1. **PRE-REQUSITES:**

* Working automation testing project, Smoke or Regression or functional, which the team desired to run periodically (Daily, Weekly)
* User has access to GitLab and automation test project available in GitLab Repo

**Pre-req for project automation team with active automated tests in GitLab but do not have a Gitlab pipeline configured:**

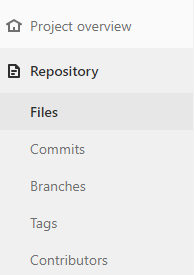
Step 1: Create a basic GitLab pipeline

1. **CONFIGURING THE PIPELINE:**

Procedure for project (LOB) team with automated test cases to configure a pipeline:

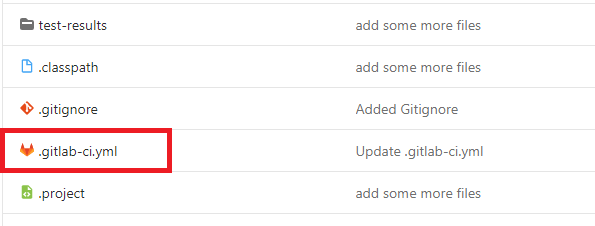
4.1 Config Setup Steps:

1. Access the repository using the base GitLab repository URL:



1. Traverse to     Project Overview  à Repository  à Files

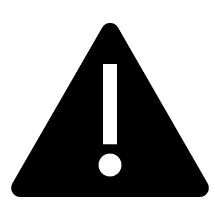
All the files under Project repository will be displayed in the right pane, like below:



1. Click on   [**.gitlab-ci.yml**](https://gitlab.lfg.com/Manoj.Kumar/quantum_cicd/-/commit/7a3a3f2ce15be5d1701684fc89ca38723deb9f2a)file to open the file where the CI is configured.
2. Traverse to [https://gitlab. .com/ -base-install/ /-/blob/master/.gitlab-ci.yml](https://gitlab.lfg.com/quantum-base-install/quantum/-/blob/master/.gitlab-ci.yml) . This is the sample project under project.
3. Copy and Paste the content of the below YAML file to your YAML (YAML file under your project) file under your project repository pipeline.



4.2 Project Information:

 **DO NOT MODIFY**

**The Include section contains the information explained below. The contents should not be modified.**

**Project –**This is the location of the project where you will find the master script. This will be read-only and is for information only.

**ref –**The branch where the master script YAML is in this gitlab repository.

**file –**Current YAML file will refer this file while executing the test cases.

4.3 Variables Information:

**THIS SECTION NEEDS YOUR ATTENTION / MODIFICATION**

**The variables section contains the information explained below :**

**RunList – (Mandatory field)**Feature file annotation for execution. This section can have multiple entries comma-separated. This field cannot be empty.

***Example values :***   “Smoke”

**EnvName – (Mandatory field)**This is the environment where the user desires to run the automated tests. Only 1 entry at one time.

***Example values : “***UAT” OR “QA” OR “PROD”

**SuccessThreshold – (Mandatory field)**This value defines the pass % for the automation execution of your suite. If the actual pass % is above the threshold % value, the pipeline result will be a pass. Or else the pipeline will fail. You can have only 1 entry at one time

***Example values :*** Any numeric value in terms of %.

**DeviceFile – (Mandatory field)**The path inside the project folder where the device file is stored.

4.4 User Parameters:

**UserParam – (Optional field)**User Defined Parameters - This is the field that the user can input any values that needs to be used by the tests. An option provided to the user to specify any        **user-defined-params** that would be used in code while running the tests.

Format of the field – **key : value**

***Example values:***

UserParam : “BrowserName:chrome”

UserParam : “MobileOS:NotRunningMobileTestCase”

UserParam : “DeviceModel:NotAMobileDevice”

* Note : More than one key:value pair can be supplied. Each pair needs to be separated using ‘,’(comma).

UserParam: "BrowserName:chrome,DeviceModel:NotAMobileDevice,myParam:100"

4.4.1 How to Use UserParam:

The values entered in this field will be available for use in the tests.

by calling the method **CommonUtilities**.**userArgsToMap();**

It returns a map and the values can be accessed via the keys.

HashMap<String, String> userParamAsMap = CommonUtilities.*userArgsToMap*();

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

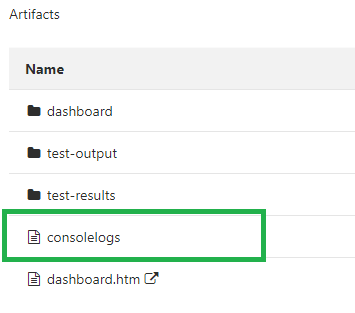
4.5 ConsoleLogs:

**ConsoleLogs:** This field is used to specify how the user wants the execution logs to be documented. User has 2 options.

1. **ConsoleLogs : “”** (Empty string) – All the console output will be displayed on the console of the pipeline.

2. **ConsoleLogs : "-l consolelogs"** – Execution output will be stored in a file called ‘Consolelogs’ under the **artifacts** folder of the project.

Under CI-CD -> Jobs -> Select the job you have run (Status) -> click on Browse button   (on the right side of page). Artifacts folder will be displayed.



User can also download the files on clicking   button.

**Java Coding Standards in Selenium**

1. Static Variable and Constant declarations  
   Static variables should be declared at the beginning of the class. They should not be declared anywhere else in the class.All the CONSTANTS should be uppercase separated with underscore.All variables should be in alphabetic order.
2. Naming Convention  
   Use meaningful names to make the code readable.  
   a. Package name, Method name and Variables.  
   Package name, Method name, Objects and Variables should be in camelCase (A naming convention in which each word within a compound word is capitalized except for the first word.)  
     
   b. File name  
   File name and class name should be in UpperCamelCase (A naming convention in which each word within a compound word is capitalized including for the first word.)
3. Spacing  
   a. General Spacing.  
   i There should be one-tab difference between class indentation and methods.  
   ii There should be one-tab difference between method and method contents.  
   iii There should be a one blank line between each method.  
   iv In the start of the Method the Curly brackets should be in the same line of the method declaration with 1 space separating it  
   . The ending curly bracket should be in same indentation as the first word of the method.  
   . This convention applies to methods and to all logic like try and if blocks.  
   v There should be a space in between the variable name and variable declaration.  
   vi There should be a space after each parameter in the method declaration. String user, String pwd  
     
   vii All the empty blocks must be in following format.  
     
   b. Spacing inside Method  
   i Use one-tab Rule inside Method.  
   ii There should be one blank line before a new logic block inside the method.  
   . Unless it is a nested logic block, if nested, a one-tab replaces the need for one blank line. (The tab allows for better readability while streamlining the code).  
     
   iii Spacing on indenting messages / streaming complex methods  
   The example below should be used only when there are more than 5 appends. There is no hard and fast rule, but by self-conscious save lines. 2 or 3 appends generally should be on a single line.
4. Annotations  
   Use annotations wherever necessary. Whether in class, method or constructor immediately after documentation block. Each annotation should be on one line.  
     
   Note: For Feature files, each annotation should be on the same line separated by a space.
5. Error/Exception Handling  
   a. try, catch, finally block  
   Be sure to use try catch blocks wherever possible in and finally blocks if required.  
     
   i Exception occurs in try block and is handled in the catch block: If a statement in try block raises an exception, then the rest of the try block doesn’t execute and control passes to the corresponding catch block. The catch block should include any error handling. After executing the catch block, the control will be transferred to the finally block if it exists. Then the rest program will be executed.  
   ii Exception doesn’t occur in try-block: In this case catch block never runs. The catch block only runs when there is an exception. If the optional finally block exists, it will be executed followed by rest of the program.  
   b. Union catch blocks  
   Catch block can be union format to handle multi exceptions  
     
   c. throw/throws clause  
   Exception must be declared in throws clause, and issued in throw clause.  
     
   d. Be sure to add failed(), critical(), or logDDFEMsg() to fail the test.  
     
   e. Error Handling to ensure fails occur in Common methods Common methods for multi teams/projects return the status of execution to calling method, and calling method handle it based on the error type.

## Prerequisites

• Unzip command should be recognized in windows command prompt to install eclipse via EclipseInsaller.  
• Install the Eclipse in the location with admin rights for proper working of all supported plugins.  
• **Newer versions of Eclipse do NOT support GIT projects larger than 2GB.** Eclipse will give the following error: “Object too large". Please ensure GIT projects do not exceed this size. Image and results should be stored outside of GIT to prevent rapid file size growth.

## Steps to Install Java

Check Java installation and version and proceed to next step when java 1.8 or higher is not installed in your machine. To check java version installed in your machine, Open windows command prompt (windows start button -> cmd) and type “Java -version”.

a) To install Java 1.8 version

Download URL: download the Java SE 8u251 from [JDK Download](http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html) or GIT clone from [Java-JDK 8.0](https://gitlab.lfg.com/quantum-installers/java/java-jdk-8.0)  
Download & Install both JDK and JRE from the above URL

* click on the download symbol and save the file in zip format in your local directory
* Unzip the file. Right click “jdk-8u20-windows-x64.exe” and Run as administrator.

b) Set the JAVA\_HOME as shown below:

Go to Control Panel\All Control Panel Items\User Accounts using Explorer (not Internet Explorer!)  
OR

* click on the Start button
* click on your picture
* Change my environment variables
* New …
* (if you don't have enough permissions to add it in the System variables section, add it to the User variables section)
* Add JAVA\_HOME as Variable name and the JDK location as Variable value as per your settings > OK
* Add your Java path to the path Environment variable as below
* Add the value %JAVA\_HOME%\bin to the path environment variable as shown below
* (if you don't have enough permissions to add it in the System variables section, add it to the User variables section)
* Verify java installation: Open a new console (cmd) Type java -version Expected output:

## Eclipse Installation Steps:

EclipseInstaller.exe will allow all users to install eclipse with all the required preconfigured plugins and settings for automation testing with the Framework. **This will not affect any existing installed Eclipse versions in your local machine, you can have multiple copies of Eclipse in the same machine.** As long as they are each installed to a unique path.

The Eclipse Installer Exe will install a new version of Eclipse without modifying or uninstalling any existing eclipse versions. If you want to remove an older version of Eclipse, delete the entire directory of where Eclipse is located (e.g., eclipse installation directory generally found in " C:\Users<UserID>\Eclipse" – delete the folder), there is no need to run any uninstaller.

Follow below steps to install

1. Copy the newest EclipseInstaller.EXE file from \\ne1nsp01\automation\_tools\Eclipse\2020-06 to your local machine (Desktop folder).  
   **(It is recommended to close all running programs with max CPU utilization, to increase installation speed)**  
     
   If you do not have access to the EclipseInstaller.EXE file path then please open a SNOW form to request access:
   * SNOW URL=
   * Request Type='Add user(s) to existing Lan/Active Directory Group(s)'
   * Select the Group='Automation\_Tools\_Read'
   * User=Your Lan ID
2. Right click on EclipseInstaller.exe and click on “Run as Administrator”  
     
   Below confirmation window appear once installation completed. Just click on “OK”
3. Installer will create “eclipse” folder in user specified location or by default “C:\Users<USERID>\Eclipse”
4. Eclipse folder contains below folders:  
   a. eclipse – Contains eclipse configurations  
   b. eclipse/LOGS – folder used to store error logs
5. Before starting Eclipse, confirm JDK version 1.8 is installed. Refer to “Steps to install Java”
6. Navigate to “C:\Users<USERID>\Eclipse" or user specified location provided in step 3 and click on eclipse(eclipse.exe) to open Eclipse application. If there are any errors, please refer to the [Trouble shooting](#trouble-shooting) section of this guide.
7. When Eclipse starts up, you will be prompted to select workspace. The default workspace location is “C:\User<userID>\workspace”. If new workspace to be created in specific location, then click on browse to navigate to that specific folder and click on “Launch”.  
   **NOTE: If user want to select the existing workspace, Delete the “.metadata” folder in the workspace before proceeding to “Launch”.**
8. Eclipse is launched in selected workspace  
     
   Eclipse installation is completed successfully with below configurations  
   a. Eclipse Version: 2020-06 or 2019-06  
   b. Just for reference, some of the Plugins Details included in EclipseInstaller:

| **Plugin Name** | **Version** | **Description** |
| --- | --- | --- |
| TestNG | 7.0.0 | Plugin to run TestNG tests from Eclipse and easily monitor their execution and output. |
| M2E | 1.12 | Maven development tool that enables end to end debugging of maven plugins, maven core and their dependencies from m2e development workspace. |
| QAF | 1.0.0 | Editor for Behavior Driven Development using QMetry Automation Framework |
| JAutodoc | 1.14.2 | JAutodoc is an Eclipse Plugin for automatically adding Javadoc and file headers to your source code. |
| Ansi Console | 1.4.2 | Plugin interprets the ANSI escape sequences to color the console output. |
| Grep Console | 3.7.0 | Grep Console used to define a series of regular expressions to test against the console output. Each expression matching a line will affect the style of either the entire line or parts of it. |

## Advanced Eclipse Preference Settings

(Mandatory)

After installing Eclipse, perform the below steps to enable “Save Actions” in preference:

1. Click on Windows menu in Eclipse -> select “Preference”
2. Click on “Import”
3. In file sector, navigate to eclipse installation folder
4. Select the file \\ne1nsp01\data\code\QTP\Automation\_Tools\Eclipse\2020-06\EclipsePreferences.epf
5. Click on ‘Finish
6. Restart the eclipse instance and open same workspace.
7. Update Codeformatter in Eclipse by selecting “Preference”->Java ->Code Style -> Formatter -> Import -> Browse and select:  
   \\ne1nsp01\data\code\QTP\Automation\_Tools\Eclipse\2020-06\CodeFormatter.xml  
   and click on Apply and click the radio button “Overwrite the existing profile” if prompted -> OK  
   -> Click “Apply and close”.  
   IMPORTANT NOTE: You need to import the Code Formatter and set all Eclipse Preference Settings if you create new workspace, so we recommend using only the single workspace which was originally installed rather than using multiple workspaces.
8. Add your full name for auto-insertion with JavaDocs:  
   Update Eclipse -> select “Preference”->Java ->JAutodoc -> Templates -> click on “Properties” -> Select “Name” -> Click on “Edit”-> Enter in Value field -> OK  
   Click “Apply and Close”.

## Points to be noted

1. Java 8 or newer version is required to run Eclipse version 2019-06. Follow [Steps to Install Java](#steps-to-install-java) to download and install latest version of Java in your machine
2. For any Queries, please drop a mail to [**QE-AdvancedTestingTeam@ .com**](mailto:QE-AdvancedTestingTeam@lfg.com)

## First time Install for project

1. By default, all lines of code in the project imported into Eclipse will be formatted based on the code style configured in eclipse (Eclipse -> Preference -> Java -> Formatter). This will introduce conflicts while committing the code to GIT. It is recommended to follow below steps to avoid conflicts.

* Update your projects codebase in GIT to above coding format I.e. Any one team member of the project should check in formatted code into GIT
* All other team members should check-out the formatted code from GIT and have “CodeFormatter” in their eclipse installation before incorporating any new features.

## Importing and Executing sample script

1. For importing project, Open [https://gitlab. .com/](https://gitlab.lfg.com/)
2. Navigate to “Projects” -> “Your Projects”. From the list, click on the project to be cloned
3. Inside the project overview window, click on Clone and copy “Clone with HTTPS” URL
4. In Eclipse, Select Import from File menu
5. In Import window, select “Git” -> “Project from Git” and click on “Next”
6. Select “Clone URI” option and click on “Next”
7. In the Source Git Repository window, paste the link copied from GitLab in URI field and enter your GIT credentials in authentication field (**not LDAP credential**) and click on Next
8. In “Branch Selection” window, select the branch to be cloned
9. In “Import project from Git” window, Select the destination directory where project to be download and click on Next
10. Eclipse will start downloading the project to the selected location. Once downloading completed click on Finish
11. Right click on the project folder (e.g.:) in Project Explorer tab and select Maven -> Update Projects
12. In the ‘Update Maven Project’ window, select the project and ‘Force Update of Snapshots/Releases’ checkbox and click OK button to start downloading jar files

## Workspace Settings Implemented in Eclipse Package

(just for Reference)

1. Updated with keyboard shortcuts
2. Window -> Preferences -> Ansi Console  
   • Check Enabled only  
   • Check Standard VGA colors
3. Updated “Colors & Fonts” eclipse configuration as per “Selenium Installation” document  
   • Header Font -> Edit -> Arial – Bold – 12  
   • Text Editor Block Selection Font -> Edit -> Arial – Regular - 11  
   • Text Font -> Edit -> Arial – Regular – 11
4. Window -> Preferences -> General -> Editors -> Text Editors -> clear “Insert spaces for tabs”  
   • clear “Highlight current line” (stop blue background when copy/paste)
5. Window -> Preferences -> General -> Workspace -> check “Refresh using native hooks or polling”  
   • Text file encoding -> Other: UTF-8 (for logs)
6. Window -> Preferences -> Install/Update -> Automatic Updates  
   • Uncheck – Automatically find new updates and notify me
7. Window -> Preferences -> Java -> Code Style -> Formatter -> Import = CodeFormatter
8. Window -> Preferences -> Java -> Editor -> Content Assist  
   • Auto activation triggers for Java: (enter the below to allow any character to open intellisense)  
   .qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM,
9. Window -> Preferences -> Java -> Editor -> Typing -> uncheck Escape text when pasting into a string literal
10. Underline hyperlinks in the Console properly:  
    • Window -> Preferences -> Grep Console -> Settings  
    • Style match length=1000  
    • Filter match length=1000
11. TestNG - Enabled – Disable default listeners
12. Window -> Preferences -> Java -> Editor -> Save Actions -> check Perform the selected actions on save  
    a. Check Additional actions -> press Configure button  
    b. Tab - Code organizing -> check Remove trailing whitespace, select “All lines”  
    c. Tab – Unnecessary Code -> only check: Remove unused imports  
    d. Uncheck Remove Unnecessary casts
13. Window -> Preferences -> General -> Network Connections  
    • Change Active Provider = Manual  
    In Eclipse, Window -> Preferences -> General -> Network Connections  
    • Set Active Provider = Manual  
    • Set the HTTP & HTTPS Host = nc2bcproxy101.us.ad. .com & Port = 8080 as shown below  
    Set the Proxy bypasses