Jenkins Setup - Pull request triggering, unit test checking and automatically merge

Overview

Figure 1: High level view for Jenkins setup

Descriptions

Jenkins is an automation server, that automate the software development process, with continuous integration. Jenkins can be configured to perform checks during different stages of development to ensure new implementation do not affect the current working functionality. This documentation serves as a setup guide for developers who wish to enforce unit testing before the pull request is allowed to be merged. This documentation covers the setup for Jenkins, configuration to trigger Jenkins build upon pull request, configuration to handle MSVC projects and how to get the unit test (GMOCK, GTEST) results, configuration to automate the merge if the build and unit test succeed otherwise reject the pull request.

Useful Information

Below are name and address used by the author. All images in the documentation are set based on the table.

Application	Name/addresses
GitHub Repository	Jenkins_Test
GitHub Repository url	git@git-id.conti.de:uia94765/Jenkins_Test.git
Jenkins project	Project_testing
Jenkins url	http://igd0257g:8080/
Jenkins Project url	http://igd0257g:8080/job/Project_testing/
Author's hostname	igd0257g
Testt	project name for production code and unit testing (folder name where codes are stored)

Setup

Setup Jenkins

1. https://jenkins.io/doc/book/installing/#war-file
2. Navigate to 'WAR file" section and download

or

1. http://mirrors.jenkins.io/war-stable/latest/jenkins.war

Start Jenkins server

Create a .BAT file

> *java -jar jenkins.war --httpPort=8080*

or Download (runJen.BAT)

Save the .BAT file in the same directory with jenkins.war, to execute Jenkins server, execute the .BAT file

Run http://localhost:8080/ on web browser

Shutdown Jenkins server

Address for Jenkins Server: http://igd0257g:8080/

To shutdown Jenkins Server: http://igd0257g:8080/exit/

Plugins

Plugins can be either added manually via .hpi file or through configuring the proxy server (FYI, hpi method is used for my setup, for convenience, you choose to obtain the proxy information from Conti)

Below are the list of main plugins required

Plugins	Descriptions/Used for
Git	Enable interaction between Jenkins and Git
GitHub	Enable interaction between Jenkins and GitHub
MSBuild	Use to allow Jenkins to build MSVC project
GitHub Pull Request Builder (ghprb.hpi)	Receive trigger when pull request is activated by the developer in Git
GitHub Pull Request Merger	Automate the merging process if the status check pass

Note: Most of the above plugins have certain dependencies plugins, please install them accordingly

Most plugins required can be obtain from here: https://updates.jenkins-ci.org/download/plugins/

or

You can retrieve all the plugins from my folder: didc0006\30_S3\S3-HMI\WD_YongChuan\Jenkins\hpi.zip

GitHub

Developers Setting

Create access token

Click on your profile in Git Developer settings Personal access tokens

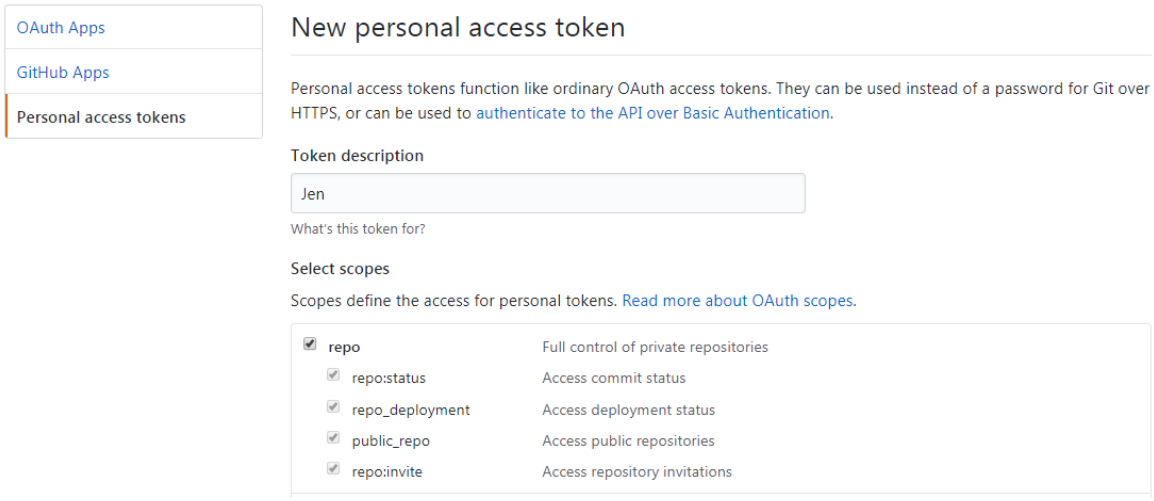


Figure 2: Creating access token in Git

Expected summary of access token

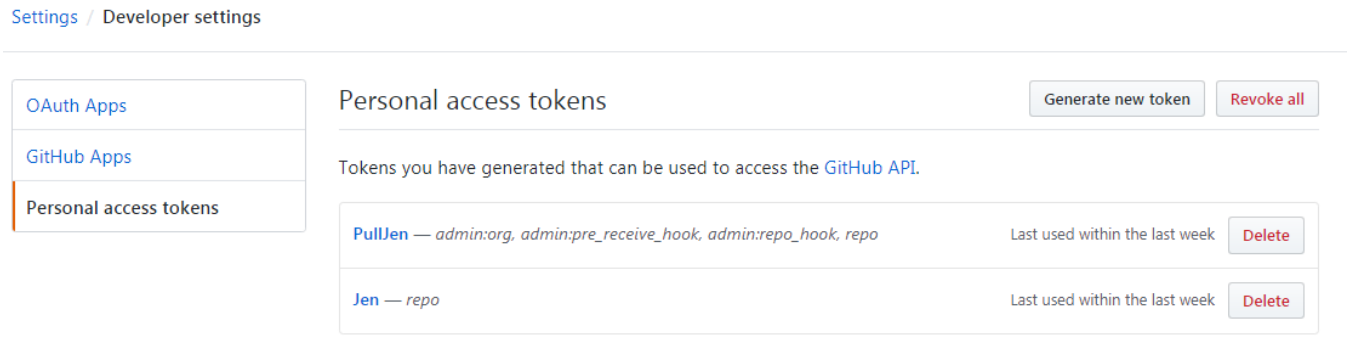


Figure 3: Tokens for Jenkins configuration

Repository Configuration

clicked on your Git repo Setting

Branches

More rules can be set based on administrator own discretion.

For my Jenkins server, I created a protection rule for the master branch, this prevent developers from merging directly into the branch without any Jenkins build.

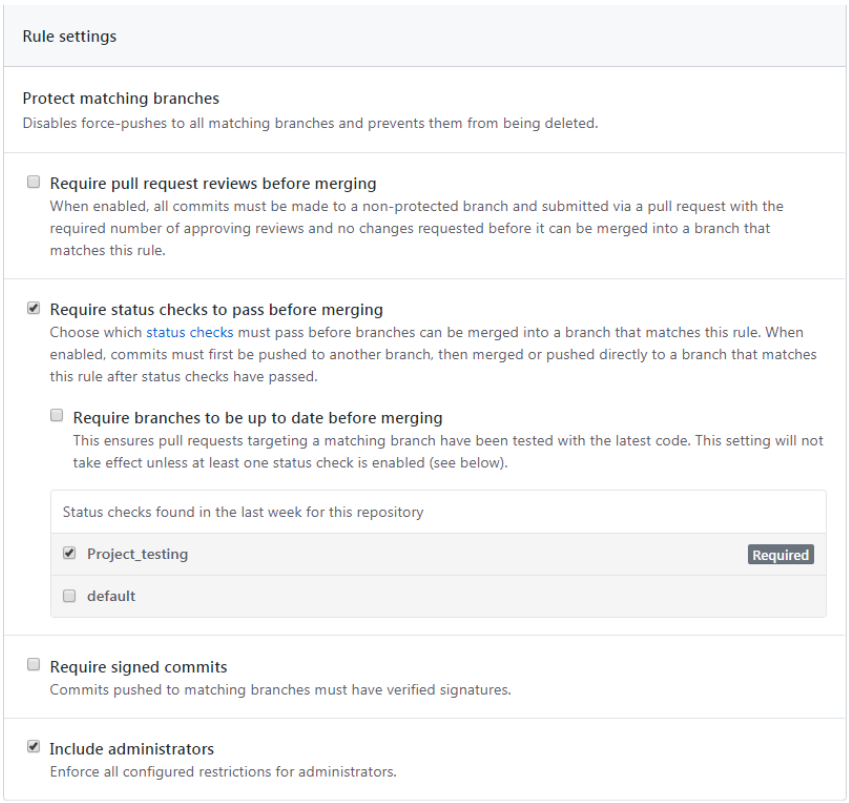


Figure 4: Git branch protection configuration

Hooks

Payload URL: {Your Jenkins address:Port}/ghprbhook/

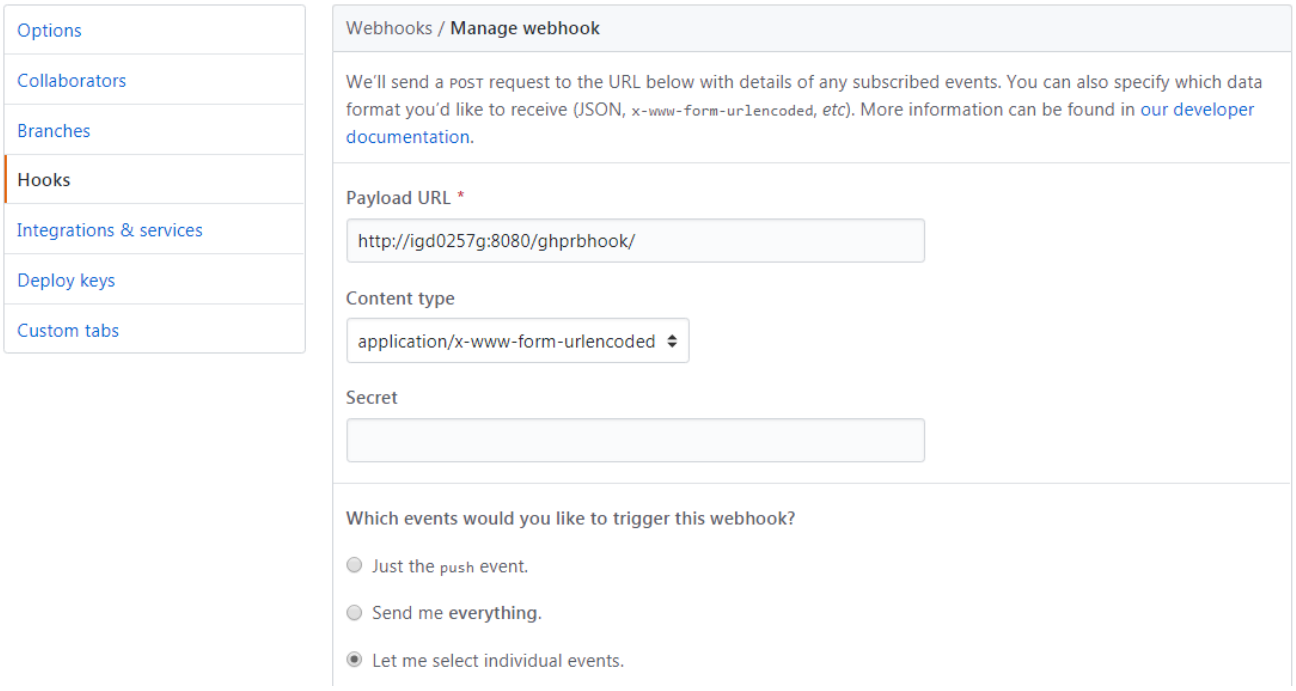


Figure 5: Configuring webhook for pull request

Select

1. Pull requests

Ensure active is checked

- You can select to receive more events at your own discretion

Integration and services

Add services for:

1. Jenkins (GitHub plugin)
2. Jenkins (Git plugin)

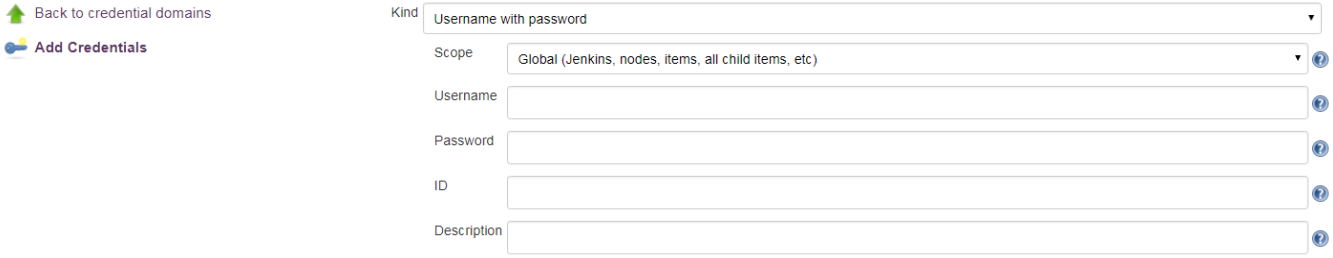
Where Jenkins hook URL: http://igd0257g:8080/github-webhook/

Jenkins

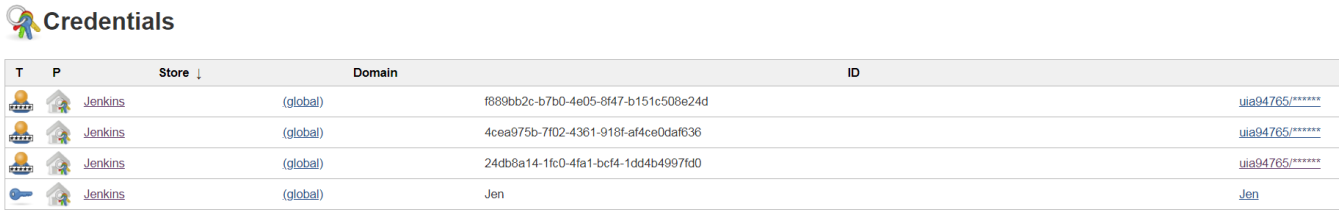
Jenkins Configuration

Configure Credentials

Navigate to add credentials:



1. Select Kind as "Secret text"
2. Scope "Global (Jenkins, nodes, items, all child items, etc)"
3. Add in the string of text from GitHub (access token)
4. Repeat the steps above
5. Select Kind as "username with password"
6. Enter your Git Repo credentials



Configure system

Clicked on Jenkins Manage Jenkins

1. Set your Jenkins URL this URL will be used to access the Jenkins server and for creating webhook in Git. {http://{hostname}:{port}}

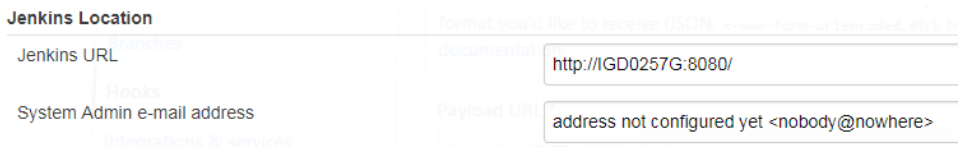


Figure 6: Setting Jenkins URL

2. Set up GitHub Server

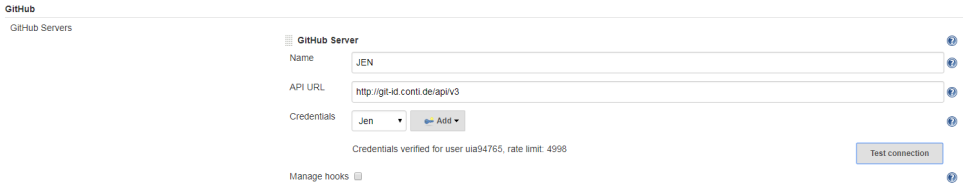


Figure 7: Configuring GitHub Server

3. Configure GitHub Pull Request Builder

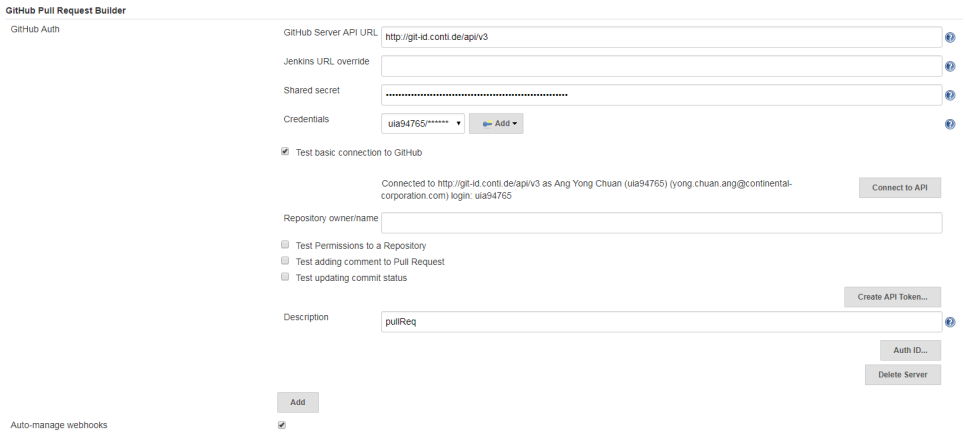


Figure 8: Configuring GitHub pull request builder

a. Change the GitHub Server API URL to Contiental API address
b. Select the credentials previously created
c. Test to ensure connectivity to GitHub

Global Tool Configuration

Under MSBuild

1. Add a random name
2. Add in the path of MSBuild.exe in your local directory

Note: path of MSBuild.exe (example C:\Program Files (x86)\Microsoft Visual Studio\2017\WDExpress\MSBuild\15.0\Bin\MSBuild.exe)

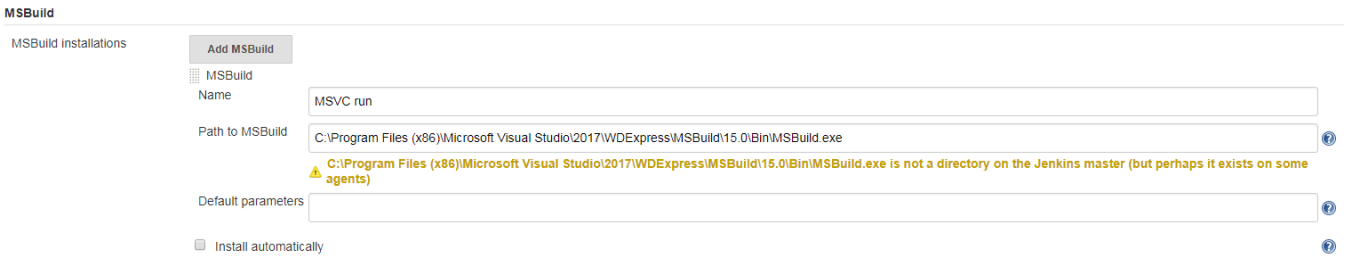


Figure 9: Setting the directory for MSBuild

Project configuration

Clicked on Jenkins Project Configure

General

1. Checked "GitHub project"
    a. Add in the project URL: http://git-id.conti.de/{hostname}/{GitHub project name}/
2. Checked "Execute concurrent builds if necessary"

Note: Go to Git and copy the address of the repository

Figure 10: Project configuration, general tab

Source Code Management

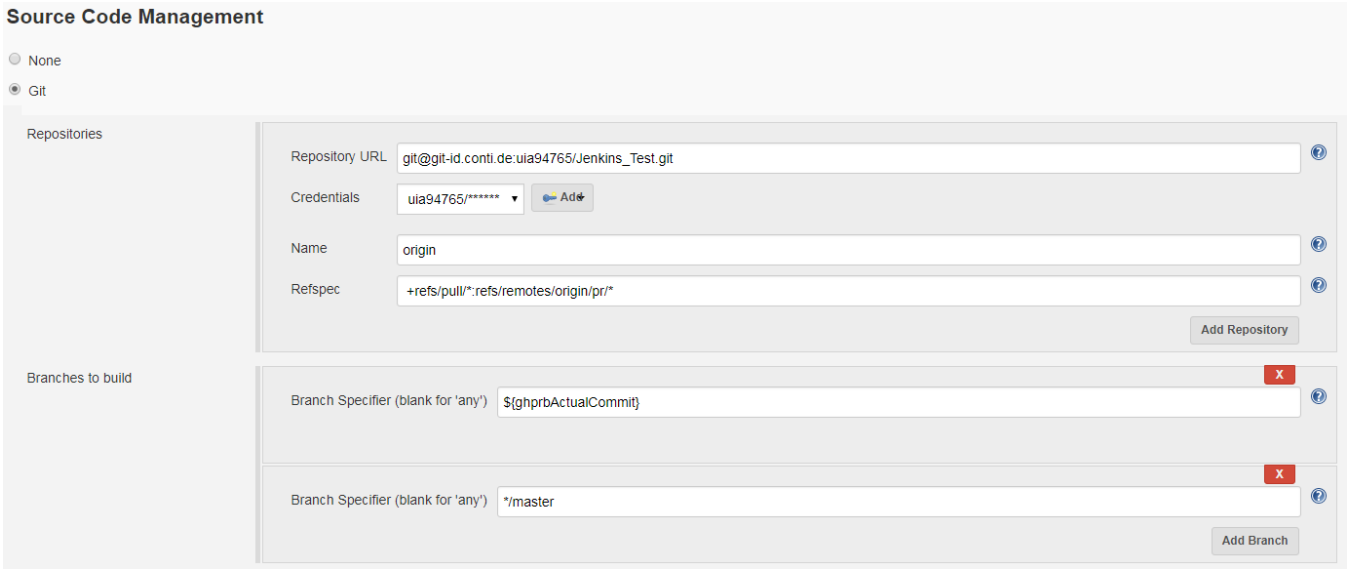


Figure 11: Settings for source code management

More information can be found here: https://wiki.jenkins.io/display/JENKINS/GitHub+pull+request+builder+plugin

Build Trigger

Select GitHub Pull Request Builder

1. Select the configuration defined previously (under Configuration System)

Figure 12: Setting to trigger Jenkins when pull request event is received

2. Click on advance
3. Select the build pull request automatically option

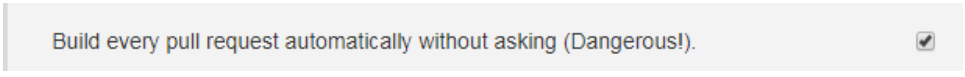


Figure 13: Ensure pull request are triggered

Build

1. Select "Build a Visual Studio project or solution using MSBuild" under "Add build step"

2. Select the projects that you want to build (for my current case, gmock project, gtest project, production code and unit test project)

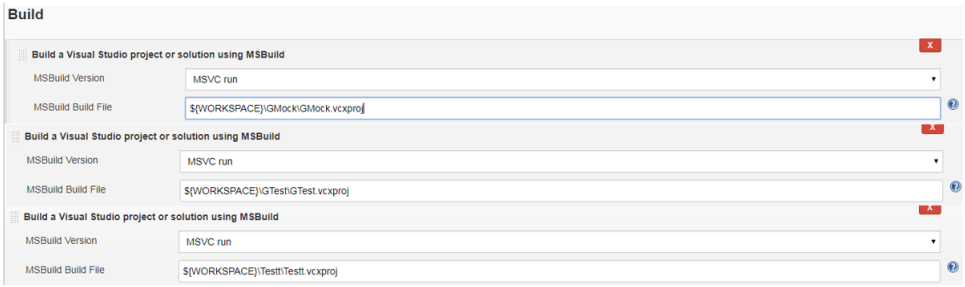


Figure 14: Setting the project to build

Inputs	Descriptions
MSVC run	Setting configured under Global Tool Configuration
${WORKSPACE}\Testt\Testt.vcxproj	${WORKSPACE} macros for Jenkins work space Testt project folder name Testt.vcxproj project file name

Refer to Workspace section for more information about the folder and directory

3. Add a new build step "Execute Windows batch command"

Figure 15: Retrieving the unit test result

The caller.BAT file is used to navigate to the location where the unit test result is stored. For google test, the unit test result can be found in the {project folder}/Debug/{project name}.exe

Below is the batch script command in caller.BAT

```
cd Testt/Debug
Testt.exe
```

Command	Descriptions
cd Testt/Debug	Navigate to the directory where the result from unit testing is stored
Testt.exe	Result of unit testing (Testt project name)

Post-build Actions

The post-build actions allow the pull request to automatically merged if the build succeeded and unit test cases are passed

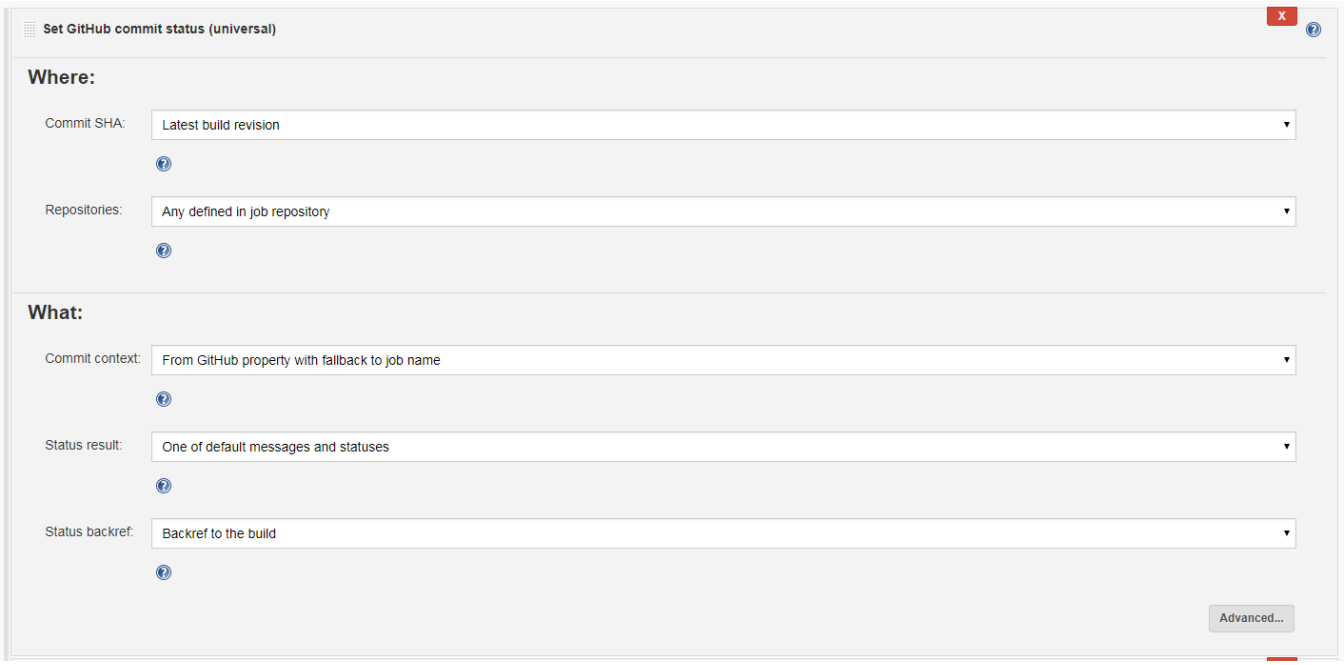


Figure 16: Setting the commit status

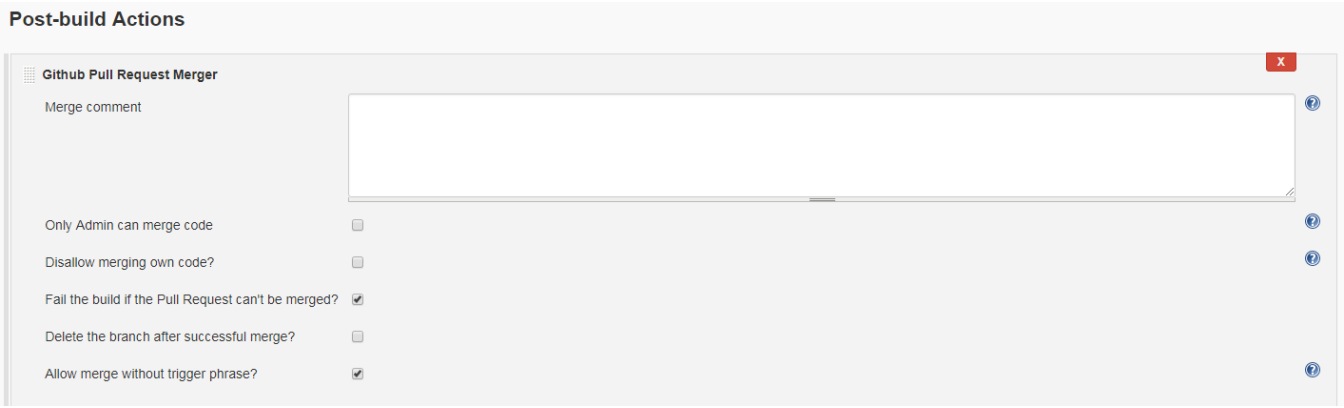


Figure 17: Trigger auto merge if build and unit test is success

Workspace

Current work space in Jenkins

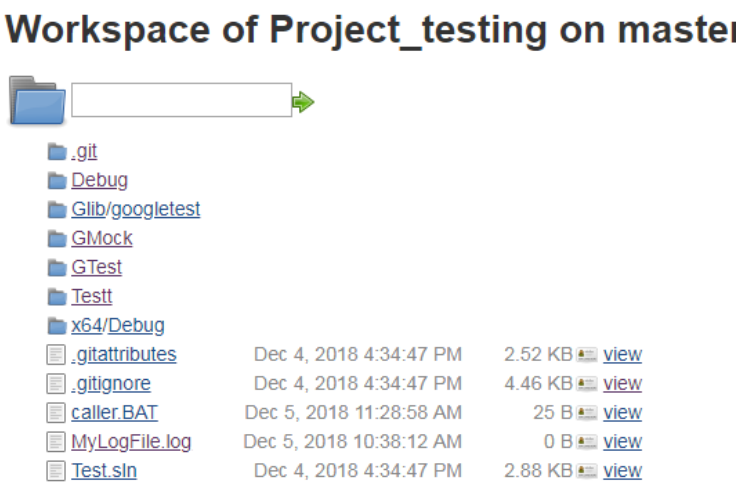


Figure 18: Workspace in Jenkins

Results and Process

Triggering Jenkins

When developers clicked the pull request button, a "gbprbhook" will be triggered and sent from Git to Jenkins as shown in figure 19

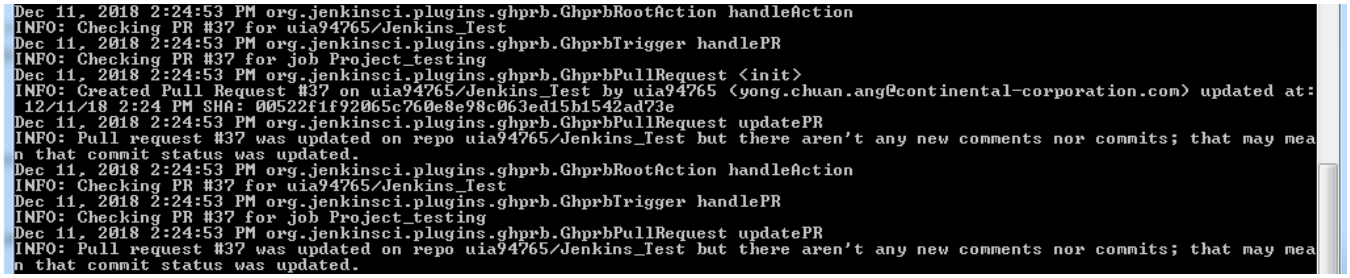


Figure 19: Trigger received by Jenkins

In the meantime the status in Git will set to pending as shown in figure 20. Observe that the "Merge pull request" button is disabled before the status check is completed.

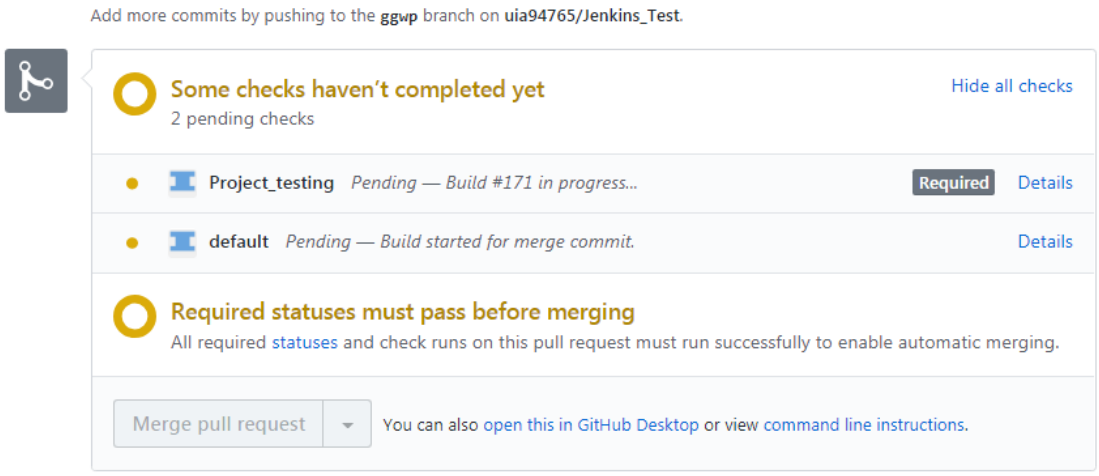


Figure 20: Pending status

When the pull request is selected, Git will send a trigger event to Jenkins server via the "ghprbhook" defined in the Git webhook section. As shown in figure 21

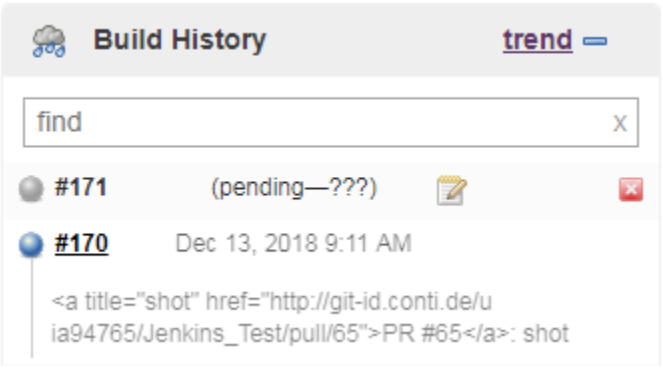


Figure 21: Jenkins triggered when ghprhook is received

Reject pull request

Error building the project

```
c:\users\uia94765\.jenkins\workspace\project_testing\testt\testt.cpp(6): error C2059: syntax error: ')' [C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj]
c:\users\uia94765\.jenkins\workspace\project_testing\testt\testt.cpp(6): warning C4305: 'initializing': truncation from 'bool (__cdecl *)(void)' to 'bool'
[C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj]
Done Building Project "C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj" (default targets) -- FAILED.

Build FAILED.

"C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj" (default target) (1) ->
(ClCompile target) ->
  c:\users\uia94765\.jenkins\workspace\project_testing\testt\testt.cpp(6): warning C4305: 'initializing': truncation from 'bool (__cdecl *)(void)' to 'bool'
[C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj]


"C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj" (default target) (1) ->
(ClCompile target) ->
  c:\users\uia94765\.jenkins\workspace\project_testing\testt\testt.cpp(6): error C2059: syntax error: ')' [C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Testt.vcxproj]

    1 Warning(s)
    1 Error(s)
```

Figure 22 shows a situation when the building of the project fails, error is shown in the console output in Jenkins (syntax error)

Fail to pass all test case

Figure 23 shows one of the test case has failed (Failing test case)

```
C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Debug>Testt.exe
Running main() from gmock_main.cc
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from bBeingHonestFunction
[ RUN      ] bBeingHonestFunction.AlwaysReturnsTrue
c:\users\uia94765\.jenkins\workspace\project_testing\testt\testt.cpp(7): error: Value of: actual
Expected: is equal to false
  Actual: true (of type bool)
[  FAILED  ] bBeingHonestFunction.AlwaysReturnsTrue (1 ms)
[----------] 1 test from bBeingHonestFunction (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (2 ms total)
[  PASSED  ] 0 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] bBeingHonestFunction.AlwaysReturnsTrue

 1 FAILED TEST
Build step 'Execute Windows batch command' marked build as failure
Build did not succeed, merge will not be run
[Set GitHub commit status (universal)] ERROR on repos [GHRepository@2f61d7c5[description=<null>,h
{},language=C++,commits={},source=<null>,parent=<null>,responseHeaderFields={null=[HTTP/1.1 200 O
GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, X-OAuth-Scopes, X-Accept
Length=[5450], Content-Security-Policy=[default-src 'none'], Content-Type=[application/json; char
Modified=[Mon, 10 Dec 2018 08:45:28 GMT], OkHttp-Received-Millis=[1544509850574], OkHttp-Response
[1544509850036], Referrer-Policy=[origin-when-cross-origin, strict-origin-when-cross-origin], Ser
OAuth-Scopes=[repo], X-Content-Type-Options=[nosniff], X-Frame-Options=[deny], X-GitHub-Enterpris
29f8-4e57-83ce-3590f172b971], X-OAuth-Scopes=[repo], X-RateLimit-Limit=[5000], X-RateLimit-Remain
Runtime-rack=[0.027283], X-XSS-Protection=[1; mode=block]},url=http://git-id.conti.de/api/v3/repo
Setting commit status on GitHub for http://git-id.conti.de/uia94765/Jenkins_Test/commit/ca73d91e9
Setting status of ca73d91e945d0357c180a5f3a5c889d9f58d7b1b to FAILURE with url http://IGD0257G:80
Finished: FAILURE
```

Figure 23: Console log in Jenkins (failed unit test)

For both of the case, the final status of the build is "Failed"

Git Status (Failure)

Figure 24 shows the result in Git for both situation stated above where either the building or the unit test fails

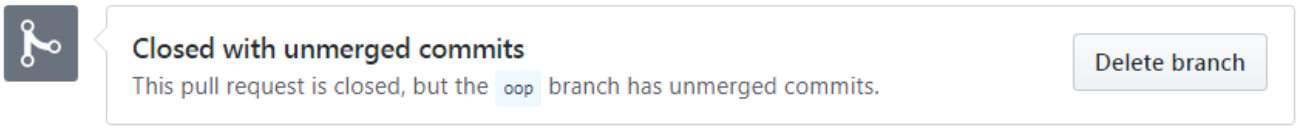


Figure 24: Git status for fail build or unit test

Accept pull request

Build and unit test passed

Figure 25 shows the result of the console logs when both the building and unit test are successful

```
C:\Users\uia94765\.jenkins\workspace\Project_testing\Testt\Debug>Testt.exe
Running main() from gmock_main.cc
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from bBeingHonestFunction
[ RUN      ] bBeingHonestFunction.AlwaysReturnsTrue
[       OK ] bBeingHonestFunction.AlwaysReturnsTrue (1 ms)
[----------] 1 test from bBeingHonestFunction (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (1 ms total)
[  PASSED  ] 1 test.
Merging the pull request
Merging
PR[GHPullRequest@17b71e1[base=org.kohsuke.github.GHCommitPointer@2124c2ae,h
<null>,assignees={},state=open,number=39,comments=0,labels=[],title=Demobra
Allow-Origin=[*], Last-Modified=[Tue, 11 Dec 2018 06:33:54 GMT], X-Runtime-
Access-Control-Expose-Headers=[ETag, Link, Retry-After, X-GitHub-OTP, X-Rat
RateLimit-Remaining=[4970], X-RBT-Optimized-By=[siwo005 (RiOS 9.1.5a) SC],
[2.15.3], X-XSS-Protection=[1; mode=block], Content-Length=[16297], X-GitHu
Content-Type-Options=[nosniff], X-RateLimit-Reset=[1544513094], Date=[Tue,
Vary=[Accept, Authorization, Cookie, X-GitHub-OTP], X-RateLimit-Limit=[5000
Pull request successfully merged
[Set GitHub commit status (universal)] SUCCESS on repos [GHRepository@4f930
{},language=C++,commits={},source=<null>,parent=<null>,responseHeaderFields
GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset, X-
Length=[5450], Content-Security-Policy=[default-src 'none'], Content-Type=[
Modified=[Mon, 10 Dec 2018 08:45:28 GMT], OkHttp-Received-Millis=[154451007
[1544510073283], Referrer-Policy=[origin-when-cross-origin, strict-origin-w
OAuth-Scopes=[repo], X-Content-Type-Options=[nosniff], X-Frame-Options=[den
fc5b-426b-b55c-d69c1dad0a18], X-OAuth-Scopes=[repo], X-RateLimit-Limit=[500
Runtime-rack=[0.036182], X-XSS-Protection=[1; mode=block]},url=http://git-i
Setting commit status on GitHub for http://git-id.conti.de/uia94765/Jenkins
Setting status of 2f6a5381f3ecb534562660963dc84b62c64be1e1 to SUCCESS with
Finished: SUCCESS
```

Figure 25: Console log in Jenkins (success build and unit test)

Git Status (Success)

Figure 26 shows the result in Git, when both the build and unit test is success, the pull request will be automatically merged and the pull request will be closed.

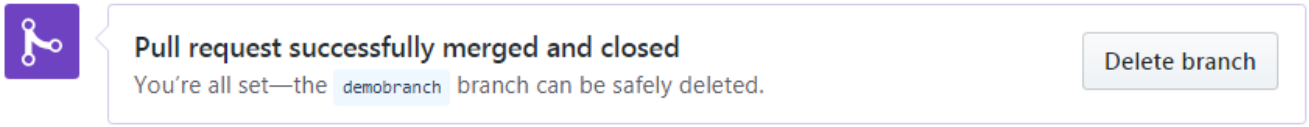


Figure 26: Git automatically merge pull request if build and unit test passed

Pull request from master branch

Under normal situation, developers will create a branch and commit their changes and implementation into the branch and perform a push to repository, in the repository the developers will then submit the pull request.

This section is to show the situation when a developers tries to commit and push on the master branch directly. Since a branch protection is set where it would required a status check to be performed before the push can be proceed. The trigger for the status check is based on the pull request, therefore, when the developers tries to push directly from the master branch, the following error in figure 27 will be shown.

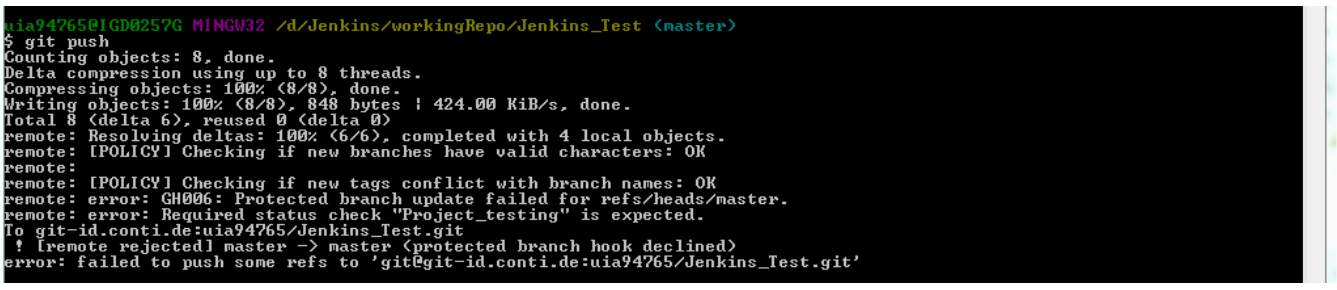


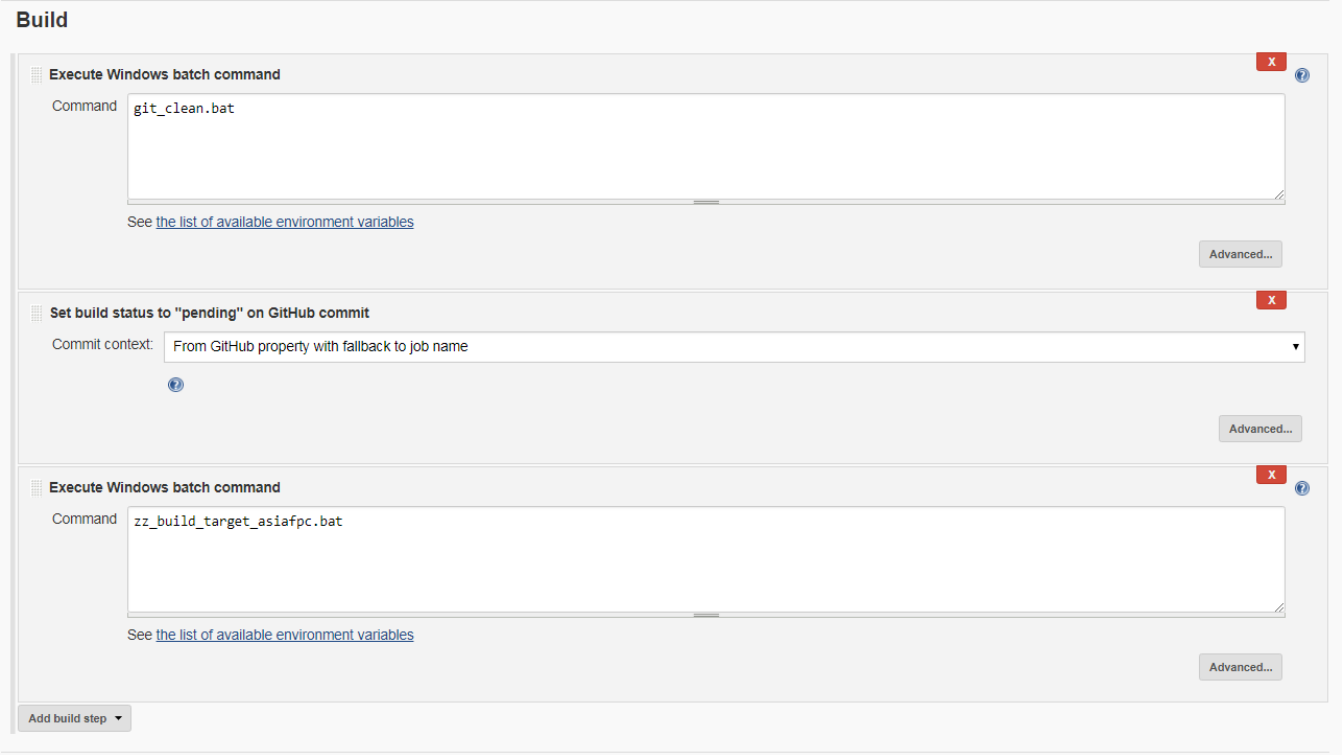
Figure 27: Error message when pushing directly from master

Working with FPK-advance repository

Previously the building stage of the project involve building only a simple MSVC project. The current section discuss on how developers can perform a build using "zz_build_target_asiafpc.bat" file in the "warning_subsystem_dev branch". The main changes to achieve this is

1. Modify the build process in Jenkins server
    a. Execute a git clean using "git_clean.bat" (optional but preferred)
    b. Execute the script to build target using "zz_build_target_asiafpc.bat"
2. Ensure Jenkins work space contain working environment of "warning_subsystem_dev"

Modification to existing configuration



Assuming unit testing files are added to the "warning_subsystem_dev" branch, the build process should follow this flow:

1. git_clean.bat
2. "zz" build file for simulation
3. git_clean.bat
4. "zz_build_target_asiafpc.bat"
5. bat script that navigate to unit test folder and execute the ".exe"

Process and result in Jenkins

Git clean

Previous Build

```
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 2fc10137e191bf5269f9198ba6916027ad40958e
Commit message: "p"
 > git.exe rev-list --no-walk 71d2267697ba8d66ea88be37713f8d71cfd44546 # timeout=10
[Project_testing] $ cmd /c call C:\Users\uia94765\AppData\Local\Temp\jenkins6357358019029245222.bat

C:\Users\uia94765\.jenkins\workspace\Project_testing>git_clean.bat
Purge repository by resetting all files to repository state
  git clean -xfd
-----------------------------------------------------
Removing NEC_V850_proj.sln
Removing NEC_V850_proj.vcxproj
Removing NEC_V850_proj.vcxproj.filters
Removing _install.log
Removing adapt/gen/
Removing ide/tmp/
Removing logfile
-----------------------------------------------------
Press any key to continue . . .
```

Execute the build

Jenkins › Project_testing › #212

```
C:\Users\uia94765\.jenkins\workspace\Project_testing>zz_build_target_asiafpc.bat
Building C:\Users\uia94765\.jenkins\workspace\Project_testing as tgt
Starting build IDE installation . . .
================================================
calling ide\adapt\extras\install_hook1.bat JCP2016_RefSys_HL3D_AC -c -p -c -p
ide\adapt\paco\paco.cmd: paco install directory D:\Tool\common\paco\v1.3.8 exists, assuming paco is installed.
ide\adapt\paco\paco.cmd: selected paco location profile "SingaporeDev"
ide\adapt\paco\paco.cmd: running D:\Tool\common\paco\v1.3.8\paco ide\adapt\paco\ProjectDependencyFile.patd C:\ProgramData\setenv_host.bat SingaporeDev
->
Paco, Version 1.3.8.0
[OK]
checking perl .............................[OK]
checking perl compatibility ...............[OK]
preparing setenv host file ................[OK]
preparing setenv file .....................[OK]
checking unixtools ........................[OK]
checking mount mode .......................[OK]
creating make link ........................[OK]
copy missing VS project template ..........[OK]
calling ide\adapt\extras\install_hook2.bat JCP2016_RefSys_HL3D_AC -c -p -c -p

================================================
running Arbor System configuration.........[OK]
 ...Generating feature tree diagram  ..........
 ...and diagram with actions for ON features...
 .........in {PROJ_LOC}\adapt\gen\arbor .......
  runtime 00:00:05.28 for tool #0 arbor
------------------------------------------------

hwhpatch processing 1414
-- hwhpatch done

[OK]
preparing IDE environment .................[OK]
preparing language environment ............[OK]
running tool report generation ............[OK]
running tool version check ................[OK]
creating VS link ..........................[OK]
Build IDE version: 5.0.41-devel
Project name:      NEC_V850
------------------------------------------------
There were warnings:
 created tool report SYSBUILD file;
Check _install.log for more details.
```

Fail Case

```
Paco, Version 1.3.8.0
[OK]
checking perl .............................[OK]
checking perl compatibility ...............[OK]
preparing setenv host file ................[OK]
preparing setenv file .....................[OK]
checking unixtools ........................[OK]
checking mount mode .......................[OK]
creating make link ........................[OK]
copy missing VS project template ..........[OK]
calling ide\adapt\extras\install_hook2.bat -c -p -c -p

================================================
running Arbor System configuration.........[Failed]
Error: The -c.rsh file doesn't exists !!
================================================

[FAILED]
Build IDE version: 5.0.41-devel
Project name:      NEC_V850

_____________________________________________
There were errors:
 install_hook2 failed;
0 warnings or errors found.
Check _install.log for more details.
Build IDE will not work!


========================================
Error: BAD_INSTALL_MASTER from install_link.bat
The system cannot find the path specified.
Press any key to continue . . .
Build step 'Execute Windows batch command' marked build a
[Set GitHub commit status (universal)] ERROR on repos [GH
Access-Control-Allow-Origin=[*], Access-Control-Expose-He
Content-Security-Policy=[default-src 'none'], Content-Typ
200], OkHttp-Selected-Protocol=[http/1.1], OkHttp-Sent-Mi
Options=[nosniff], X-Frame-Options=[deny], X-GitHub-Enter
[1545364035], X-RBT-Optimized-By=[siwo005 (RiOS 9.1.5a) S
Setting commit status on GitHub for http://git-id.conti.c
Build did not succeed, merge will not be run
Setting status of 71d2267697ba8d66ea88be37713f8d71cfd4454
Finished: FAILURE
```

Success Case

```
CC pkg/midware/pkg/ace/core/ace/Recursive_Thread_Mutex.cpp
CC pkg/midware/pkg/ace/core/ace/Semaphore.cpp
CC pkg/midware/pkg/ace/core/ace/Task.cpp
CC pkg/midware/pkg/ace/core/ace/Thread.cpp
CC pkg/midware/pkg/ace/core/ace/Thread_Adapter.cpp
CC pkg/midware/pkg/ace/core/ace/Thread_Manager.cpp
CC pkg/midware/pkg/cpcont/core/cpcont_base.cpp
CC pkg/midware/pkg/ace/core/ace/Thread_Mutex.cpp
CC pkg/midware/pkg/ace/core/ace/Time_Policy.cpp
CC pkg/midware/pkg/ace/core/ace/Time_Value.cpp
CC pkg/midware/pkg/ace/core/ace_aros.cpp
CC pkg/midware/pkg/ace/core/ace/Unbounded_Set_Ex.cpp
CC pkg/midware/pkg/exea/adapt/exea_trace.cpp
CC pkg/midware/pkg/crhdl/adapt/crhdl_trace.cpp
CC pkg/midware/pkg/crhdl/core/crhdl_checkaliveclass.cpp
CC pkg/midware/pkg/crhdl/core/crhdl_class.cpp
CC pkg/midware/pkg/ipc/adapt/ipc_trace.cpp
CC pkg/midware/pkg/crhdl/core/crhdl_serviceclass.cpp
CC pkg/midware/pkg/dpool/adapt/dpool2_trace.cpp
CC pkg/midware/pkg/ipc/core/ipc_cinterface.cpp
CC pkg/midware/pkg/evhd/adapt/evhd2_trace.cpp
CC pkg/midware/pkg/ipc/core/ipc_signals.cpp
CC pkg/midware/pkg/ipc/adapt/ipc_tools.cpp
CC pkg/midware/pkg/ipc/core/ipc_signalspod.cpp
CC pkg/midware/pkg/ipc/core/ipc_connecthdl.cpp
CC pkg/midware/pkg/ipc/core/ipc_dispatcher.cpp
CC pkg/midware/pkg/ipc/core/ipc_signalsrpc.cpp
CC pkg/midware/pkg/ipc/core/ipc_siginterp.cpp
CC pkg/midware/pkg/ipc/core/ipc_signalsbase.cpp
CC pkg/midware/pkg/ipc/core/ipc_signalstimer.cpp
CC pkg/midware/pkg/ipc/core/ipc_signalsrcvsel.cpp
CC pkg/midware/pkg/ipc/core/ipc_timerlist.cpp
CC pkg/midware/pkg/ipc/core/ipc_timer.cpp
CC pkg/midware/pkg/ipc/core/ipc_signalsreg.cpp
CC pkg/trace/core/trace_command.cpp
CC pkg/trace/core/trace_cache.cpp
CC pkg/trace/core/trace_interface.cpp
CC pkg/trace/core/trace_interfacebaseclass.cpp
CC pkg/trace/core/trace_socket_base.cpp
CC pkg/trace/core/trace_sink.cpp
CC pkg/trace/core/trace_sinkref.cpp
CC pkg/trace/core/trace_main.cpp
CC pkg/trace/core/trace_socket_windows.cpp
CC pkg/trace/core/trace_tracecantp.cpp
CC pkg/trace/core/trace_traceisotp.cpp
CC pkg/trace/core/trace_tracetimebase.cpp
LN ide/out/hex/NEC_V850_proj.abs
Finished building project NEC_V850_proj (00:02:03.537)
Finished CRC patch operation (post_build)
Make the Debugger DNM files to have date modified.(post_build)
Finished Debugger DNM file date update.(post_build)
Finished build_workspace (00:02:05.138)
Press any key to continue . . .
[Set GitHub commit status (universal)] SUCCESS on repos [GHRepos
Access-Control-Allow-Origin=[*], Access-Control-Expose-Headers=[
Content-Security-Policy=[default-src 'none'], Content-Type=[appl
[CONDITIONAL_CACHE 304], OkHttp-Selected-Protocol=[http/1.1], Ok
Scopes=[repo], X-Content-Type-Options=[nosniff], X-Frame-Options
X-RateLimit-Reset=[1545364035], X-RBT-Optimized-By=[siwo005 (RiO
Setting commit status on GitHub for http://git-id.conti.de/uia94
```

```
Merging the pull request
Merging PR[GHPullRequest@2037c729[base=org.kohsuke.github.GHComm
<null>,locked=false,responseHeaderFields={null=[HTTP/1.1 200 OK]
Access-Control-Expose-Headers=[ETag, Link, Retry-After, X-GitHub
[unknown, github.v3], Content-Security-Policy=[default-src 'none
Content-Type-Options=[nosniff], X-RateLimit-Reset=[1545364035],
id.conti.de/api/v3/repos/uia94765/fpkAsiaTest/pulls/25,id=64878]
Pull request successfully merged
Setting status of 2fc10137e191bf5269f9198ba6916027ad40958e to SU
Finished: SUCCESS
```

Overview and suggestion for FPK-advance Jenkins

1. Under build stage
    a. Perform git clean
    b. Run .BAT script to build simulation
    c. Perform git clean
    d. Run .BAT script to build target
    e. Build unit test project
    f. Retrieve unit test result

Note: Current Jenkins setup if any of the point above fails, the whole Jenkins build will be deem as failed and the pull request will be reject.

Useful knowledge

Useful Tips	Steps
Directory of Jenkins workspace stored in local	C:\Users\uia94765\.jenkins\workspace\Project_testing
How to remove plugins	C:\Users\uia94765\.jenkins\plugins 1. Delete .hpi 2. Delete folder
Manually installing plugins, plugins installed fail due to dependencies	Failed plugins will be automatically installed once u restart the Jenkins server. Continue installing the dependencies, once done perform a Jenkins restart, you should see the plugins successfully installed (Look at the notification tab)