

Continuous analysis with Team Foundation Server 2017

This guide will show you how to integrate Kiuwan into Team Foundation Server 2017.

Contents:

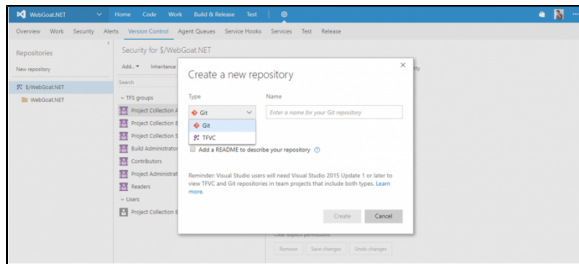
- [1. Install the KLA in the agent machines](#)
- [2. Select an existing project in TFS or create a new one](#)
- [3. Configure a new build step](#)
- [4. Define variables](#)
- [5. Configure a new build step](#)
- [6. Configure the command line task](#)
- [7. Define triggers to run the build](#)
- [8. Run a build from Visual Studio and see the results](#)

1. Install the KLA in the agent machines

See how to install KLA here: [Install and Start Up Kiuwan Local Analyzer](#)

2. Select an existing project in TFS or create a new one

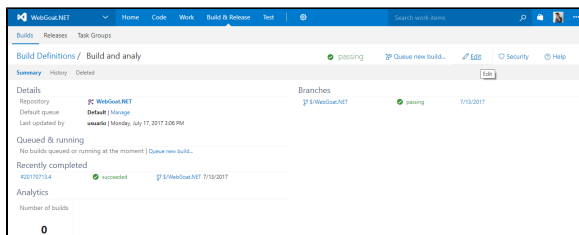
In TFS, every project must have one or more associated Version Control repositories. This way the build agents take care of extracting the source code automatically. In TFS 2017, the supported repositories are Git and TFS itself.



For existing projects, it is likely that the repositories are already configured and the code should be already there. For new projects, add your code to version control from Visual Studio. The repositories you configure here will be the ones that new Build definitions will use by default.

3. Configure a new build step

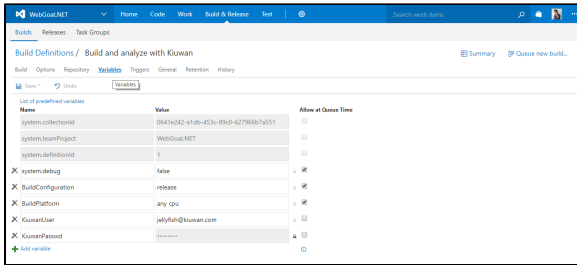
The new build step will run the Kiuwan analysis in a project's existing build configuration. It is possible to also create a new build configuration to include the Kiuwan analysis step.




Click the build definition edit link to add new build steps.

4. Define variables

In the **Build Definitions** settings screen, define two variables that will be available for the build agent at execution time, to hold the Kiuwan credentials necessary to run the analysis. To do this, first look for the variables section in the top **Build definition** sub-menu.

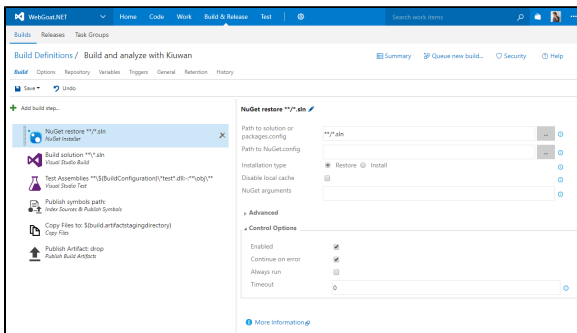


Click **Add variable** and give the new variables name such as KiuwanUser and KiuwanPasswd.

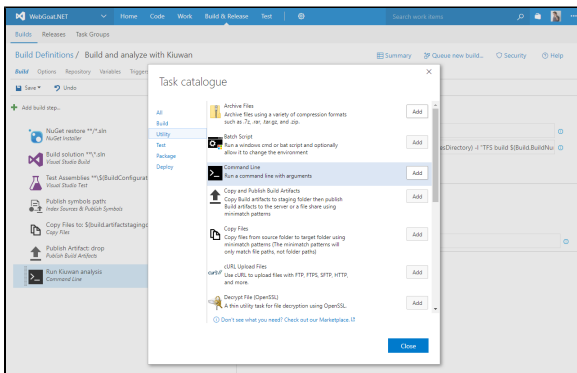
 Click the lock icon to the left of the password variable value to hide it!

5. Configure a new build step

Go back to the **Build Definition configuration page**. Click **Add build step...** to configure a new build step.



Depending on the build definition template you selected to create your build, you will have some pre-configured build steps.



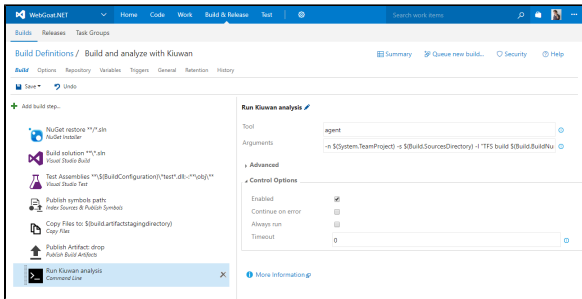
From the **Task catalogue**, in the **Utility** section, select the **Command Line** task.

6. Configure the command line task

Configure the command line task to run a Kiuwan analysis using the Kiuwan Local Analyzer CLI.

Title	Description
Name	Run Kiuwan Analysis (in this case)
Tool	agent

Arguments	<pre>-n \$(System.TeamProject) -s \$(Build.SourcesDirectory) -l "TFS build \$(Build.BuildNumber)" --user \$(KiuwanUser) --pass \$(KiuwanPasswd)</pre> <p>We are using System and Build predefined variables from TFS for the name of the application to analyze (it will be the project name), the directory where the source code is (it will be the checkout directory), and the label for the analysis (it will be the build number).</p> <p>Finally, we use the two variables we previously defined for the Kiuwan credentials. Using variables ensures consistency for every analysis and make them independent of the build agent machines that run them.</p>
Control Options	Continue on error = if checked, the build continues even if the analysis fails (the analysis could not run for some reason, it does not affect the results of the analysis).



This configuration will run a Kiuwan Baseline analysis.

If you want to run a delivery analysis for a change request, you can define another build configuration where the analysis step will have extra parameters for the KLA agent to run a delivery analysis.

Refer to the Kiuwan Local Analyzer documentation for more information on the available parameters.

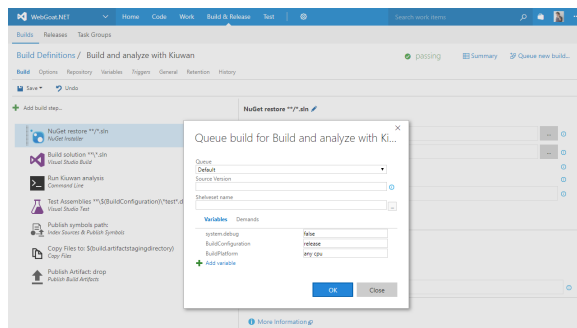
7. Define triggers to run the build

The last thing is to define triggers to run the build on the project on given conditions.

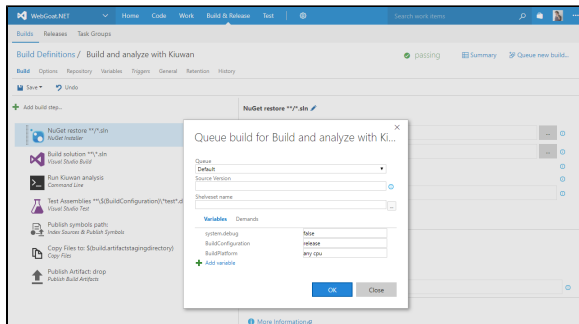
The possibilities here are:

Description	Image
-------------	-------

1. **Continuous integration (CI):** Run the build when changes are checked-in in the repository (this will work when checking-in changes from Visual Studio)
2. **Scheduled:** Run the build periodically based on a defined schedule.
3. **Gated Check-in**

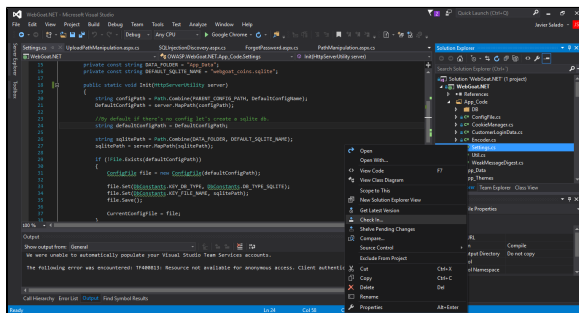


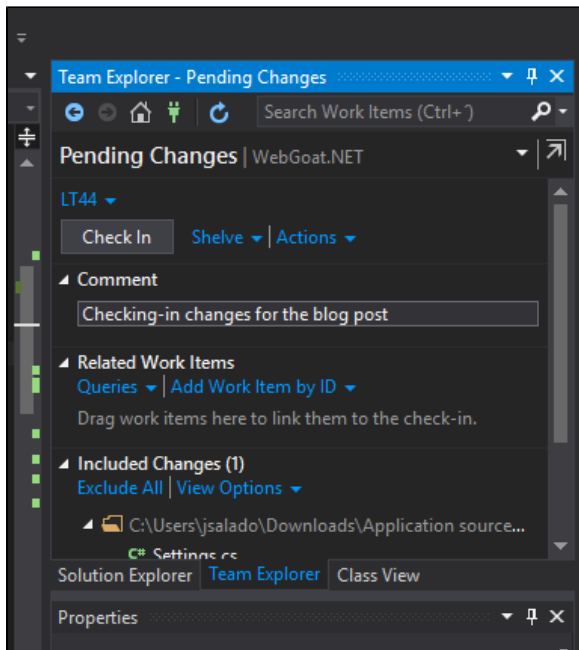
For testing purposes, click **Queue new build...** to run the build.



8. Run a build from Visual Studio and see the results

The configured build will trigger automatically after checking-in and committing changes to the project files from Visual Studio.

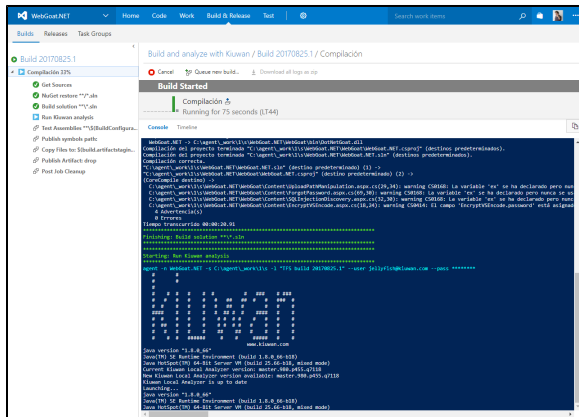




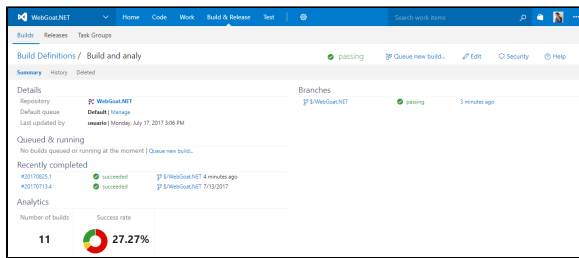
Once the check-in finishes in Visual Studio, go to TFS to confirm that the "Build and analyze with Kiwan" build has been queued and is running:

Agents for pool Default								
Agents	Roles							
Enabled	Name	Current Status						
✓	LT44	Running build 20170825.1						
✓	TFS Server	Idle						
			Requests	Capabilities				
ID	Type	Definition	Name	Date Queued	Date Assigned			
11	Build	Build and analyze with ...	20170825.1	8/25/2017 1:49:31 ...	8/25/2017 1:49:31			
10	Build	Build and analyze with ...	20170713.4	7/13/2017 11:50:00 ...	7/13/2017 11:50:00			
9	Build	Build and analyze with ...	20170713.3	7/13/2017 11:17:55 ...	7/13/2017 11:17:55			
8	Build	Build and analyze with ...	20170713.2	7/13/2017 9:36:43 ...	7/13/2017 9:36:43			
7	Build	Build and analyze with ...	20170713.1	7/13/2017 9:35:08 ...	7/13/2017 9:35:08			
6	Build	Build and analyze with ...	20170712.6	7/12/2017 9:52:48 ...	7/12/2017 9:52:48			
5	Build	Build and analyze with ...	20170712.5	7/12/2017 9:48:05 ...	7/12/2017 9:48:05			
4	Build	Build and analyze with ...	20170712.4	7/12/2017 9:45:49 ...	7/12/2017 9:45:49			
3	Build	Build and analyze with ...	20170712.3	7/12/2017 9:11:04 ...	7/12/2017 9:11:04			
2	Build	Build and analyze with ...	20170712.2	7/12/2017 10:02:3 ...	7/12/2017 10:02:3			
1	Build	Build and analyze with ...	20170712.1	7/12/2017 9:57:56 ...	7/12/2017 9:57:56			

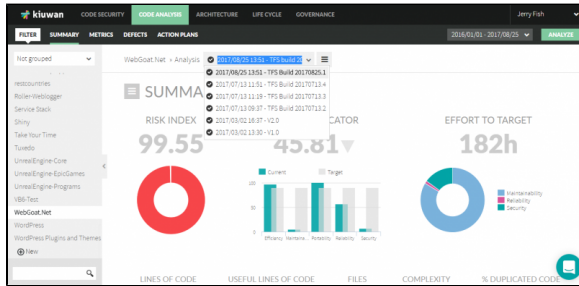
To see the console output from the build agent and the progress of the different steps, click the build name in the table on the right.



After all the steps finished successfully, you can view the build definition summary with the results of the recent builds and more historical information.



Login to your Kiwan account in the cloud to see the results of this build analysis.



The screenshot shows the 'kiwan CODE SECURITY' dashboard. The 'VULNERABILITIES' section displays a list of vulnerabilities. The table has columns for 'Severity', 'Count', 'Description', 'Category', 'Status', 'Risk', 'Action', and 'Time'. The first vulnerability is 'WASC09 Improper initialization of CRLF Sequences in HTTP headers' with a severity of 'High' and a risk of '113'. The second vulnerability is 'OWASP2013A1 POENR4.3.1 WASC09 WebGoat.NET/WebGoat/Goat/CustomLogin.aspx.cs' with a severity of 'Medium' and a risk of '113'. The third vulnerability is 'Sqli at line 60' with a severity of 'High' and a risk of '113'. The fourth vulnerability is 'Sink at line 72' with a severity of 'High' and a risk of '113'. The fifth vulnerability is 'WebGoat.NET/WebGoat/Content/ForgePassword.aspx.cs' with a severity of 'High' and a risk of '113'.

Severity	Count	Description	Category	Status	Risk	Action	Time
High	5	WASC09 Improper initialization of CRLF Sequences in HTTP headers	OWASP2013A1 POENR4.3.1 WASC09	Open	113	Security	3h 00
Medium	2	WebGoat.NET/WebGoat/Goat/CustomLogin.aspx.cs	OWASP2013A1 POENR4.3.1 WASC09	Open	113	Security	3h 00
High	1	Sqli at line 60	OWASP2013A1 POENR4.3.1 WASC09	Open	113	Security	3h 00
High	1	Sink at line 72	OWASP2013A1 POENR4.3.1 WASC09	Open	113	Security	3h 00
High	1	WebGoat.NET/WebGoat/Content/ForgePassword.aspx.cs	OWASP2013A1 POENR4.3.1 WASC09	Open	113	Security	3h 00