

Pipeline Builds for PLM Applications

Index

- Getting Started with Pipeline
 - **Checklist**
 - **Jenkins configuration with tools, plugins and servers**
 - Add GITHUB User credentials
 - Configure Artifactory
 - Generate API key in Artifactory
 - Add Credentials in Jenkins
 - Add Artifactory Server in Jenkins
 - Configure Coverity
 - Overview & Requirements
 - Register your application with Coverity
 - Install Coverity in Jenkins
 - Get Coverity Authentication key
 - Add Credentials in Jenkins
 - Add Coverity connect server in Jenkins

- 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)

Getting Started with pipeline

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

- 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'
```

2. **User credentials**

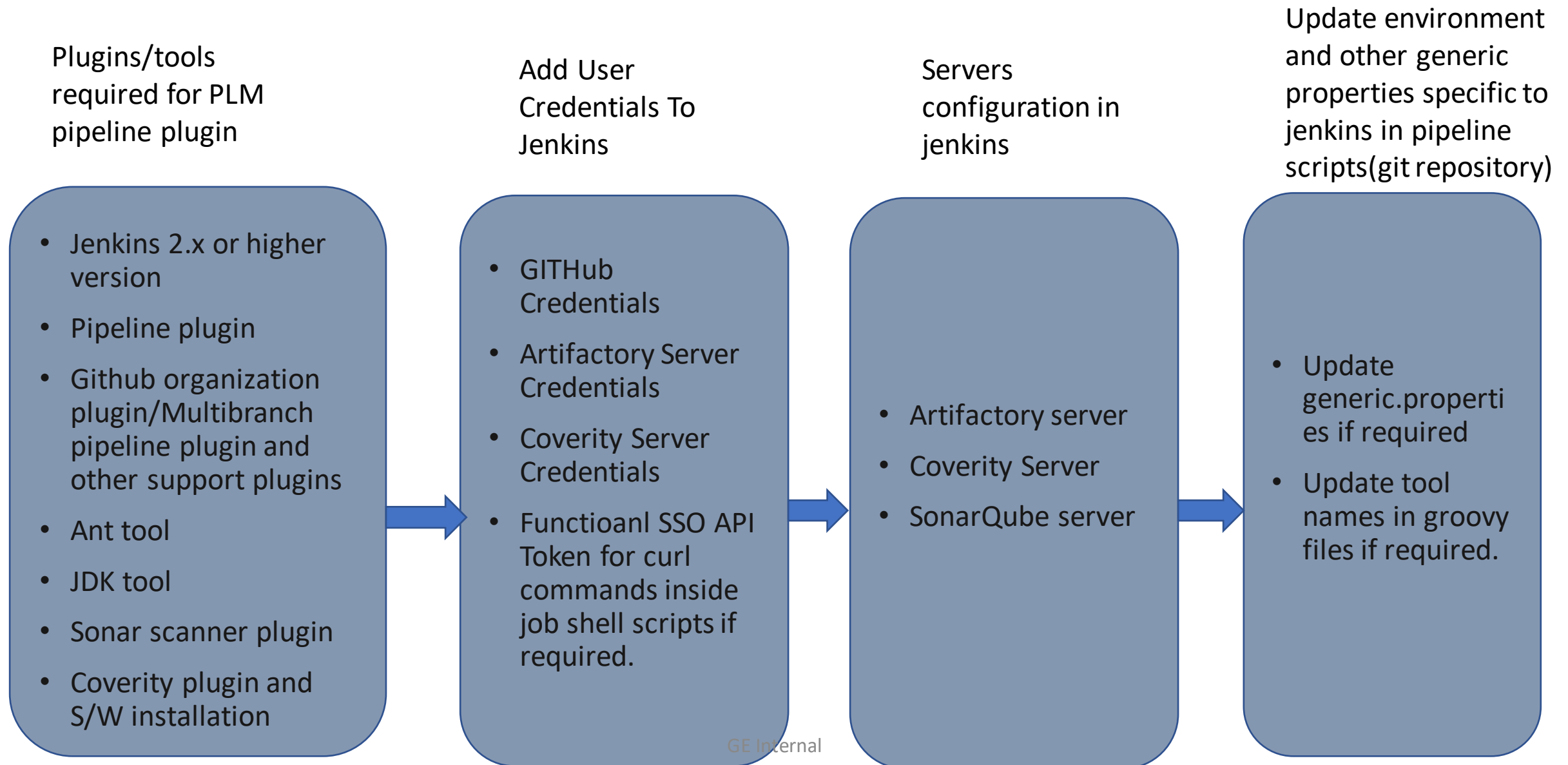
Need to configure credentials in order to connect different servers like GitHub, Artifactory, SonarQube and Coverity .

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 otherwise build will be stopped with errors.

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



Checklist

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



Pipeline_powerJen
kins_Checklist_V1

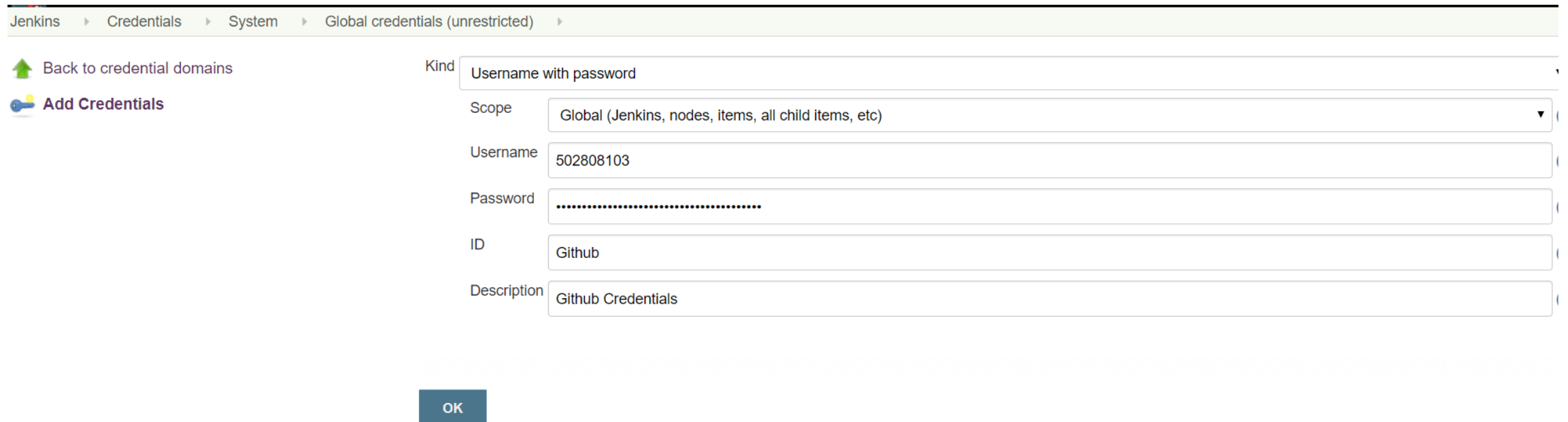
Jenkins configuration with tools and plugins

Add GITHUB User Credentials

(functional SSO)

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.



The screenshot shows the Jenkins web interface for adding a new credential. The breadcrumb trail at the top indicates the path: Jenkins > Credentials > System > Global credentials (unrestricted). On the left sidebar, there is a green arrow icon labeled 'Back to credential domains' and a blue key icon labeled 'Add Credentials'. The main form is titled 'Kind: Username with password'. It contains several input fields: 'Scope' is a dropdown menu currently showing 'Global (Jenkins, nodes, items, all child items, etc)'; 'Username' is a text field containing '502808103'; 'Password' is a text field filled with dots; 'ID' is a text field containing 'Github'; and 'Description' is a text field containing 'Github Credentials'. At the bottom of the form is a blue button labeled 'OK'.

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

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

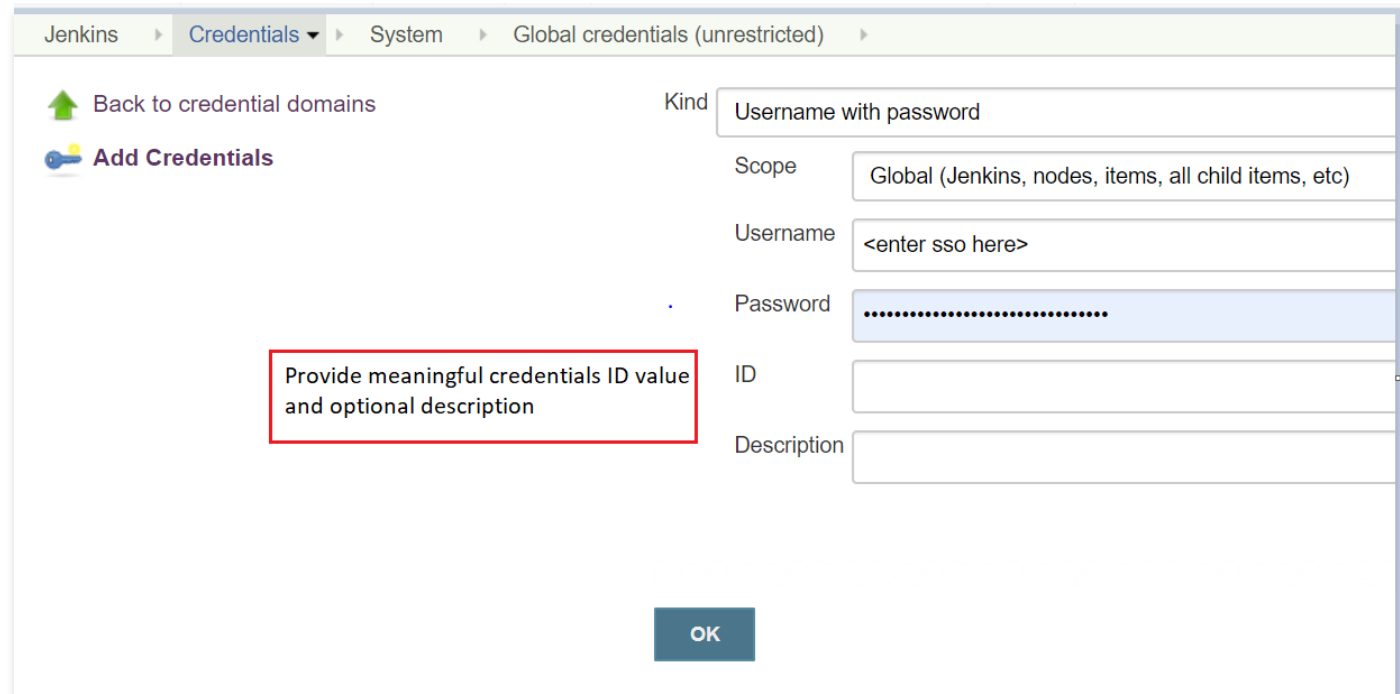
- Login with functional SSO and click on profile.
- To edit your profile, you first need to unlock it by entering your current password and clicking **Unlock**.
- To create an API Key, once you have unlocked your profile, click the "Generate" button next to the **API Key** field.



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.



The screenshot shows the Jenkins 'Add Credentials' form. The breadcrumb navigation at the top reads: Jenkins > Credentials > System > Global credentials (unrestricted). On the left sidebar, there is a green arrow icon labeled 'Back to credential domains' and a blue key icon labeled 'Add Credentials'. The main form area has the following fields:

- Kind:** A dropdown menu with 'Username with password' selected.
- Scope:** A dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected.
- Username:** A text input field containing '<enter sso here>'. There is a small blue dot to the left of this field.
- Password:** A text input field filled with dots, indicating a password.
- ID:** A text input field.
- Description:** A text input field.

A red rectangular box highlights the 'ID' and 'Description' fields with the text: 'Provide meaningful credentials ID value and optional description'. At the bottom right of the form is a blue button labeled 'OK'.

3. Configure Artifactory server in Jenkins

- To 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.

Artifactory

Artifactory servers

☐ Enable Push to Bintray (deprecated)

☒ Use the Credentials Plugin

Artifactory

Server ID

0078001260123214300707

URL

https://central.maven.org/artifactory

Default Deployer Credentials

Credentials

00000000000000000000000000000000 (Artifactory) ▼

Add ▼



Advanced...

Found Artifactory 5.9.7

Test Connection

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

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.

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)

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

Coverity Static Analysis Tools

Coverity Static Analysis Tools installations

Add Coverity Static Analysis Tools



Coverity Static Analysis Tools

Name

cov-analysis-2018.12

Installation Directory

/var/lib/jenkins/cov-analysis-linux64-2018.12



Analysis installation directory has been verified.

Delete Coverity Static Analysis Tools

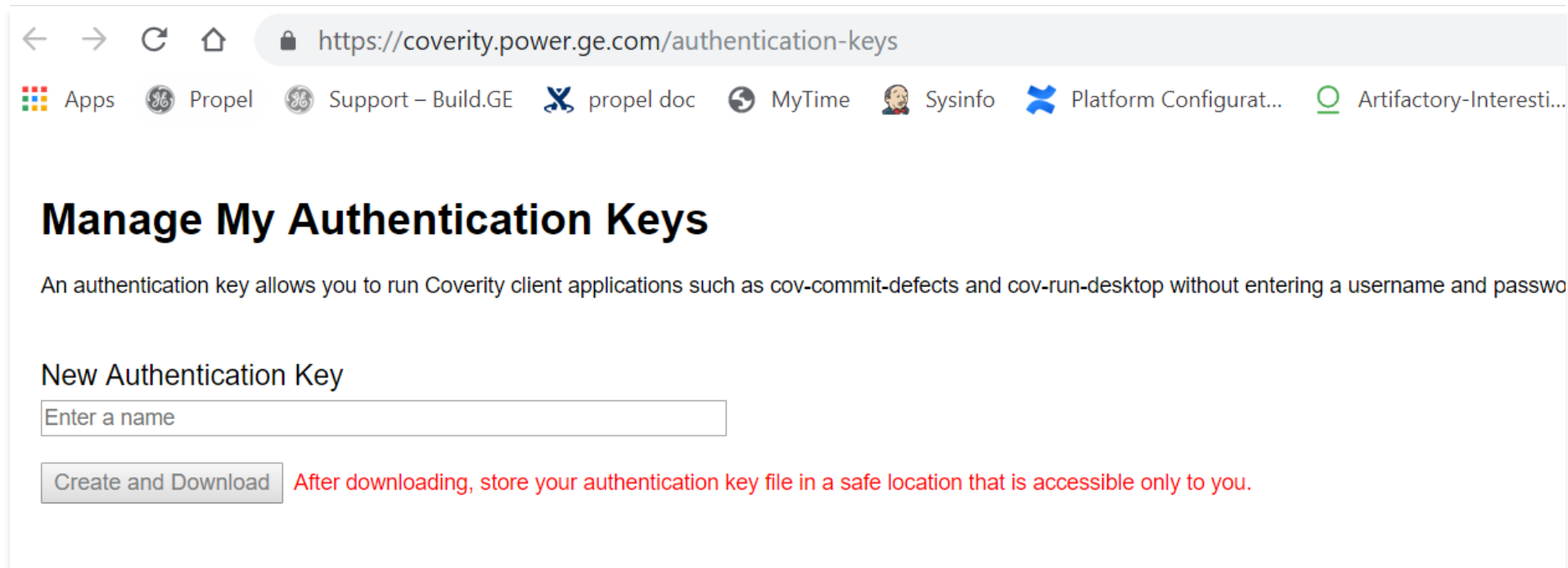
Add Coverity Static Analysis Tools

List of Coverity Static Analysis Tools installations on this system

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

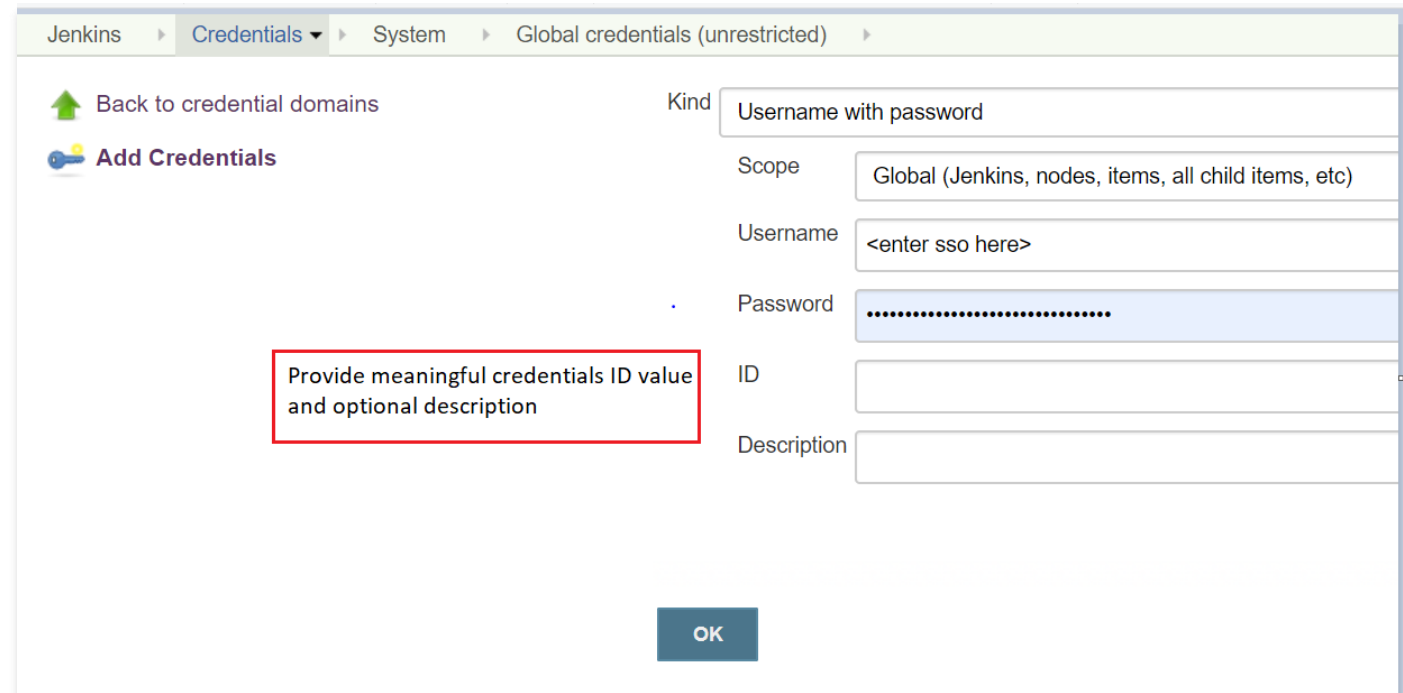


The screenshot shows a web browser window with the address bar displaying <https://coverity.power.ge.com/authentication-keys>. The browser's tab bar includes icons for 'Apps', 'Propel', 'Support – Build.GE', 'propel doc', 'MyTime', 'Sysinfo', 'Platform Configurat...', and 'Artifactory-Interesti...'. The main content area has the heading 'Manage My Authentication Keys' and a subtext: 'An authentication key allows you to run Coverity client applications such as cov-commit-defects and cov-run-desktop without entering a username and password'. Below this, there is a section titled 'New Authentication Key' with a text input field labeled 'Enter a name'. A 'Create and Download' button is positioned to the left of a red text instruction: 'After downloading, store your authentication key file in a safe location that is accessible only to you.'

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.



The screenshot shows the Jenkins 'Add Credentials' form. The breadcrumb trail at the top is 'Jenkins > Credentials > System > Global credentials (unrestricted)'. On the left, there are links for 'Back to credential domains' and 'Add Credentials'. The form fields are as follows:

Field	Value
Kind	Username with password
Scope	Global (Jenkins, nodes, items, all child items, etc)
Username	<enter sso here>
Password
ID	
Description	


A red box highlights the 'ID' and 'Description' fields with the text: 'Provide meaningful credentials ID value and optional description'. An 'OK' button is located at the bottom right.

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 coverity server host and port as 443 as details shared by the coverity server team.
- Select coverity credentials .

Coverity

Coverity Connect instances

 **Coverity** Connect Instance

Name

plm-coverity

?

Host

coverity.plm.com

?

Port

443

?


Use SSL

☒


?

Credentials

502808103/***** (Coverity)

 Add

?

 "502808103" does not have following global permission(s): "Commit to a stream"

Check

Configure Sonarqube

1.Overview

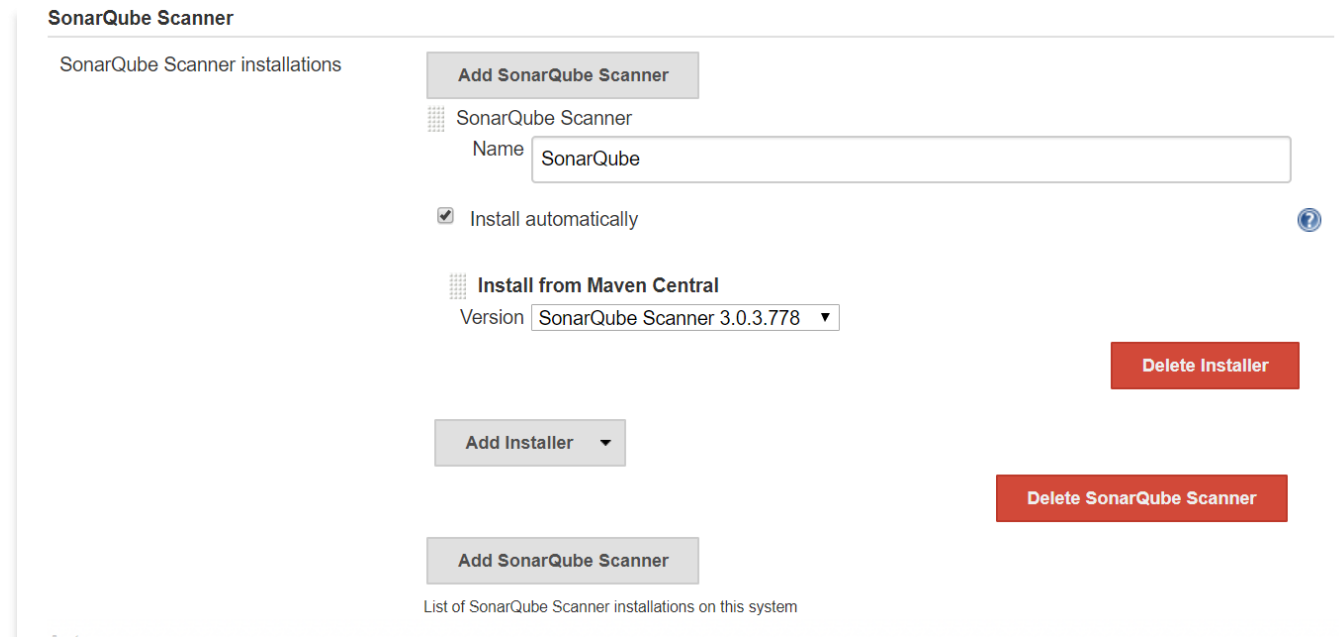
- **Sonar scan is for** static code analysis
- **SonarQube Scanner 3.3** is Compatible with SonarQube 5.6+ (LTS)
- NOLA team provides SonarQube server and SonarQube scanner and you are no need to configure them as they are part of the jenkins image.
- You need to generate SonarQube server user token and update it in SonarQube server configuration.

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.



The screenshot shows the 'SonarQube Scanner' configuration page in Jenkins. The page title is 'SonarQube Scanner'. Below the title, there is a section 'SonarQube Scanner installations'. On the right side of this section, there is a button 'Add SonarQube Scanner'. Below this button, there is a form for adding a new scanner. The form has a 'Name' field with the value 'SonarQube'. Below the 'Name' field, there is a checkbox labeled 'Install automatically' which is checked. Below the checkbox, there is a section 'Install from Maven Central' with a 'Version' dropdown menu showing 'SonarQube Scanner 3.0.3.778'. To the right of the version dropdown, there is a red button 'Delete Installer'. Below the version dropdown, there is a button 'Add Installer' with a dropdown arrow. Below the 'Add Installer' button, there is another 'Add SonarQube Scanner' button. At the bottom right, there is a red button 'Delete SonarQube Scanner'. At the bottom of the page, there is a text label 'List of SonarQube Scanner installations on this system'.

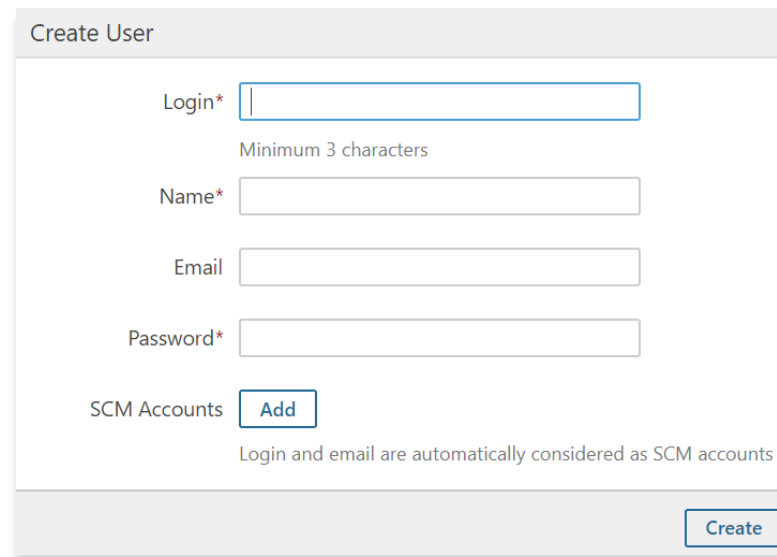
3. Generate SonarQube Server Authentication token and update it in Jenkins

Step#1 Create new user

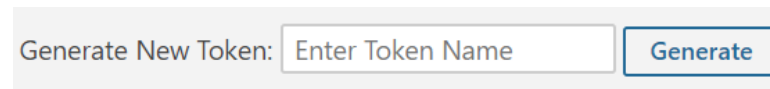
- Navigate through Manage Jenkins -> Configure System
- Find the SonarQube servers section and copy the server URL and open it in browser
- Login into server using admin credentials.
- Navigate through Administration -> Security -> Users -> Create User
- Create new user by providing user name, password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



The screenshot shows the 'Create User' form in SonarQube. It includes input fields for 'Login*', 'Name*', 'Email', and 'Password*'. A note below the 'Login*' field states 'Minimum 3 characters'. There is an 'Add' button next to the 'SCM Accounts' label. At the bottom right, there is a 'Create' button. A footer note states 'Login and email are automatically considered as SCM accounts'.



The screenshot shows the 'Generate New Token' form in SonarQube. It includes an input field labeled 'Enter Token Name' and a 'Generate' button.

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

SonarQube servers

Environment variables

☐ Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

SonarQube

Server URL

https://sonarqube-01-jca11003-002-1ing-progamlvtg

Default is http://localhost:9000

Server authentication token

.....

Create GitHub organization and other util jobs in Jenkins

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.

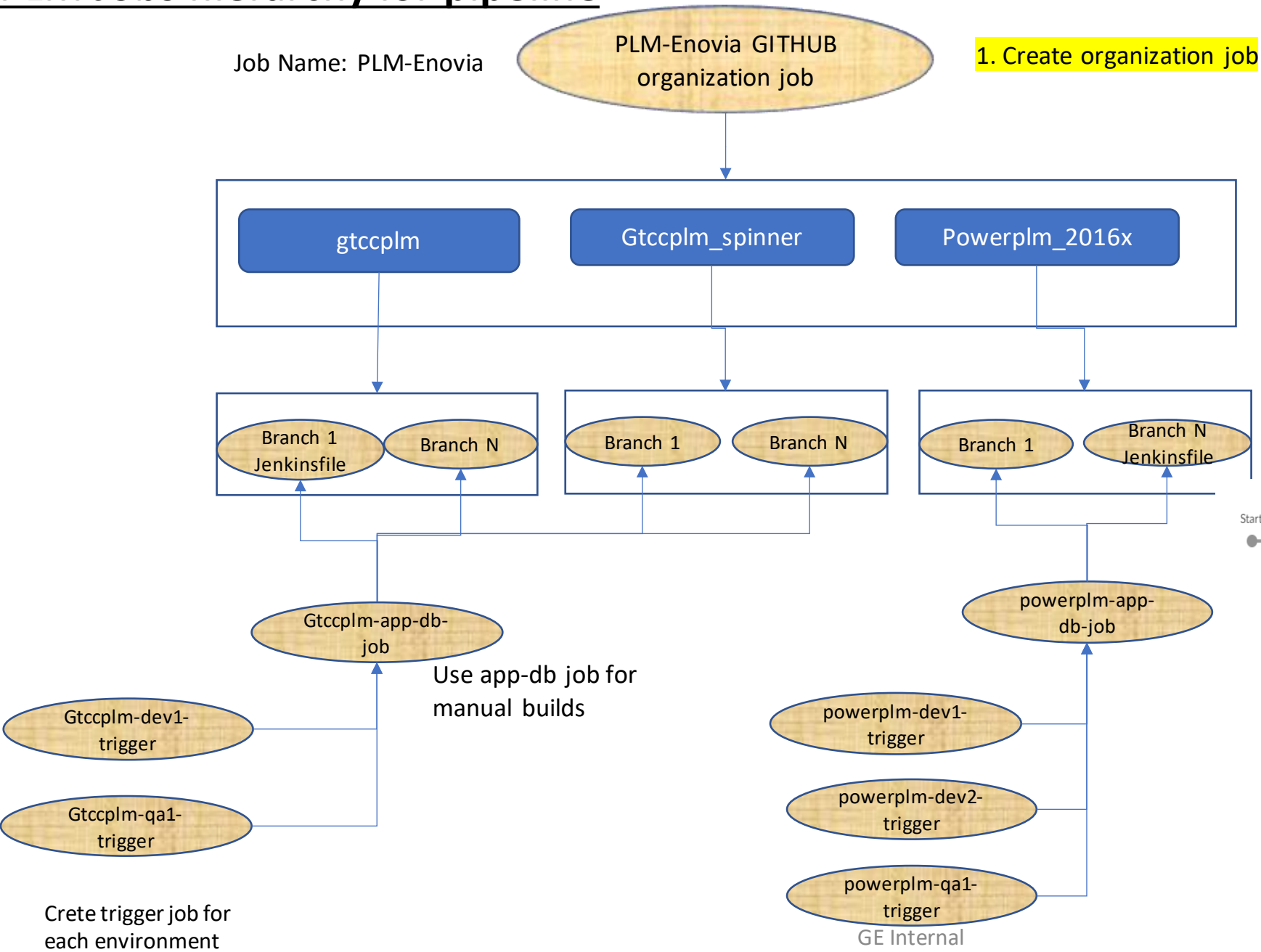
Fundamentally, an organization's repositories can be viewed as a hierarchy, where each repository may have child elements of branches and pull requests.

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.

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.

PLM Jobs hierarchy for pipeline



- 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



Create Organization folder Job

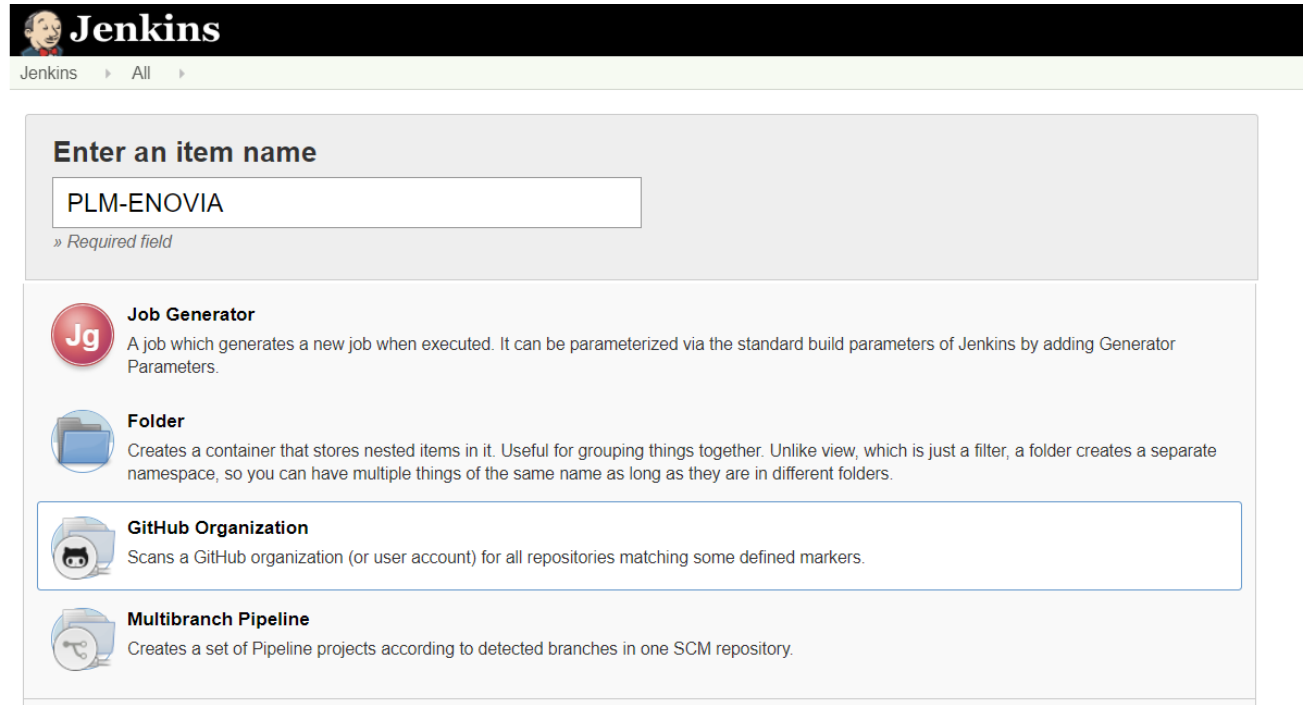
Create organization job

- 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.

Demonstration

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



Jenkins

Jenkins > All >

Enter an item name

PLM-ENOVIA

» Required field

- Job Generator**
A job which generates a new job when executed. It can be parameterized via the standard build parameters of Jenkins by adding Generator Parameters.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

- 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.

The screenshot shows the 'Projects' configuration page. It includes the following fields and sections:

- GitHub Organization**: A dropdown menu.
- API endpoint**: A text field with the value 'Enterprise GitHub (https://github.build.ge.com/api/v3)'.
- Credentials**: A dropdown menu with the value '502808103/***** (github)' and an 'Add' button.
- User**: A text field with the value '502808103'.
- Owner**: A text field with the value 'PLM-ENOVIA'.
- Behaviors**: A section containing:
 - Repositories**: A section with a 'Filter by name (with regular expression)' label and a text field containing the regular expression 'Renewablesplm-2013X|Renewablesplm-2013X-Spinner|gtccplm|gtccplm_spinner|powerplm-2016X|'.
 - Within repository**: A section with a text field.
 - Discover branches**: A section with a text field.

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-ENOVIA organization

 **PLM-ENOVIA**





















Repositories (5)

S	W	Name ↓	Description
		gtccplm	
		gtccplm_spinner	
		powerplm-2016X	Web Code
		Renewablesplm-2013X	WEB Code
		Renewablesplm-2013X-Spinner	Spinner Code

Branches under gtccplm which contains Jenkinsfile at branch root directory

 **gtccplm**

Branches (5) Pull Requests (0)

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
		Dev_2_1_50	1 mo 4 days - #37	1 mo 4 days - #36	5 min 22 sec	 
		Dev_2_1_60	8 min 9 sec - #103	12 days - #61	6 min 6 sec	 
		Release_2_1_30_DEV2	1 mo 4 days - #15	1 mo 4 days - #17	3 min 36 sec	 
		Release_2_1_40_DEV2	N/A	N/A	N/A	 
		secure_pipeline_Dev_2_1_50	1 mo 13 days - #78	1 mo 14 days - #70	9.9 sec	 

Icon: S M L

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

- 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

Add New Application to pipeline

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.

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 artifacts 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.

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 artifacts 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

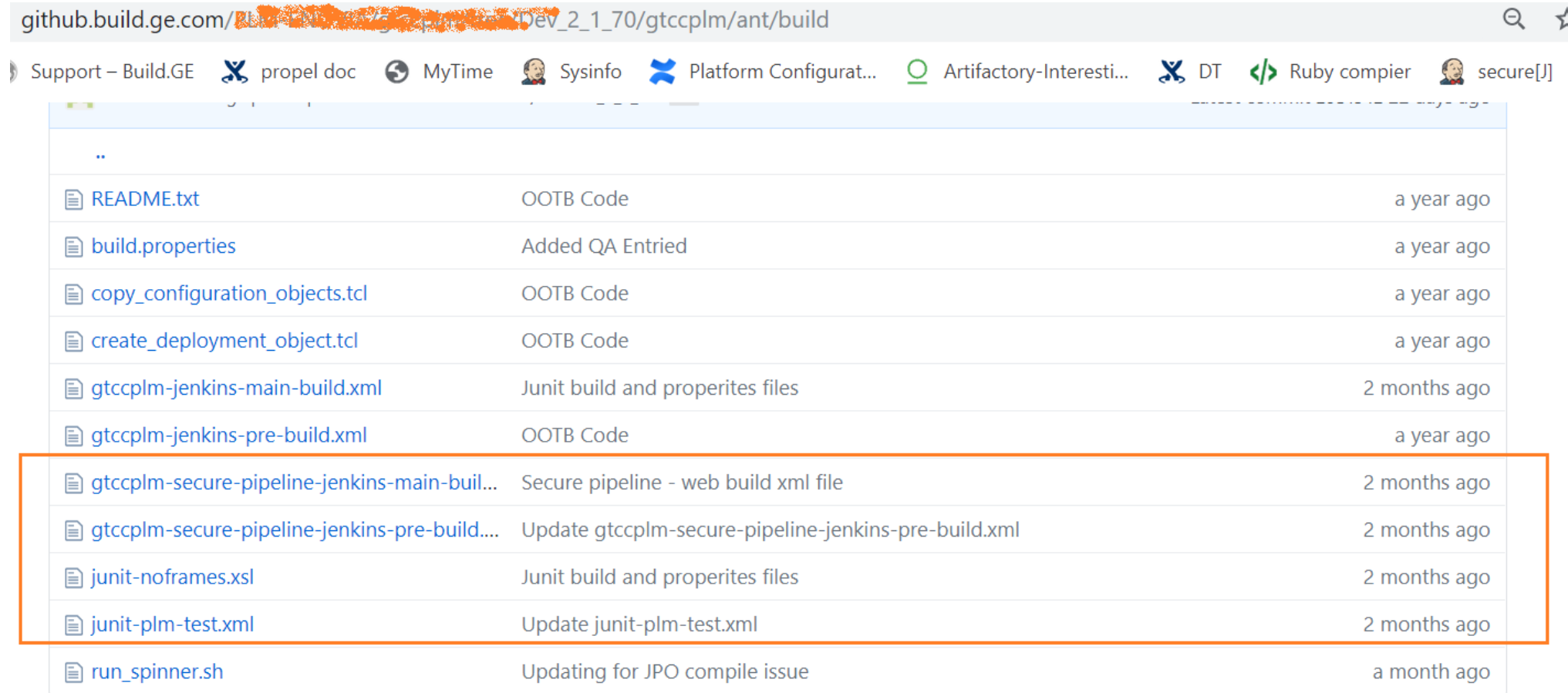
- Add junit-plm-test.xml in ant/build folder . This file is used to run junit test cases on spinner code. May need some changes in file if application specific parameters are used.
- Add junit-noframes.xsl file used as css for report formatting. File location : ant/build
- Add environment specific junit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

Screenshot#



github.build.ge.com/PLM-2025-07-25-01/Dev_2_1_70/gtccplm/ant/build

Support – Build.GE | propel doc | MyTime | Sysinfo | Platform Configurati... | 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.xml	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 modify 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.

- *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 number 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 Configurati... Artifactory-Interesti... DT Ruby compier secure[J] PLM Dashboard

search 502752327

PLM-ENOVIA-PIPELINE

add description

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-securepipeline.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-securepipeline.properties

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

- Screenshot(s)#


 **PLM-ENOVIA** / **plm-secure-pipeline-scripts**



 Watch 2  Star 0  Fork 1




 Code  Issues 0  Pull requests 0  Projects 0  Wiki  Insights  Settings

Branch: power_jenkins_... Find file Copy path

[plm-secure-pipeline-scripts](#) / [resources](#) / [applicationproperties](#) / [gtccplm](#) / [gtccplm-web-securepipeline.properties](#)

 502752327 Update gtccplm-web-securepipeline.properties 7962f2c a day ago

2 contributors  

49 lines (37 sloc) | 2.19 KB Raw Blame History   

```
1  #GTCCPLM web properties Common for all Enviornments for GTCC Application web builds
2  application = GTCCPLM
3  publishType = app
4
5  # web build file and ant targets.
6  buildFile = gtccplm/ant/build/gtccplm-secure-pipeline-jenkins-main-build.xml
7  build_targets = pre-set-up set-up cleanup set-flags mkdirs client war tar-war summary
8
9  # Source file pattern using to copy artifacts
10 application_artifacts_filepattern = gtccplm/war/build/*.tar.gz
11
12 # Properties using for Coverity
13 coverity_project_name =1100214481_PowerPLM_GTCC
14 coverity_stream=GtccPLM_<BRANCH>
15 coverity_view=Untriaged
16
17 base_folder=gtccplm
18
19 # lastDeployed.jsp file location
20 lastdeployed_jsp=war/web/GEMonitor/LastDeployed.jsp
21
```

<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki


📊 Insights

⚙ Settings

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

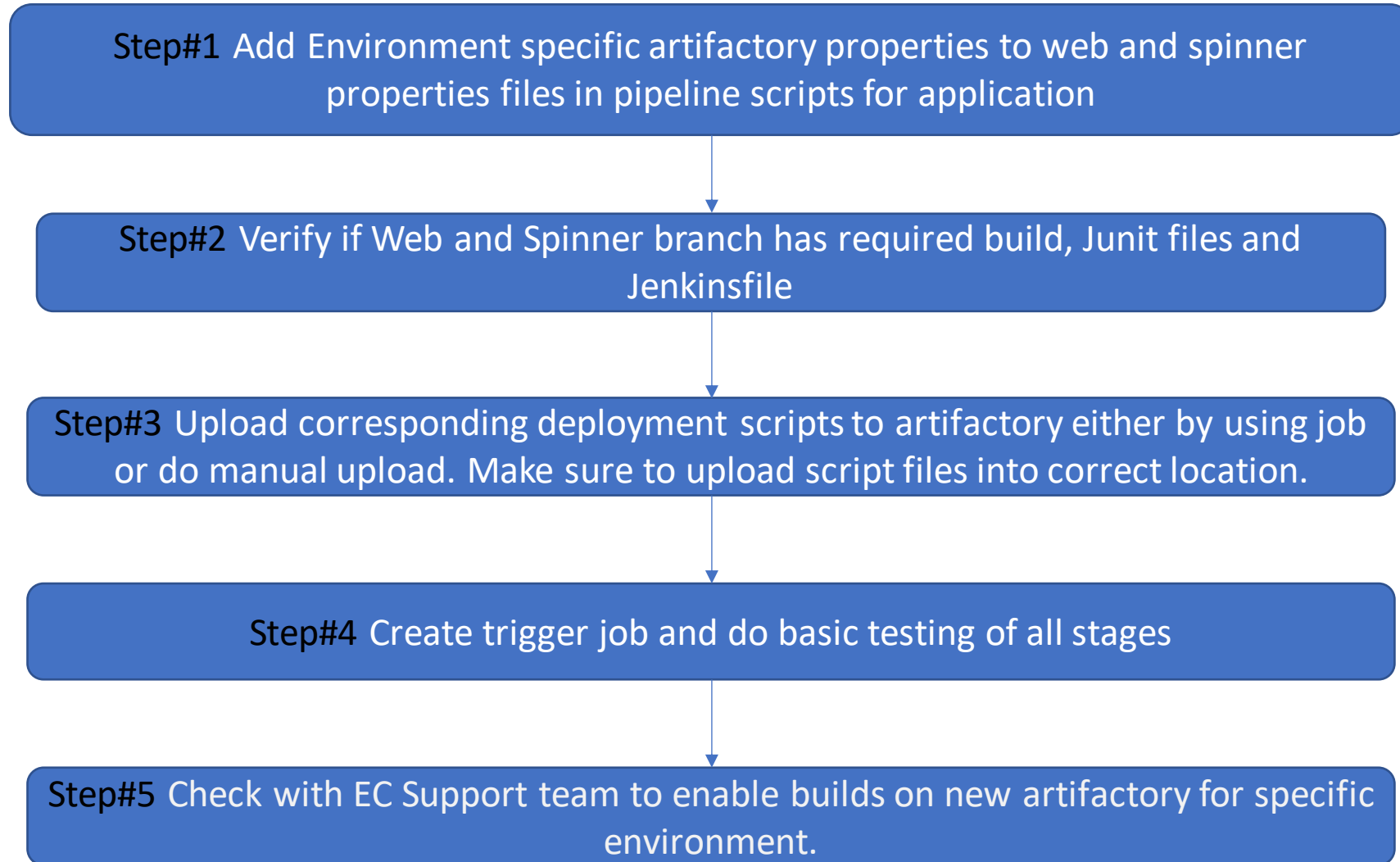
History



```
1 #GTCCPLM Spinner properties Common for all Enviornments for GTCC Application spinner builds
2 application = GTCCPLM
3 publishType = db
4 git_base_folder=gtccplm_spinner
5
6 # Spinner build file and ant targets.
7 buildFile = gtccplm-secure-pipeline-jenkins-pre-build.xml
8 build_targets = pre-set-up set-up custom-jar-deploy copy-spinner tar-spinner
9
10 # Source file pattern using to copy artifacts
11 application_artifacts_filepattern = *tar.gz
12
13 # properties using to web code checkout to copy build.xml and other files required for spinner build.
14 webcode_git_repo_url = https://github.build.ge.com/PLM-ENOVIA/gtccplm.git
15 web_repo_base_folder = gtccplm
16 web_repo_name = gtccplm
17
18 # Properties using for Coverity
19 coverity_project_name = 1100214481_PowerPLM_GTCC
20 coverity_stream=GtccPLM_Spinner_<BRANCH>
21 coverity_build_targets = pre-set-up set-up custom-jar-deploy copy-spinner spinner copy_all_ino_spinner coverity
```

Add New Environment

Adding New Environment to Pipeline



GTCCPLM-DEV1 Demonstration

- Step#1 Add Environment specific artifactory properties to pipeline

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

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

Following are the web entries corresponding to DEV1:

```
23 #Following properties are specific to Enviroment. Its required to run builds on specific envioronment.
24 # Always start property name with environment as its configured in trigger job and append it with _ (Ex: DEV1_)
25 # <env>_artifactory_appconfig_file is the appconfig.json file location path in artifactory (Mandatory)
26 # <env>_artifactory_artifact_file path is used as target to copy artifacts into artifactory (Mandatory)
27 # <env>_artifactory_deployment_script property is optional. it will not impact the build. Its using for maintaining and providing correct
28
29 #DEV1  properties
30 DEV1_artifactory_appconfig_file=GTCCPLM/GTCCPLM-3DSPACE-DEV1/app/appConfig.json
31 DEV1_artifactory_artifact_file=GTCCPLM/app/<BUILD_NUMBER_TO_REPLACE>/gtccplm.tar.gz
32 DEV1_artifactory_deployment_script=GTCCPLM/app/app-GTCCPLM-3DSPACE-DEV1.sh
33
```

- **File Name : gtccplm-Spinner-securepipeline.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

Artifact Repository BrowserSet Me UpDeploy

Tree Simple

app-GTCCPLM-3DSPACE-DEV1.sh

app-GTCCPLM-3DSPACE-DEV1.sh

Download View Acti

General

Effective Permissions

Properties

Watchers

Info

Name:

app-GTCCPLM-3DSPACE-DEV1.sh

Repository Path:

QQHDK/Applications/GTCCPLM/app/app-GTCCPLM-3DSPACE-DEV1.sh

Module ID:

N/A

Deployed By:

502752327

Size:

5.99 KB

- GTCCPLM-DEV1 Spinner deployment script file location

Artifact Repository Browser

Tree Simple

db-GTCCPLM-3DSPACE-DEV1.sh

db-GTCCPLM-3DSPACE-DEV1.sh

GeneralEffective PermissionsPropertiesWatchers

Info

Name:db-GTCCPLM-3DSPACE-DEV1.sh

Repository Path:QQHDK/Applications/GTCCPLM/db/db-GTCCPLM-3DSPACE-DEV1.sh

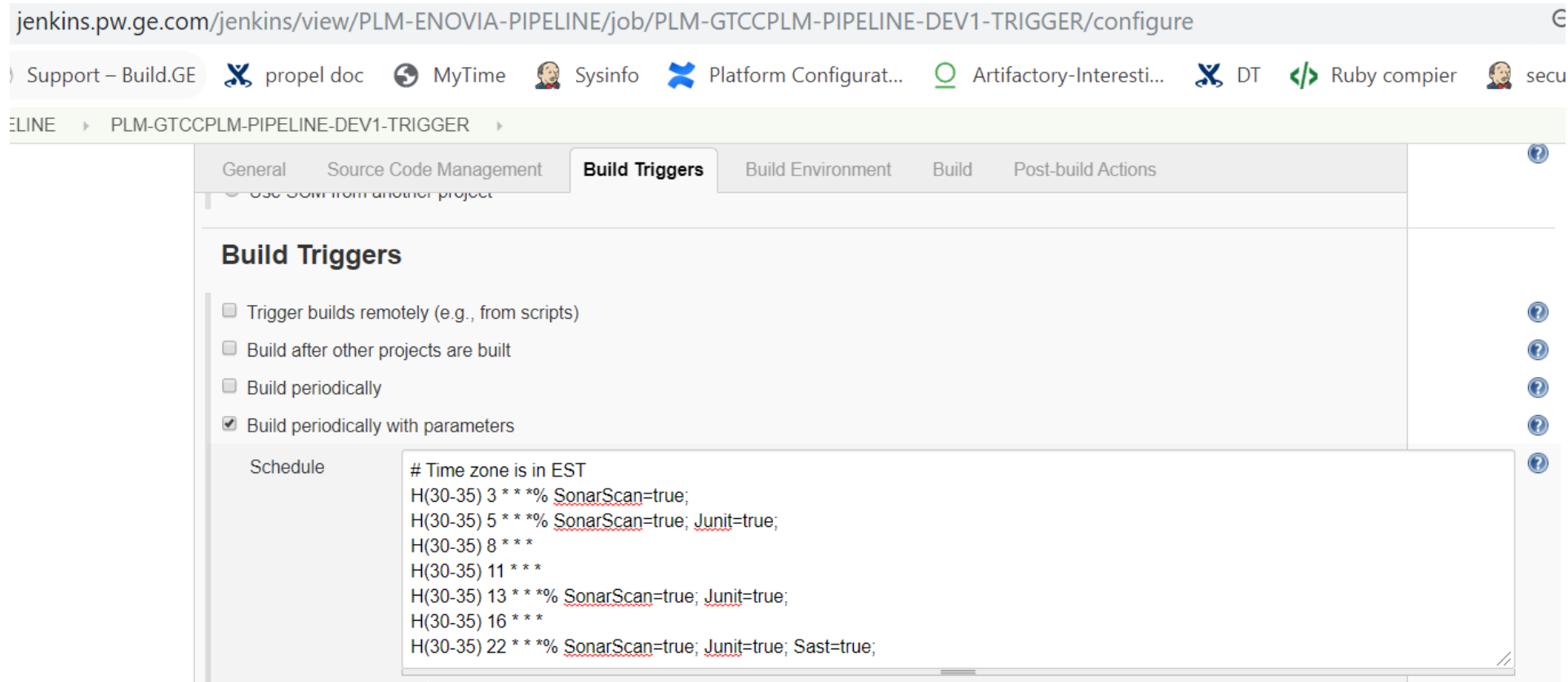
Module ID:N/A

Set Me U

Download

• 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.



The screenshot shows the Jenkins configuration page for the job 'PLM-GTCCPLM-PIPELINE-DEV1-TRIGGER'. The browser address bar shows the URL: `jenkins.pw.ge.com/jenkins/view/PLM-ENOVIA-PIPELINE/job/PLM-GTCCPLM-PIPELINE-DEV1-TRIGGER/configure`. The page has a navigation bar with links to 'Support - Build.GE', 'propel doc', 'MyTime', 'Sysinfo', 'Platform Configurat...', 'Artifactory-Interesti...', 'DT', 'Ruby compier', and 'secu'. Below the navigation bar, the job name 'PLM-GTCCPLM-PIPELINE-DEV1-TRIGGER' is displayed. The configuration tabs include 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Build Triggers' tab is selected, showing the 'Build Triggers' section. It contains a list of checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', and 'Build periodically with parameters'. The 'Build periodically with parameters' checkbox is checked. Below this, the 'Schedule' section is visible, containing a text area with the following content:

```
# Time zone is in EST
H(30-35) 3 * * * % SonarScan=true;
H(30-35) 5 * * * % SonarScan=true; Junit=true;
H(30-35) 8 * * *
H(30-35) 11 * * *
H(30-35) 13 * * * % SonarScan=true; Junit=true;
H(30-35) 16 * * *
H(30-35) 22 * * * % SonarScan=true; Junit=true; Sast=true;
```

- 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.**

jenkins.pw.ge.com/jenkins/view/PLM-ENOVIA-PIPELINE/job/PLM-ENOVIA/job/gtccplm_spinner/job/Dev_2_1_70/

Apps Propel Support – Build.GE propel doc MyTime Sysinfo Platform Configurati... Artifactory-Interesti... DT Ruby compier secure[J] PLM Dashboard

Jenkins

search 502752327

Jenkins > PLM-ENOVIA-PIPELINE > PLM-ENOVIA > gtccplm_spinner > Dev_2_1_70 > [ENABLE AUTO REFRESH](#)

Up

Status

Changes

Build with Parameters

View Configuration

Full Stage View

Job Config History

GitHub

Pipeline Syntax

Branch Dev_2_1_70

Full project name: PLM-ENOVIA/gtccplm_spinner/Dev_2_1_70

[Recent Changes](#)

Stage View

Average stage times:
(Average full run time: ~2min 27s)

	Declarative: Checkout SCM	Initilize	Code Quality parallel tests	SonarScan	Junit	SAST	SonarScan Quality Gate	Junit Quality Gate	SAST Quality Gate	Build	Publish	Declarative: Post Actions
#18 Sep 12, 2019 08:42 10 commits	1s	9s	187ms	1min 9s	5min 26s	NaNy NaNd	NaNy NaNd	NaNy NaNd	NaNy NaNd	28s	21s	247ms
#17 Sep 12, 2019 05:33 1 commit	1s	9s	168ms							25s	19s	250ms
	1s	7s	173ms	1min 9s	5min 26s					26s	21s	239ms

Build History

find x

- #18 Sep 12, 2019 8:42 AM
- #17 Sep 12, 2019 5:33 AM
- #16 Sep 12, 2019 3:40 AM
- #15 Sep 11, 2019 10:30 PM
- #14 Sep 11, 2019 1:32 PM

- *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.