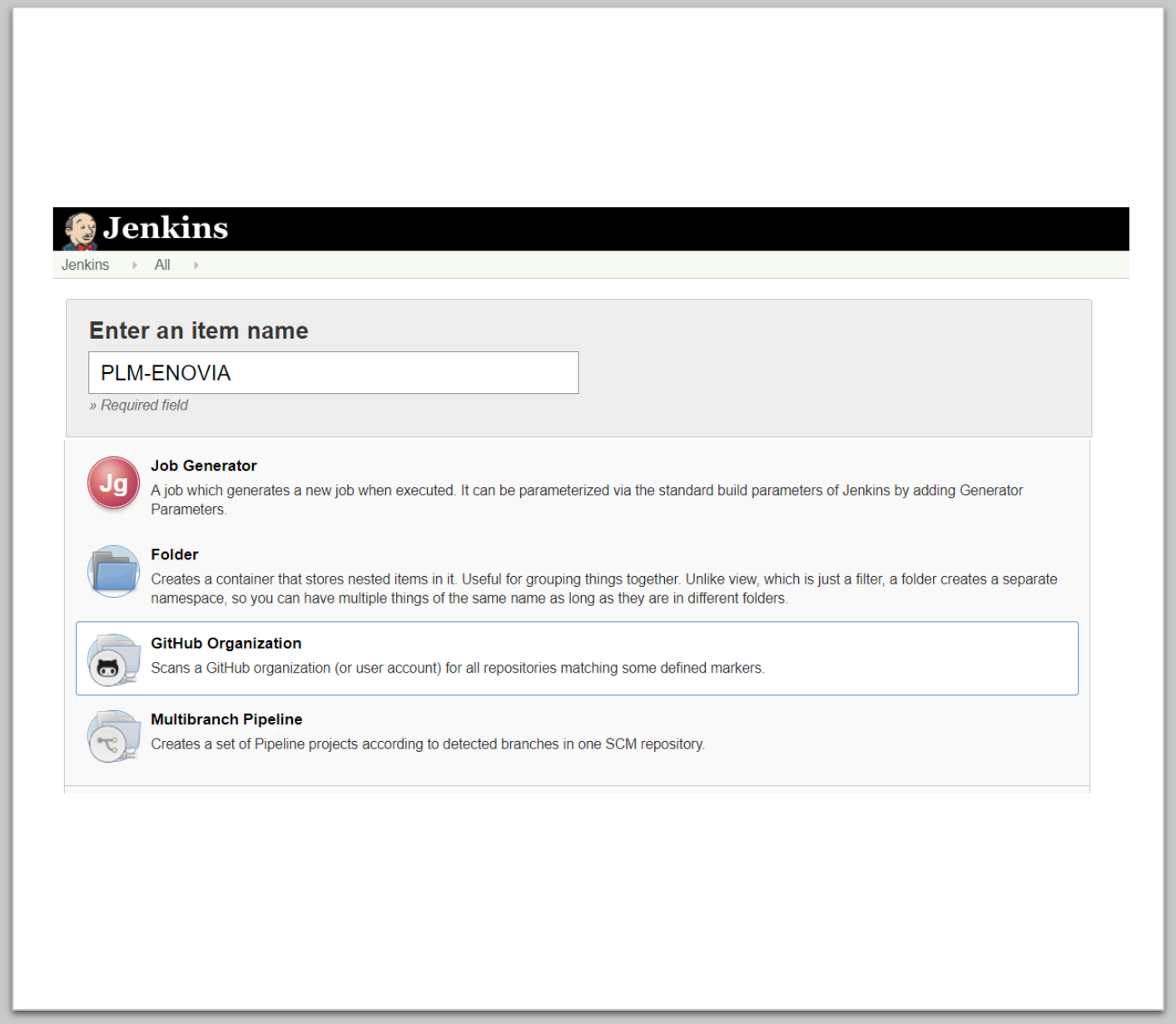
- Other options available are:
- **Repository name pattern** - a regular expression to specify which repositories are **included**
- **API endpoint** - an alternate API endpoint to use a self-hosted GitHub Enterprise
- **Checkout credentials** - alternate credentials to use when checking out the code (cloning)

After configuring these items and saving the configuration, Jenkins will automatically scan the organization and import appropriate repositories and resulting branches.

GE Internal

Demonstration

- Go to **New Item**. Enter ‘PLM-ENOVIA’ for the item name.
- Select **GitHub Organization** and click **OK**.



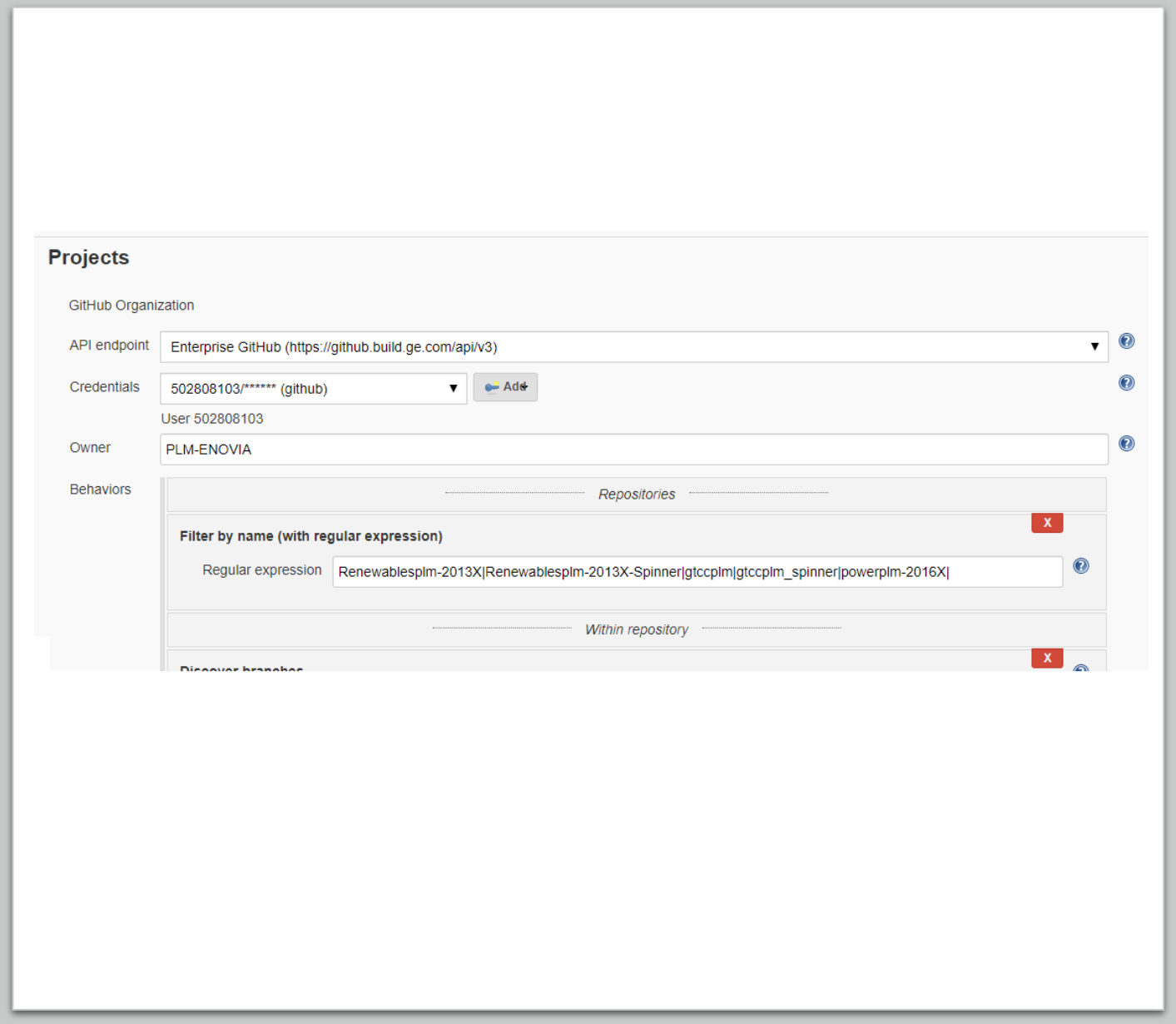
GE Internal

- Make sure the **owner** matches the GitHub Organization name exactly.
- select or add new "credentials"

we'll enter our GitHub username and access token as the password.

- You can specify repositories names or use .* to refer all the repositories in Regular expression under Repositories section.

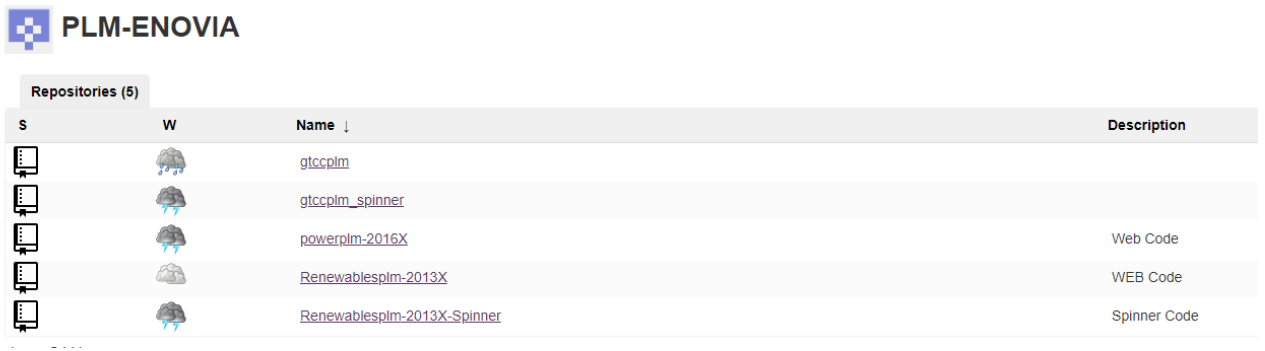
After saving, the "Folder Computation" will run to scan for eligible repositories, followed by multibranch builds.



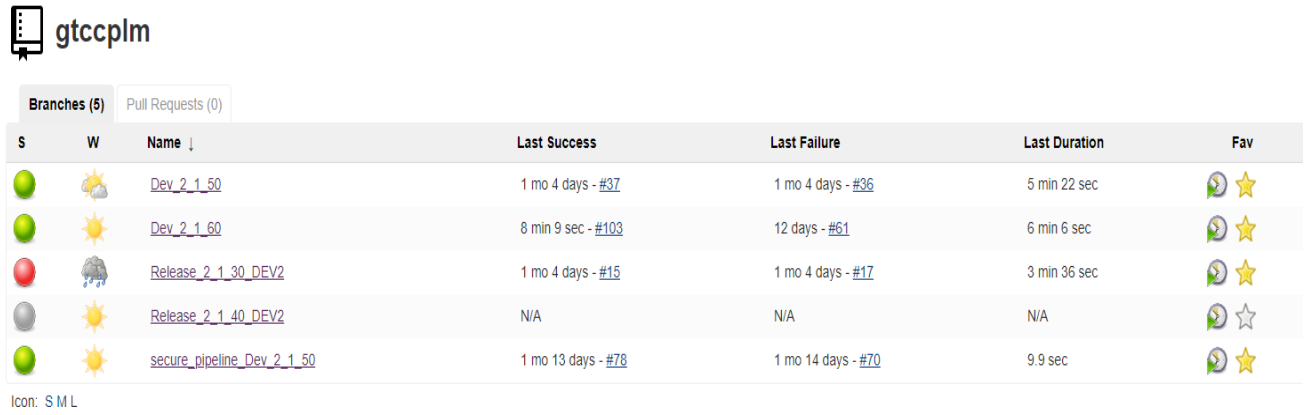
At this point, you're finished with basic project configuration and can now explore your imported repositories.

You can also investigate the results of the jobs run as part of the initial *Folder Computation*.

Repositories hierarchy for PLM-ENOIVA organization



Branches under gtccplm which contains Jenkinsfile at branch root directory



GE Internal

Pipeline build sample screenshot#

Generally we run sonarscan, Junit and SAST as part of builds. In above example only sonar scan stage is executed. Stage execution can be controlled based on parameters passed to the build. Its configured in trigger job.

How to add a new application and Environment to pipeline

GE Internal

- Steps for adding new **application** and **environment** to pipeline are already mentioned in checklist excel file.
- Same are demonstrated in following slides for GTCCPLM application

GE Internal

Add New Application to pipeline

GE Internal

Add New application to pipeline

- Prerequisites:
  - Jenkins pre setup (Refer checklist section in previous slides)
  - GITHUB organization job along with pipeline library configuration in Jenkins
  - Basic knowledge in pipeline and PLM builds.

GE Internal

Add GTCC Application to Pipeline

- *Application Code changes*
  - *Web code changes*

web build xml file changes (Ex: gtccplm-secure-pipeline-jenkins-main-build.xml)

- Copy Existing web build.xml files into ant/build folder and rename them.
- Remove hard coded usercontent library path and replace it with dynamic parameter values send during build.

  Replace

  ```
  <fileset dir="/jenkins/data/jenkins_home/userContent/pw_web_plm/lib"
  includes="*.jar"/>
  <fileset dir="/jenkins/data/jenkins_home/userContent/pw_web_plm/bin"
  includes="*.jar"/>
  ```

  With

  ```
  <fileset dir="${classpath-lib}" includes="*.jar"/>
  <fileset dir="${classpath-bin}" includes="*.jar"/>
  ```

- Add target to prepare tar.gz file if application arfifacts are moved as .tar file into artifactory. For some applications only war file is being copied so tar target is not required in this case.

GE Internal

Web code changes

Spinner build xml file changes (Ex: gtccplm-secure-pipeline-jenkins-pre-build.xml)

- Copy Existing Spinner build.xml files into ant/build folder and rename them.
- Remove hard coded usercontent library path and replace it with dynamic parameter values send during build.

  Replace

  ```
  <fileset dir="/jenkins/data/jenkins_home/userContent/pw_web_plm/lib"
  includes="*.jar"/>
  <fileset dir="/jenkins/data/jenkins_home/userContent/pw_web_plm/bin"
  includes="*.jar"/>
  ```

  With

  ```
  <fileset dir="${classpath-lib}" includes="*.jar"/>
  <fileset dir="${classpath-bin}" includes="*.jar"/>
  ```

- Add target to prepare tar.gz file if application arficts are moved as .tar file into artifactory. For some applications only war file is being copied so tar target is not required in this case.
- Add targets spinner_coverity and spinner_copy_alljpo to compile GE specific files as part of coverity testing.

- *Web code changes*

Junit Related changes

- Add junit-plm-test.xml in ant/build folder . This file is used to run junit test cases on spinner code. May need some changes in file if application specific parameters are used.
- Add junit-noframes.xsl file used as css for report formatting. File location : ant/build
- Add environment specific junit context properties file in war/src folder
  - Ex: war/src/dev1-junit-context.properties

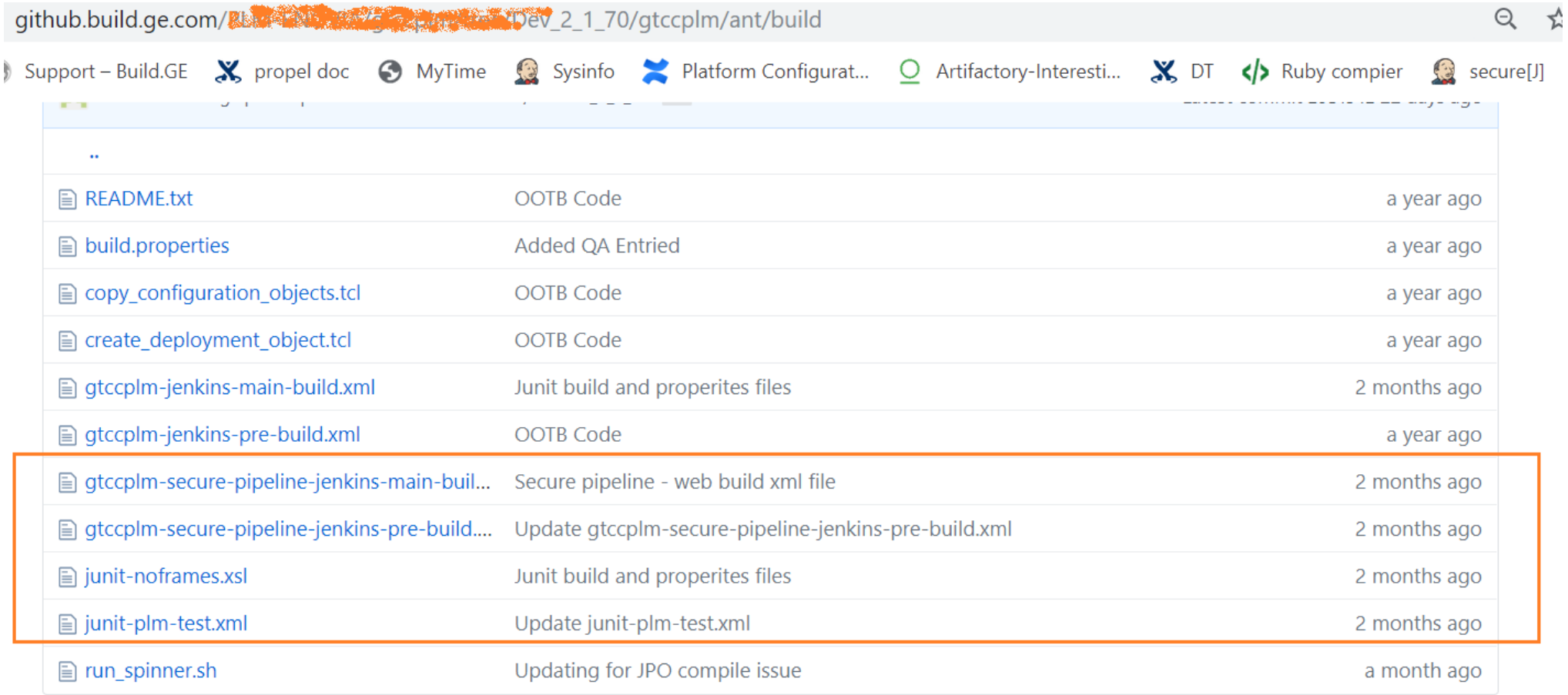
Jenkinsfile

- Add Jenkinsfile to the root directory of the web code.

Note: Copy all these files to master branch as well after testing is completed on at least one environment.

Confidential

Screenshot#


github.build.ge.com/[redacted]dev_2_1_70/gtccplm/ant/build
Support – Build.GE
propel doc
MyTime
Sysinfo
Platform Configurat...
Artifactory-Interesti...
DT
Ruby compier
secure[J]
..
README.txt OOTB Code a year ago
build.properties Added QA Entried a year ago
copy_configuration_objects.tcl OOTB Code a year ago
create_deployment_object.tcl OOTB Code a year ago
gtccplm-jenkins-main-build.xml Junit build and properites files 2 months ago
gtccplm-jenkins-pre-build.xml OOTB Code a year ago
gtccplm-secure-pipeline-jenkins-main-buil... Secure pipeline - web build xml file 2 months ago
gtccplm-secure-pipeline-jenkins-pre-build.... Update gtccplm-secure-pipeline-jenkins-pre-build.xml 2 months ago
junit-noframes.xsl Junit build and properites files 2 months ago
junit-plm-test.xml Update junit-plm-test.xml 2 months ago
run_spinner.sh Updating for JPO compile issue a month ago

- *Spinner code changes*

Sonar-Project properties file

- Add Sonar-Project.properties file and modifyy the properties. Developer need to update project name, key every time for the new branch

Jenkinsfile

- Add Jenkinsfile to the root directory of the Spinner code.

Note: Copy all these files to master branch as well after testing is completed on at least one environment.

GE Internal

- *New job(s) in Jenkins*
- These are the common jobs for GTCC Application.

APP-DB job for Application. (Example: PLM-GTCCPLM-APP-DB)

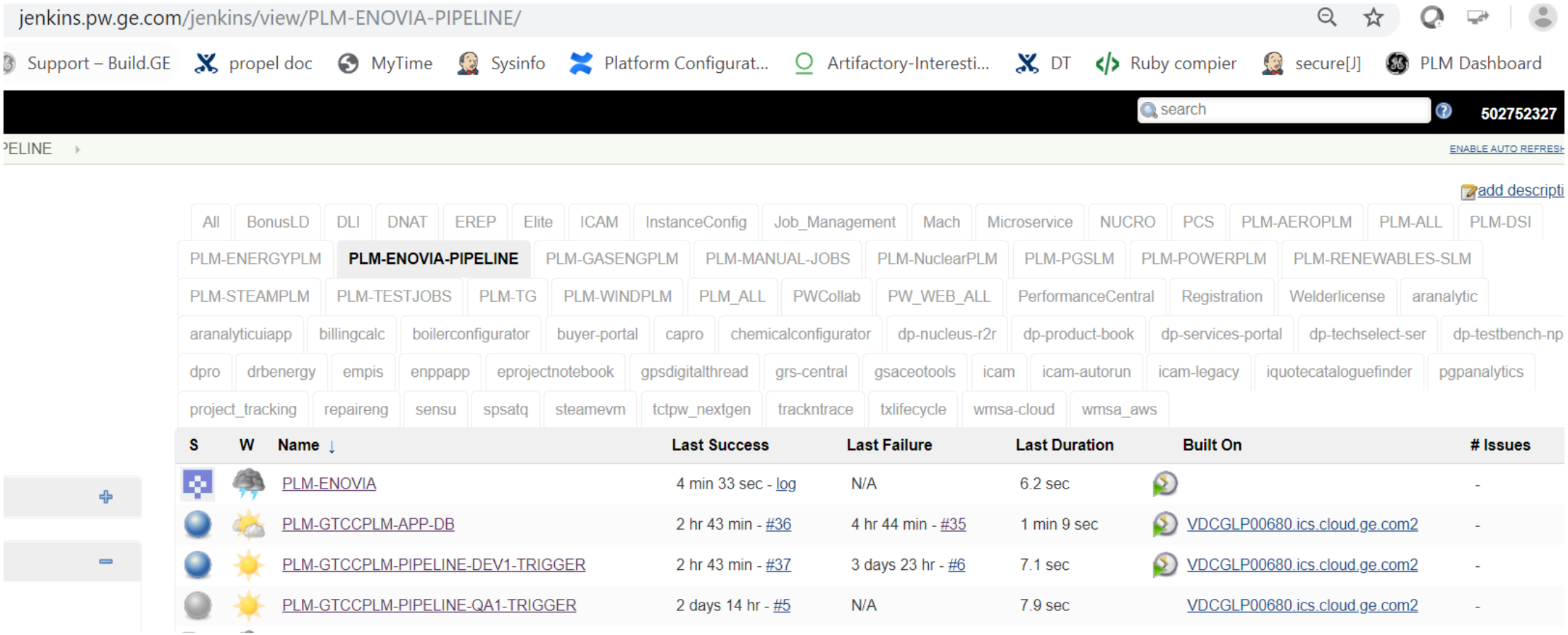
- Each trigger job specific to application triggers APP-DB job. Build artifacts generated from pipeline will be moved to the folder created with build numbe in artifactory.

Example: if APP-DB job build number is 100, then appconfig file is updated with 100 and artifacts will be copied to the folder "100" under application.

- **Post production job for Application** (Example: PLM-SPINNER-POST-PROD)

- This job will poll the spinner git repository to checkout latest post production code.
- Scheduling to run one time every day.
- Update coverity_postprod_branch_workspace property with this job workspace path in spinner pipeline properties file (ex: gtccplm-spinner-secure-pipeline.properties) in pipeline scripts

Screenshot#


jenkins.pw.ge.com/jenkins/view/PLM-ENOVIA-PIPELINE/
Support – Build.GE
propel doc
MyTime
Sysinfo
Platform Configurat...
Artifactory-Interesti...
DT
Ruby compier
secure[J]
PLM Dashboard
search
502752327
ENABLE AUTO REFRESH
add descripti
All
BonusLD
DLI
DNAT
EREP
Elite
ICAM
InstanceConfig
Job_Management
Mach
Microservice
NUCRO
PCS
PLM-AEROPLM
PLM-ALL
PLM-DSI
PLM-ENERGYPLM
PLM-ENOVIA-PIPELINE
PLM-GASENGPLM
PLM-MANUAL-JOBS
PLM-NuclearPLM
PLM-PGSLM
PLM-POWERPLM
PLM-RENEWABLES-SLM
PLM-STEAMPLM
PLM-TESTJOBS
PLM-TG
PLM-WINDPLM
PLM_ALL
PWCollab
PW_WEB_ALL
PerformanceCentral
Registration
Welderlicense
aranalytic
aranalyticuiapp
billingcalc
boilerconfigurator
buyer-portal
capro
chemicalconfigurator
dp-nucleus-r2r
dp-product-book
dp-services-portal
dp-techselect-ser
dp-testbench-np
dpro
drbenergy
empis
enppapp
eprojectnotebook
gpsdigitalthread
grs-central
gsaceotools
icam
icam-autorun
icam-legacy
iquotecataloguefinder
pgpanalytics
project_tracking
repaireng
sensu
spsatq
steamevm
tctpw_nextgen
trackntrace
txlifecycle
wmsa-cloud
wmsa_aws
S
W
Name
Last Success
Last Failure
Last Duration
Built On
Issues
PLM-ENOVIA
4 min 33 sec - log
N/A
6.2 sec
-
PLM-GTCCPLM-APP-DB
2 hr 43 min - #36
4 hr 44 min - #35
1 min 9 sec
VDCGLP00680.ics.cloud.ge.com2
-
PLM-GTCCPLM-PIPELINE-DEV1-TRIGGER
2 hr 43 min - #37
3 days 23 hr - #6
7.1 sec
VDCGLP00680.ics.cloud.ge.com2
-
PLM-GTCCPLM-PIPELINE-QA1-TRIGGER
2 days 14 hr - #5
N/A
7.9 sec
VDCGLP00680.ics.cloud.ge.com2
-

Pipeline library script changes

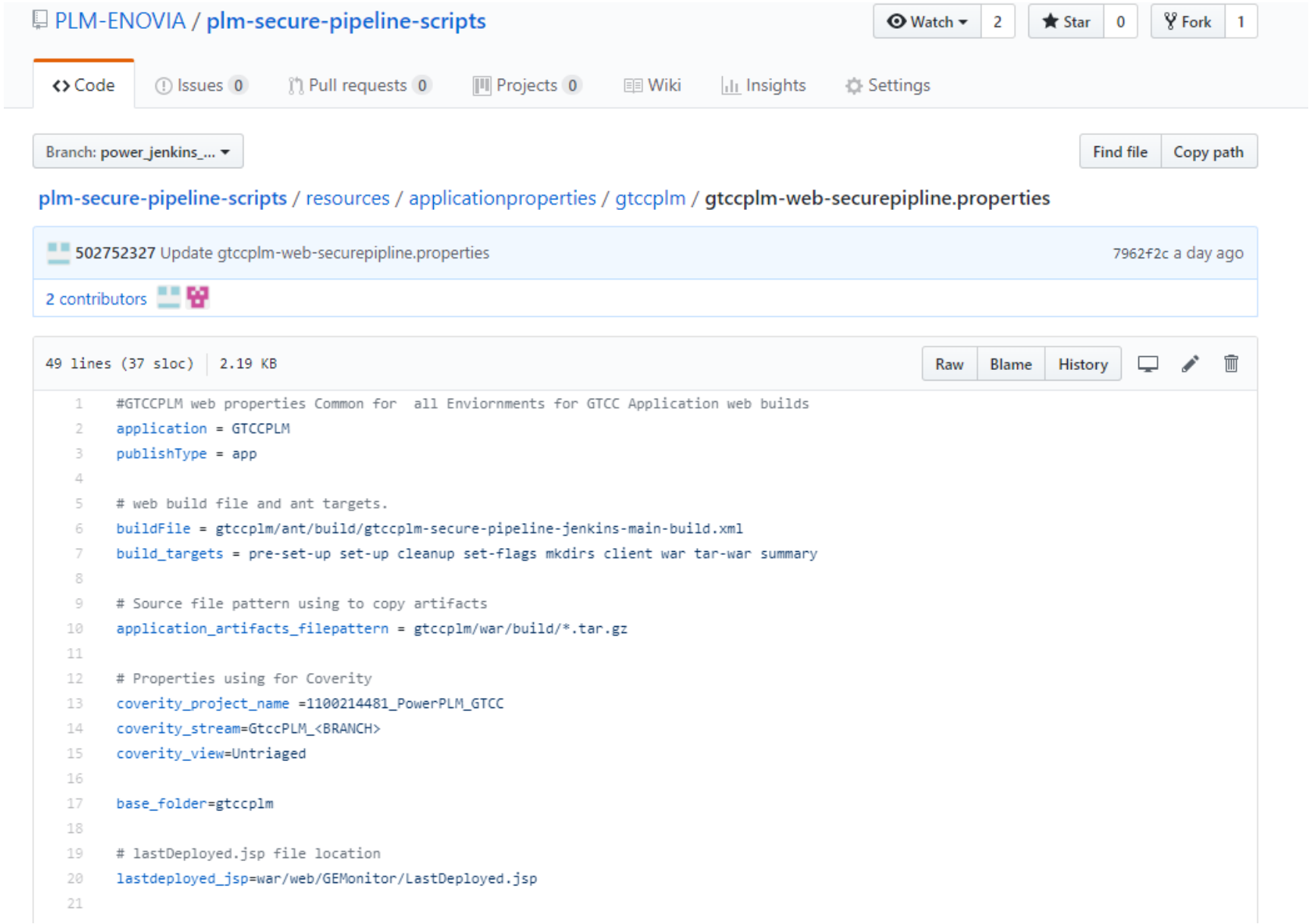
- Clone pipeline GitHub URL : https://github.build.ge.com/PLM-ENOVIA/plm-secure-pipeline-scripts
- Checkout ***power_jenkins_branch*** for builds in power jenkins.
- Create folder with application name or any meaningful name under ***resources/applicationproperties*** and add properties files for web and spinner builds. You can give any name to the properties file.
  - Example:
  - resources/applicationproperties/gtccplm/gtccplm-web-securepipline.properties
  - resources/applicationproperties/gtccplm/gtccplm-spinner-secure-pipeline.properties
- Map above two properties files with application name in *resources/application_mapping.properties file.* Key should be the git repository name. Pipeline scripts unable to read properties file for specific application if key name does not match with repository name.

  Example mapping :

  gtccplm = applicationproperties/gtccplm/gtccplm-web-securepipline.properties

  gtccplm_spinner = applicationproperties/gtccplm/gtccplm-spinner-secure-pipeline.properties

- Screenshot(s)#



Branch: power_jenkins_... | Find file | Copy path

plm-secure-pipeline-scripts / resources / applicationproperties / gtccplm / **gtccplm-spinner-secure-pipeline.properties**

502752327 add description to properties — b3deefa a day ago

2 contributors

58 lines (44 sloc) | 2.74 KB | Raw | Blame | History

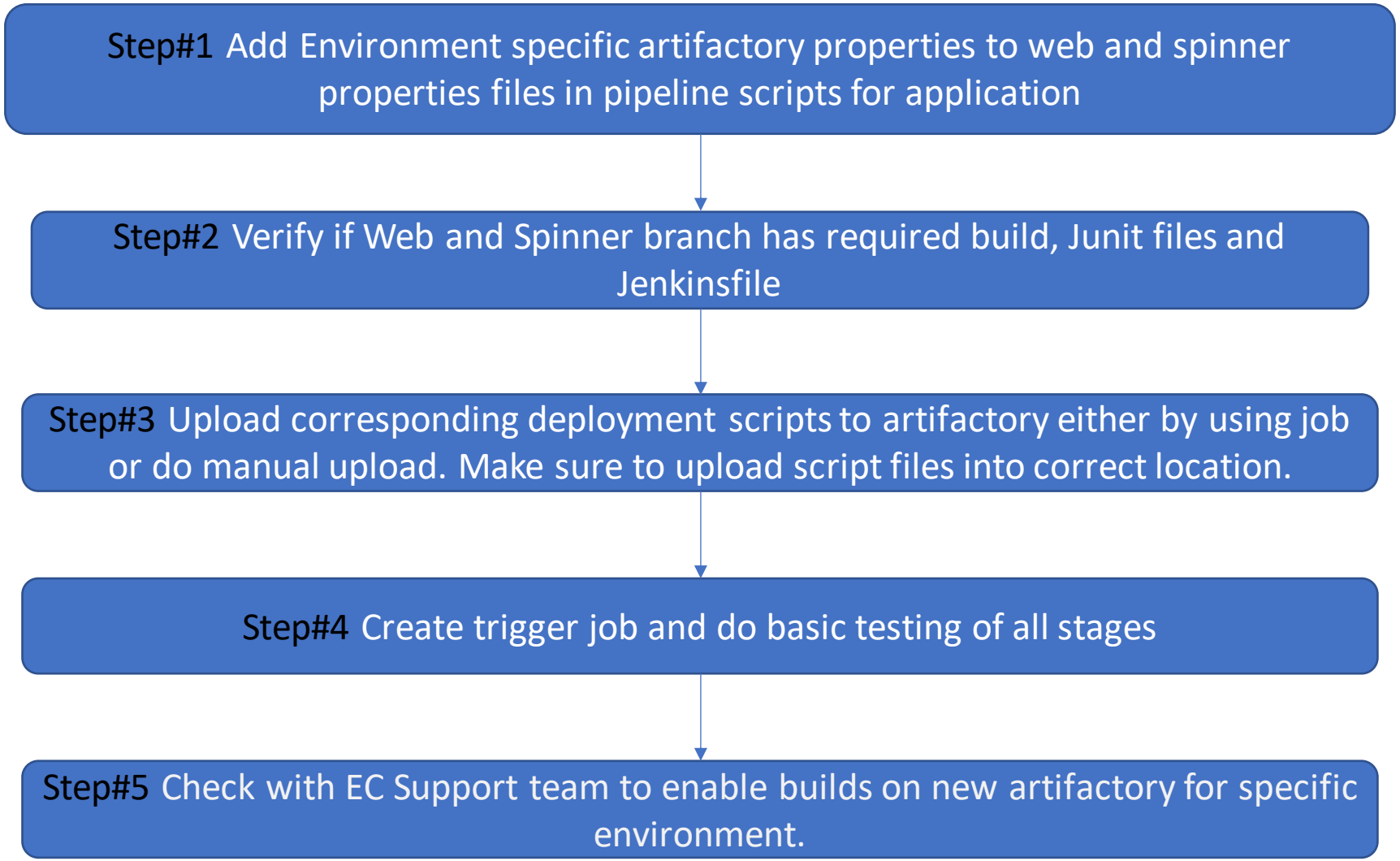
```
#GTCCPLM Spinner properties Common for  all Enviornments for GTCC Application spinner builds
application = GTCCPLM
publishType = db
git_base_folder=gtccplm_spinner

# Spinner build file and ant targets.
buildFile = gtccplm-secure-pipeline-jenkins-pre-build.xml
build_targets = pre-set-up set-up custom-jar-deploy copy-spinner tar-spinner

# Source file pattern using to copy artifacts
application_artifacts_filepattern = *tar.gz

# properties using to web code checkout to copy build.xml and other files required for spinner build.
webcode_git_repo_url = https://github.build.ge.com/PLM-ENOVIA/gtccplm.git
web_repo_base_folder = gtccplm
web_repo_name = gtccplm

# Properties using for Coverity
coverity_project_name = 1100214481_PowerPLM_GTCC
coverity_stream=GtccPLM_Spinner_<BRANCH>
coverity_build_targets = pre-set-up set-up custom-jar-deploy copy-spinner spinner_copy_all_jno spinner_coverity
```

GE Internal

Add New Environment

GE Internal

Adding New Environment to Pipeline



GE Internal

GTCCPLM-DEV1 Demonstration

- *Step#1 Add Environment specific artifactory properties to pipeline*

  - **File Name : gtccplm-web-securepipline.properties**

URL: https://github.build.ge.com/PLM-ENOVIA/plm-secure-pipeline-scripts/blob/power_jenkins_branch/resources/applicationproperties/gtccplm/gtccplm-web-securepipline.properties

Following are the web entries corresponding to DEV1:

```
#Following properties are specific to Enviroment. Its required to run builds on specific envioronment.
# Always start property name with environment as its configured in trigger job and append it with _ (Ex: DEV1_)
# <env>_artifactory_appconfig_file is the appconfig.json file location path in artifactory (Mandatory)
# <env>_artifactory_artifact_file path is used as target to copy artifacts into artifactory (Mandatory)
# <env>_artifactory_deployment_script property is optional. it will not impact the build. Its using for maintaining and providing correct

#DEV1  properties
DEV1_artifactory_appconfig_file=GTCCPLM/GTCCPLM-3DSPACE-DEV1/app/appConfig.json
DEV1_artifactory_artifact_file=GTCCPLM/app/<BUILD_NUMBER_TO_REPLACE>/gtccplm.tar.gz
DEV1_artifactory_deployment_script=GTCCPLM/app/app-GTCCPLM-3DSPACE-DEV1.sh
```

- **File Name : gtccplm-Spinner-securepipline.properties**

Following are the spinner entries corresponding to DEV1:

```
#DEV1 Spinner properties
DEV1_artifactory_appconfig_file=GTCCPLM/GTCCPLM-3DSPACE-DEV1/app/appConfig.json
DEV1_artifactory_artifact_file=GTCCPLM/db/<BUILD_NUMBER_TO_REPLACE>/gtccplm_spinner.tar.gz
DEV1_artifactory_deployment_script=GTCCPLM/db/db-GTCCPLM-3DSPACE-DEV1.sh
```

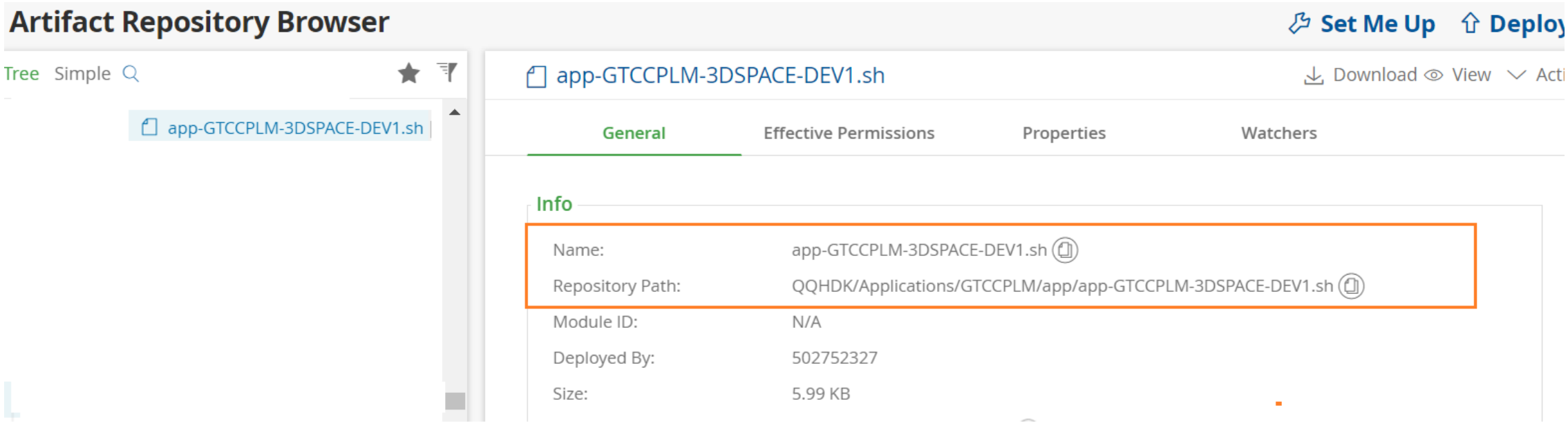
- *Step#2 Verify files in web and spinner code*

- Required files in web branch
  - Ant/build/<APPLICATION SPECIFIC WEB BUILD FILE> Example: gtccplm-secure-pipeline-jenkins-main-build.xml
  - Ant/build/< APPLICATION SPECIFIC SPINNER BUILD FILE > Example: gtccplm-secure-pipeline-jenkins-pre-build.xml
  - Ant/build/junit-plm-test.xml
  - Ant/build/junit-noframes.xsl
  - <ENVIROMENT>-junit-context.properties Example: dev1- junit-context.properties
  - Jenkinsfile

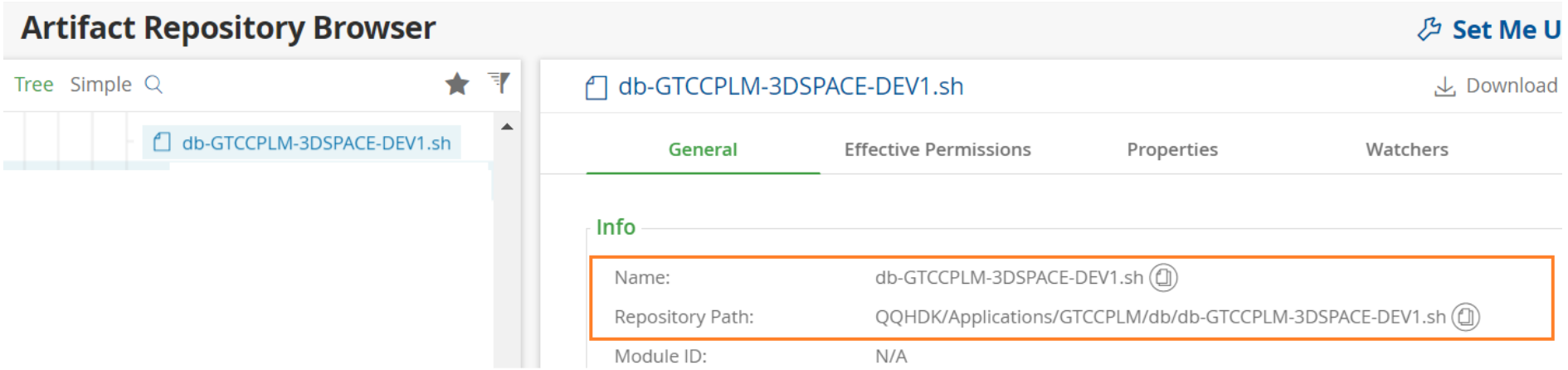
- Required files in Spinner branch
  - Sonar-Project.properties
  - Jenkins file

- *Step#3 Upload Deployment script files to Artifactory*
- You can upload the scripts using either by script upload job or deploy it manually in artifactory.
- File upload path is <REPOSITORY>/Applications/<APPLICTION NAME>/<app|db>

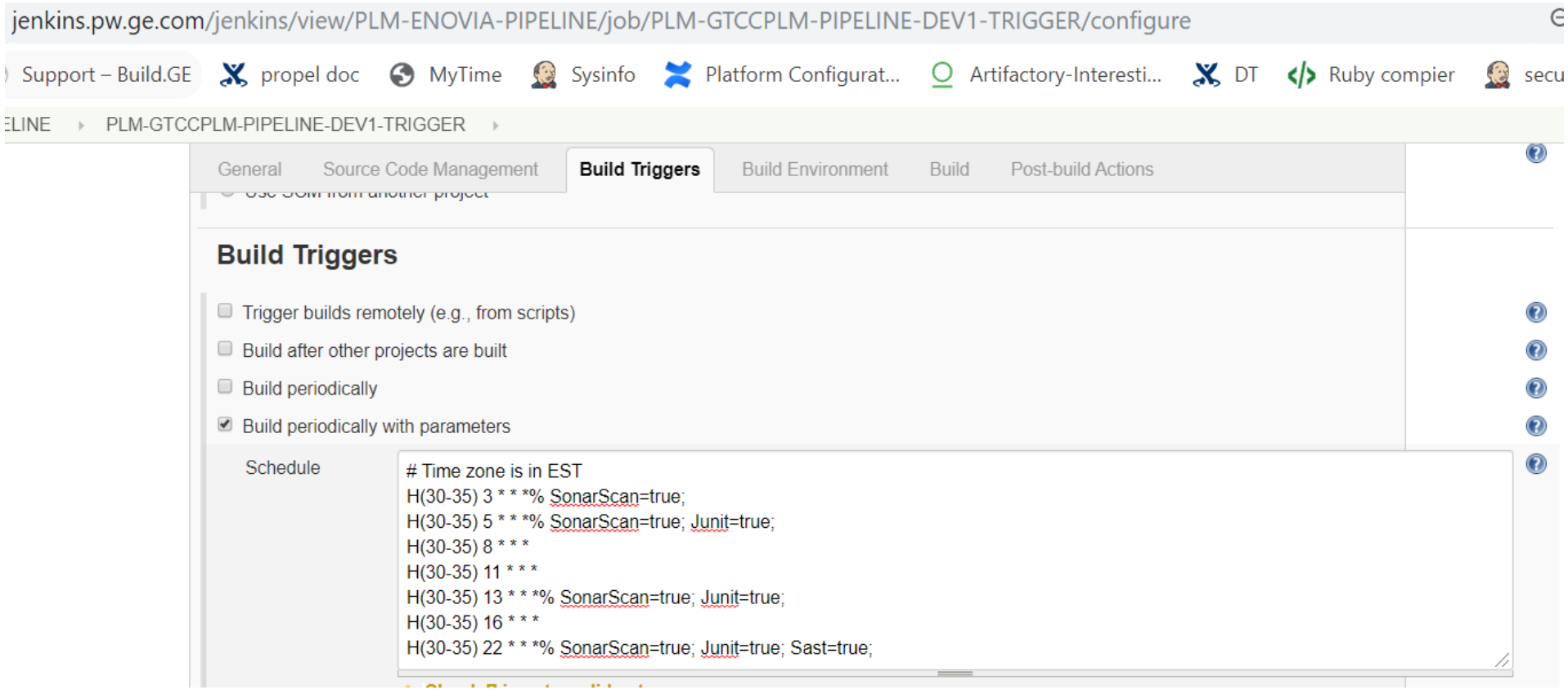
    - GTCCPLM-DEV1 web deployment script file location



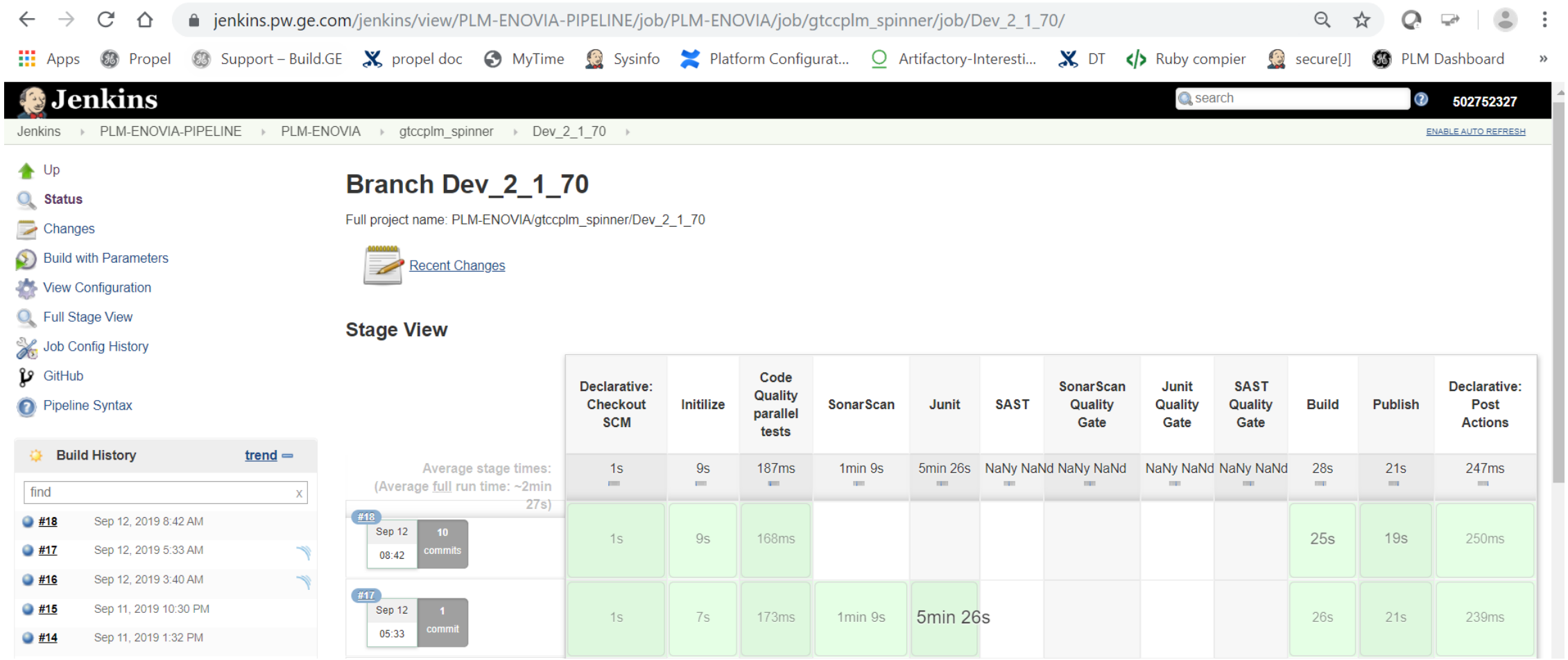
- GTCCPLM-DEV1 Spinner deployment script file location



- ## *Step#4 Create Trigger Job*
  - Recommending to copy existing pipeline trigger job instead of create new one.
  - After create job, modify Git Details, any hardcoded values, email content etc..
    - Example: PLM-GTCCPLM-PIPELINE-DEV1-TRIGGER
  - Schedule the builds using Build Trigger and pass sonar scan , junit and sast values as true if specific build need to go through those code quality checks.



- Run builds and verify the logs for pipeline stages in PLM-ENOVIA GitHub organization job.
- Verify the artifacts in artifactory.
- Verify the build number updates in appconfig.json file
- **Extract the artifacts and compare them with artifacts generated from freestyle job for the same branch. Both should be the same.**



- *Step#5 Enable chef cron tab on new Artifactory*

- If you are using existing Artifactory for deployment then this step can be skipped.
- Otherwise check with EC support team to enable chef cron tab to do deployments from new Artifactory.
- Verify the deployment logs.

GE Internal