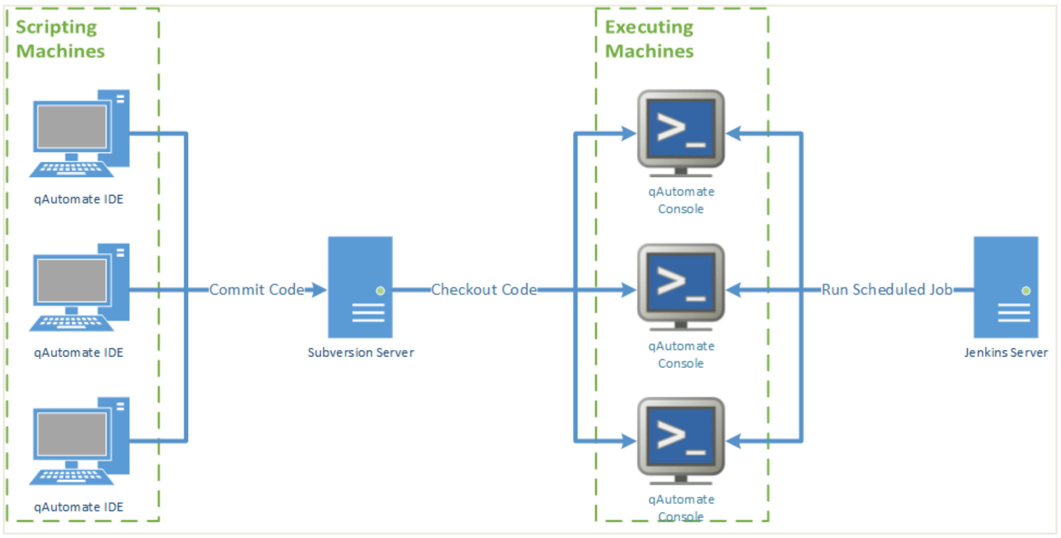
Bright Minds, Brilliant Solutions

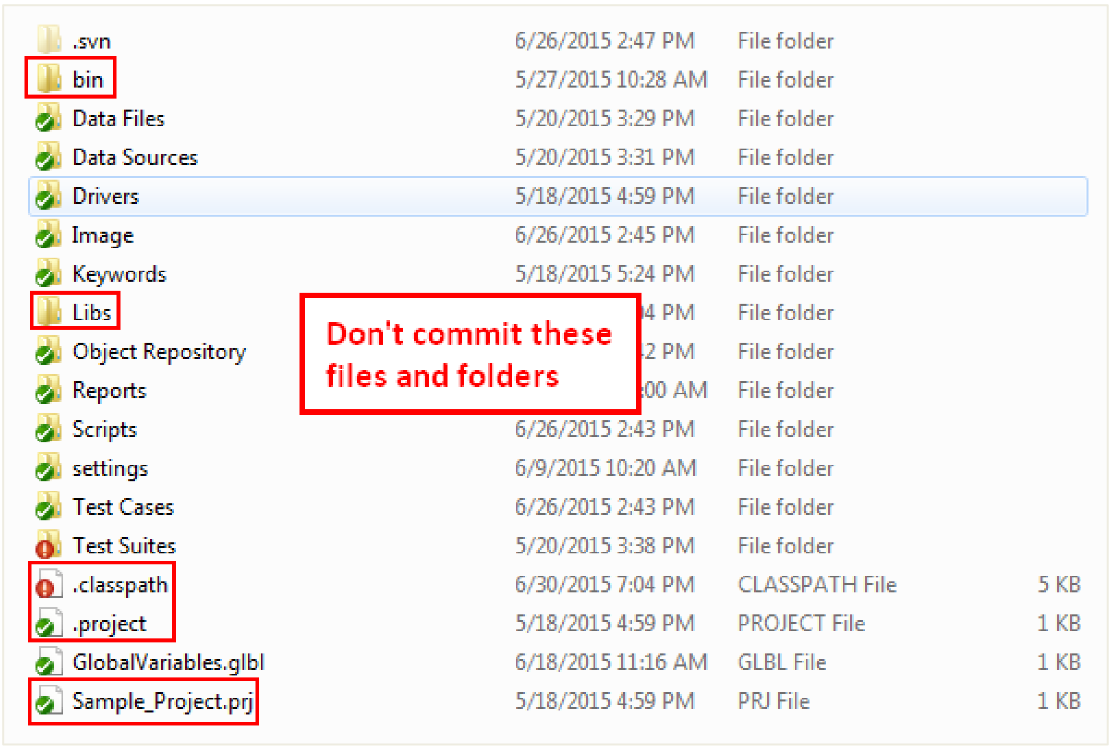
Guide to Run Katalon in Console Mode via Jenkins

Assume that we have two machines: machine A for scripting and machine B for executing. We will use machine B to execute test suite via Jenkins job.



1. **Prepare Katalon and test project**

- On machine A, open Katalon and create a Katalon test project, then import it to SVN repository. SVN folder and file .prj should have the same name. Later, when we modify test project, we should NOT commit those folders and files to SVN: **bin, Libs, Reports, .classpath, .project, .properties, {project name}.prj**

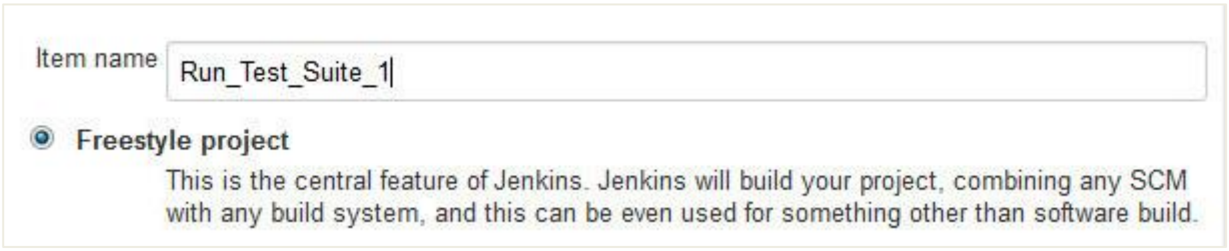


.svn	6/26/2015 2:47 PM	File folder	
bin	5/27/2015 10:28 AM	File folder	
Data Files	5/20/2015 3:29 PM	File folder	
Data Sources	5/20/2015 3:31 PM	File folder	
Drivers	5/18/2015 4:59 PM	File folder	
Image	6/26/2015 2:45 PM	File folder	
Keywords	5/18/2015 5:24 PM	File folder	
Libs	4 PM	File folder	
Object Repository	2 PM	File folder	
Reports	00 AM	File folder	
Scripts	6/26/2015 2:43 PM	File folder	
settings	6/9/2015 10:20 AM	File folder	
Test Cases	6/26/2015 2:43 PM	File folder	
Test Suites	5/20/2015 3:38 PM	File folder	
.classpath	6/30/2015 7:04 PM	CLASSPATH File	5 KB
.project	5/18/2015 4:59 PM	PROJECT File	1 KB
GlobalVariables.glbl	6/18/2015 11:16 AM	GLBL File	1 KB
Sample_Project.prj	5/18/2015 4:59 PM	PRJ File	1 KB

- On machine B, prepare location to store Katalon build and test project, install Jenkins.

2. Create Jenkins job

Login Jenkins > **New Item** > select **Freestyle project.**

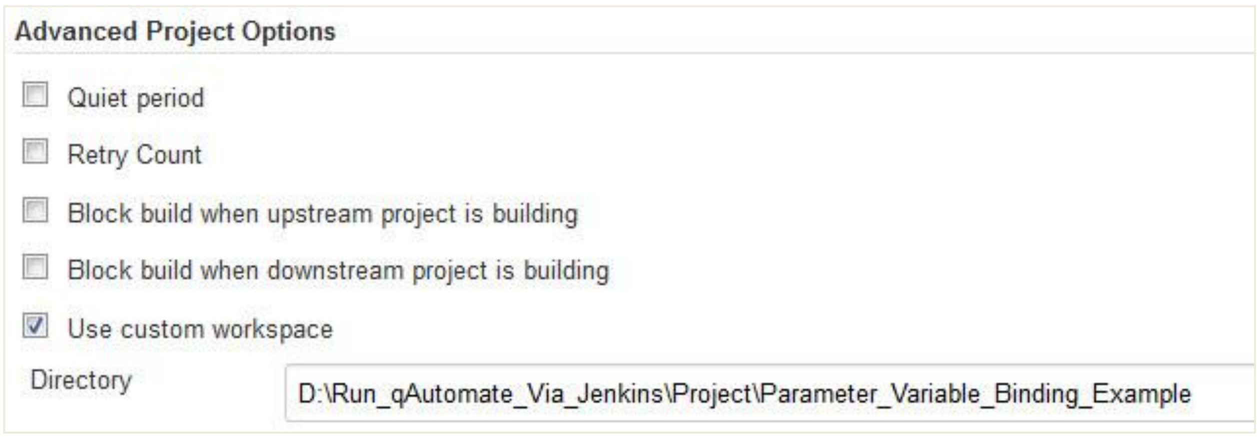


Item name: Run_Test_Suite_1

● **Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

At **Advanced Project Options**, check **Use custom workspace** and specify Directory where we want to check out the project (the location is on machine B – where we'll execute the test suite).

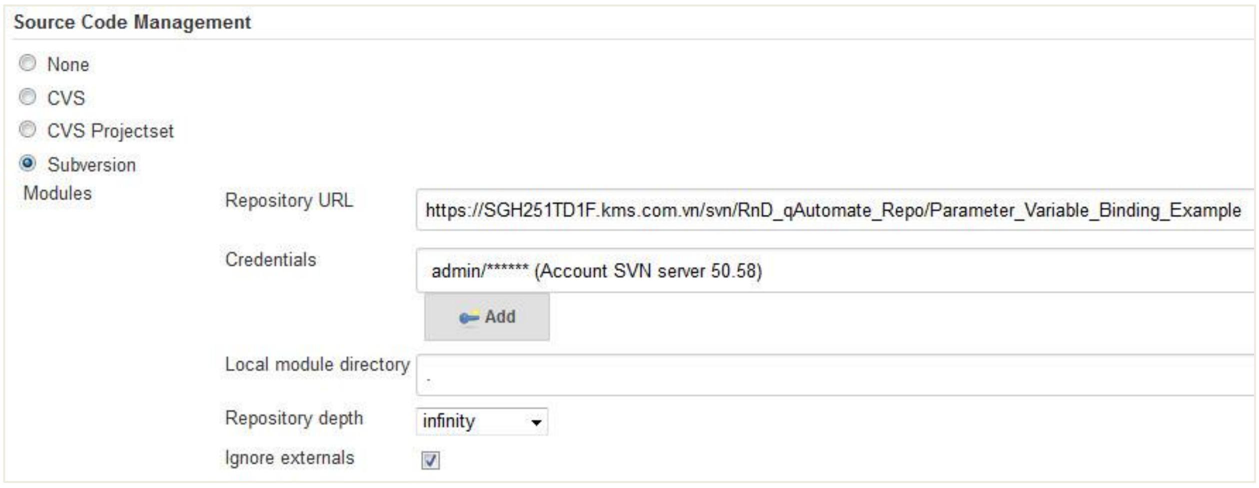


Advanced Project Options

- [ ] Quiet period
- [ ] Retry Count
- [ ] Block build when upstream project is building
- [ ] Block build when downstream project is building
- [✓] Use custom workspace

Directory: D:\Run_qAutomate_Via_Jenkins\Project\Parameter_Variable_Binding_Example

At **Source Code Management** section, select **Subversion** and fill in required information.



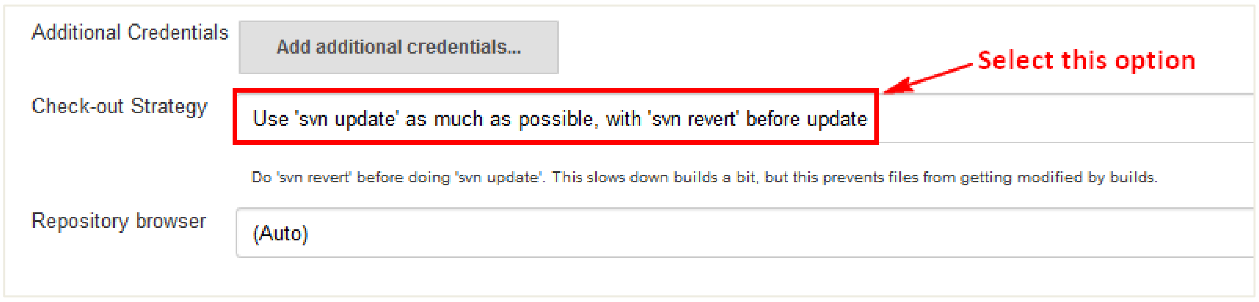
Source Code Management

○ None
○ CVS
○ CVS Projectset
● Subversion

Modules

Repository URL	https://SGH251TD1F.kms.com.vn/svn/RnD_qAutomate_Repo/Parameter_Variable_Binding_Example
Credentials	admin/****** (Account SVN server 50.58)
Local module directory	.
Repository depth	infinity
Ignore externals	✓

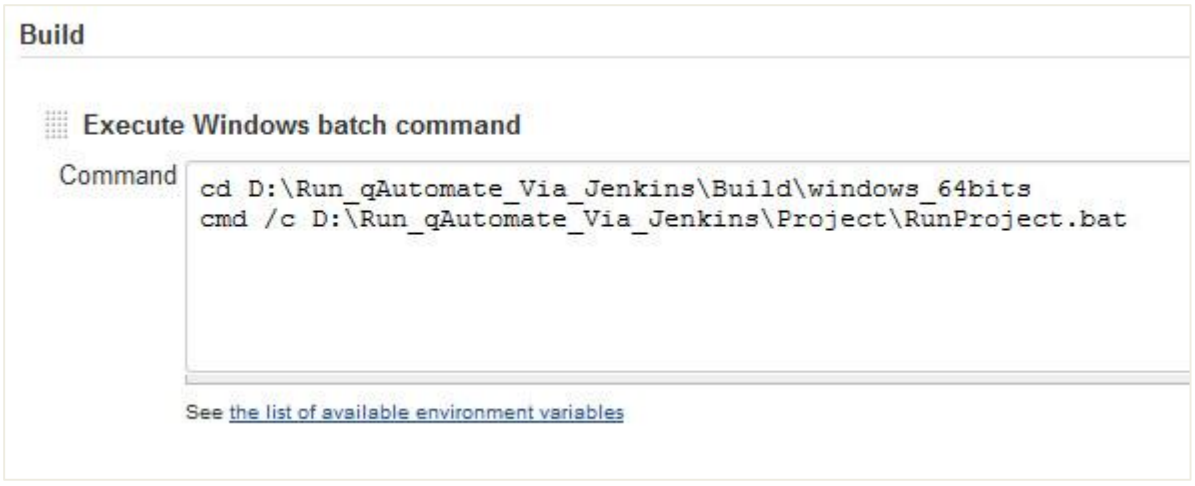
At **Check-out Strategy**, select **Use 'svn update' as much as possible, with 'svn revert' before update**.



Additional Credentials	Add additional credentials...
Check-out Strategy	Use 'svn update' as much as possible, with 'svn revert' before update
Repository browser	(Auto)

Do 'svn revert' before doing 'svn update'. This slows down builds a bit, but this prevents files from getting modified by builds.

At **Build** section, select **Execute Windows batch command** and enter command to run the test suite in console mode.


Build

Execute Windows batch command

Command

```
cd D:\Run_qAutomate_Via_Jenkins\Build\windows_64bits
cmd /c D:\Run_qAutomate_Via_Jenkins\Project\RunProject.bat
```

See the list of available environment variables

For the above example:

First command is to navigate to folder containing Katalon executable file.

Second command is to execute a batch file. This batch file contains command to execute test suite in console mode.

An example for the batch file:

```
katalon -runMode=console -summaryReport -statusDelay=30 -
projectPk="C:\RegressionTest\RegressionTest.prj" -execute -testSuiteID="Test
Suites\KeywordRegression\TS_AlertClassKeywordRegression" -browserType=Firefox
```

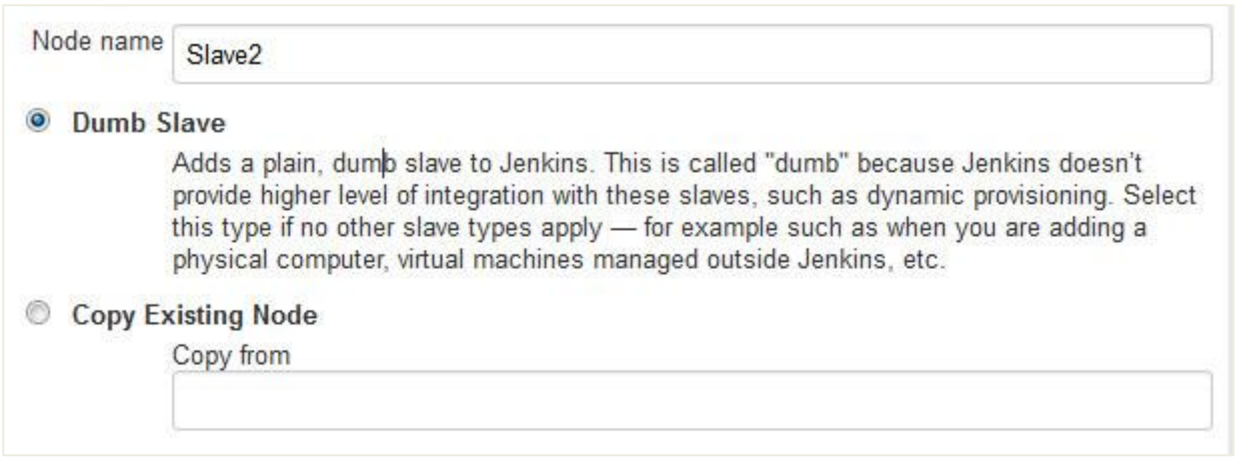
At this time, we can modify test project and commit code to SVN on machine A, then execute test suite on machine B via Jenkins job.

3. Create Jenkins slave node (optional)

When we want to execute test suites on multiple machines, we can create slave nodes on those machines and through Jenkins, designate jobs to them.

- **Create Jenkins slave node**

Login Jenkins > **Manage Jenkins** > **Manage Nodes** > **New Node**. Enter **Node name** and select **Dumb Slave**.

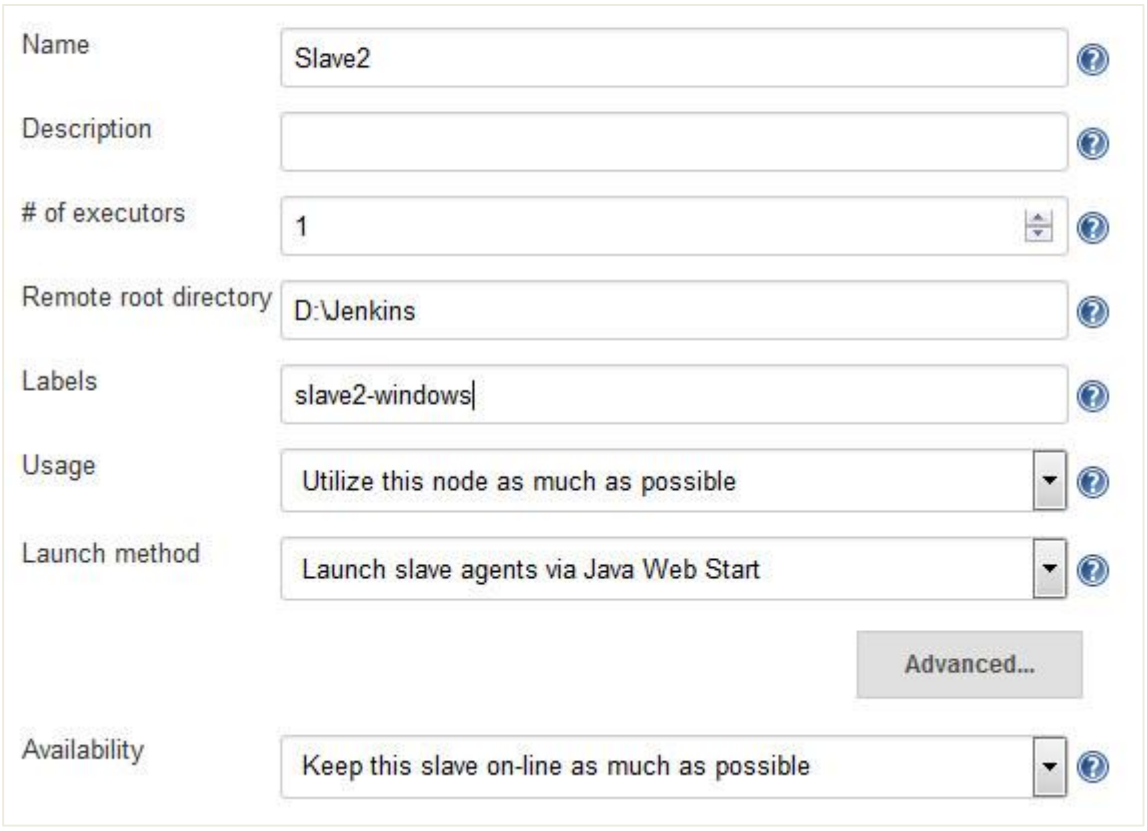

Node name: Slave2

[✓ Dumb Slave]
Adds a plain, dumb slave to Jenkins. This is called "dumb" because Jenkins doesn't provide higher level of integration with these slaves, such as dynamic provisioning. Select this type if no other slave types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Copy Existing Node
Copy from

Enter **Name**, **Remote root directory**, **Labels** for the slave.

At **Launch method** section, select **Launch slave agents via Java Web Start**.


Name	Slave2
Description	
# of executors	1
Remote root directory	D:\Jenkins
Labels	slave2-windows
Usage	Utilize this node as much as possible
Launch method	Launch slave agents via Java Web Start
Availability	Keep this slave on-line as much as possible

- **Start slave node**

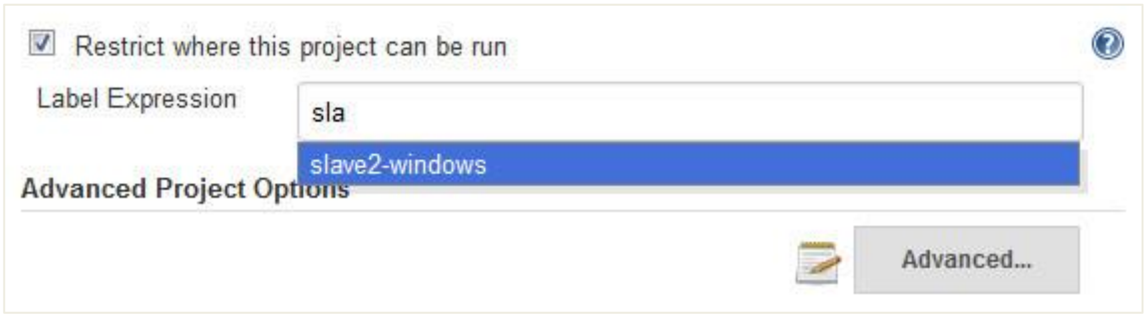
Assume we have machine C to act as slave node. From machine C, login **Jenkins** > **Manage Jenkins** > **Manage Nodes** > **{slave node}**. Click **slave.jar**, save the file to your selected location.

Open command prompt, navigate to **.jar** location and run the command line, for example:
```
java -jar slave.jar -jnlpUrl
http://192.168.50.58:9090/computer/Slave2/slave-agent.jnlp -secret
fc4e5fc20e966f64c75c74f3db5c6331dff5882a4d12371a745a9ec7c93980fb
```

- **Designate Jenkins job to slave node**

Login **Jenkins** > **{job}** > **Configure** > check **Restrict where this build can be run**

At **Label Expression**, enter slave's label.



☑ Restrict where this project can be run

Label Expression: sla

slave2-windows