

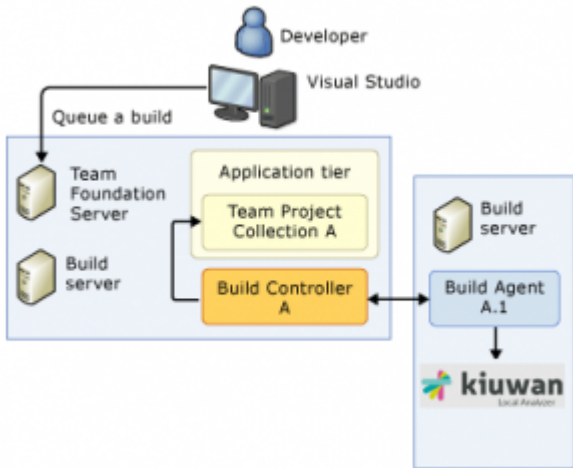
Continuous Inspection with Team Foundation Server

This guide will show you how to integrate Kiuwan inspections in your software development cycle using Team Foundation Server.

Contents:

- [Requirements](#)
- [1. Download sample application](#)
- [2. Add the application to Source Control](#)
- [3. Build definition](#)
- [4. Commit the changes and get the analysis results in Kiuwan](#)
- [5. Compare each commit against the previous](#)

Requirements

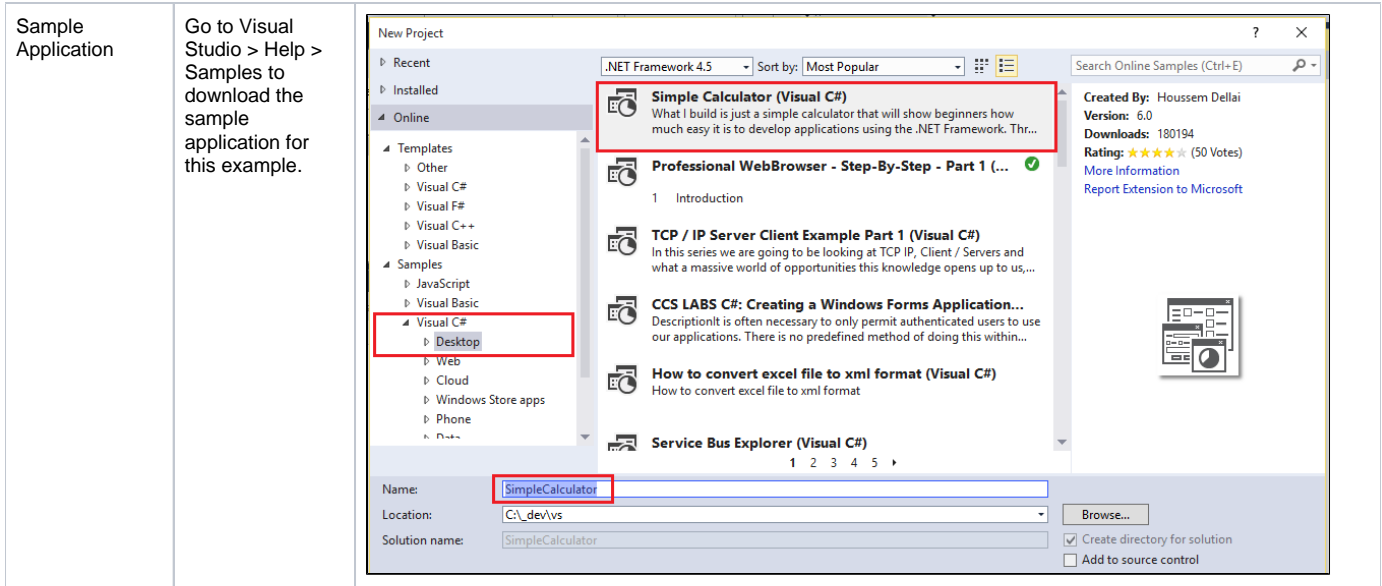
Requirement	Description	Image
Visual Studio 2013 and Team Foundation Server 2013	Both installed "on-premises".	
Kiuwan Local Analyzer	<p>Installed on every machine where a TFS Build Agent is installed.</p> <p>Log into your Kiuwan Account via the KLA to ensure a working connection.</p> <p>Your credentials will be saved in a cipher form for subsequent analysis when they are run from the command line interface as well.</p> <p>Assuming kiuwan has been installed in c:\, we need to create a small script that will be invoked from the TFS build agent.</p>	 <p>The diagram illustrates the integration of Kiuwan with Team Foundation Server (TFS). A Developer uses Visual Studio to 'Queue a build'. This triggers a process within the TFS infrastructure, which includes a 'Team Foundation Server' and a 'Build server'. The build is processed by the 'Application tier' and 'Team Project Collection A', which then sends the build to the 'Build Controller A'. The 'Build Controller A' is connected to a 'Build Agent A.1'. The 'Build Agent A.1' is shown with the Kiuwan logo and the text 'Local Analyzer', indicating that Kiuwan's analysis is performed locally on the build agent.</p>

```
@echo off
::
tfs2kiuwan
.cmd
:: script
to launch
kiuwan
analysis
from team
foundation
server
build.
::
-----
-----
-----
-----
-----
-----
-----
setlocal

set
KIUWAN_HOM
E=C:
\KiuwanLocalAnalyzer
set
KIUWAN=%
KIUWAN_HOM
E%
\bin\agent
.cmd

%KIUWAN% -
c -n "%
TF_BUILD_B
UILDDEFINI
TIONNAME%"
-l "%
TF_BUILD_B
UILDNUMBER
%" -s "%
TF_BUILD_S
OURCESDIRE
CTORY%"

endlocal
```

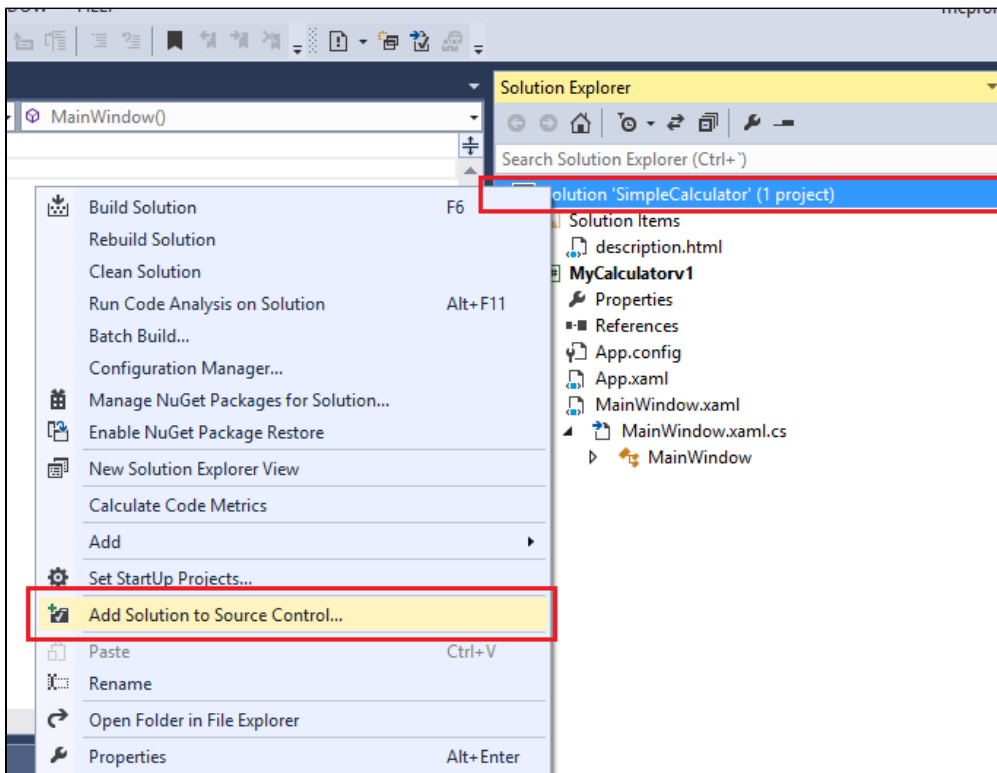


1. Download sample application

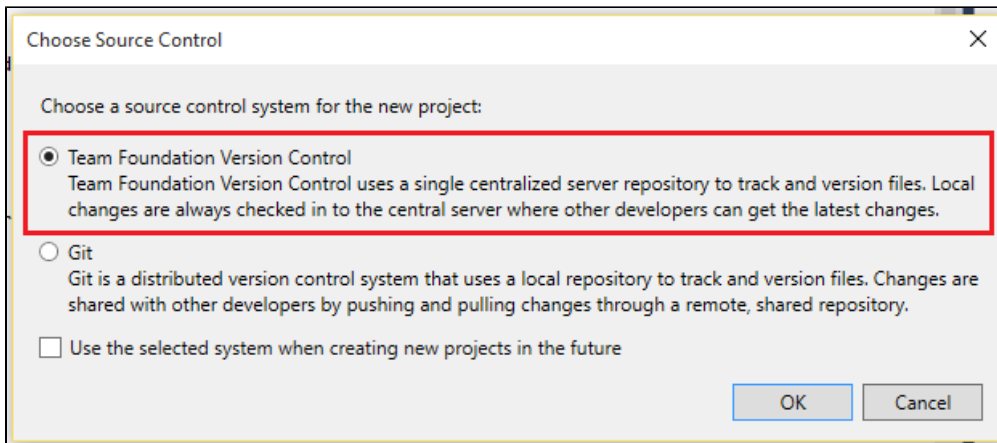
Download the sample application as described in the requirements.

2. Add the application to Source Control

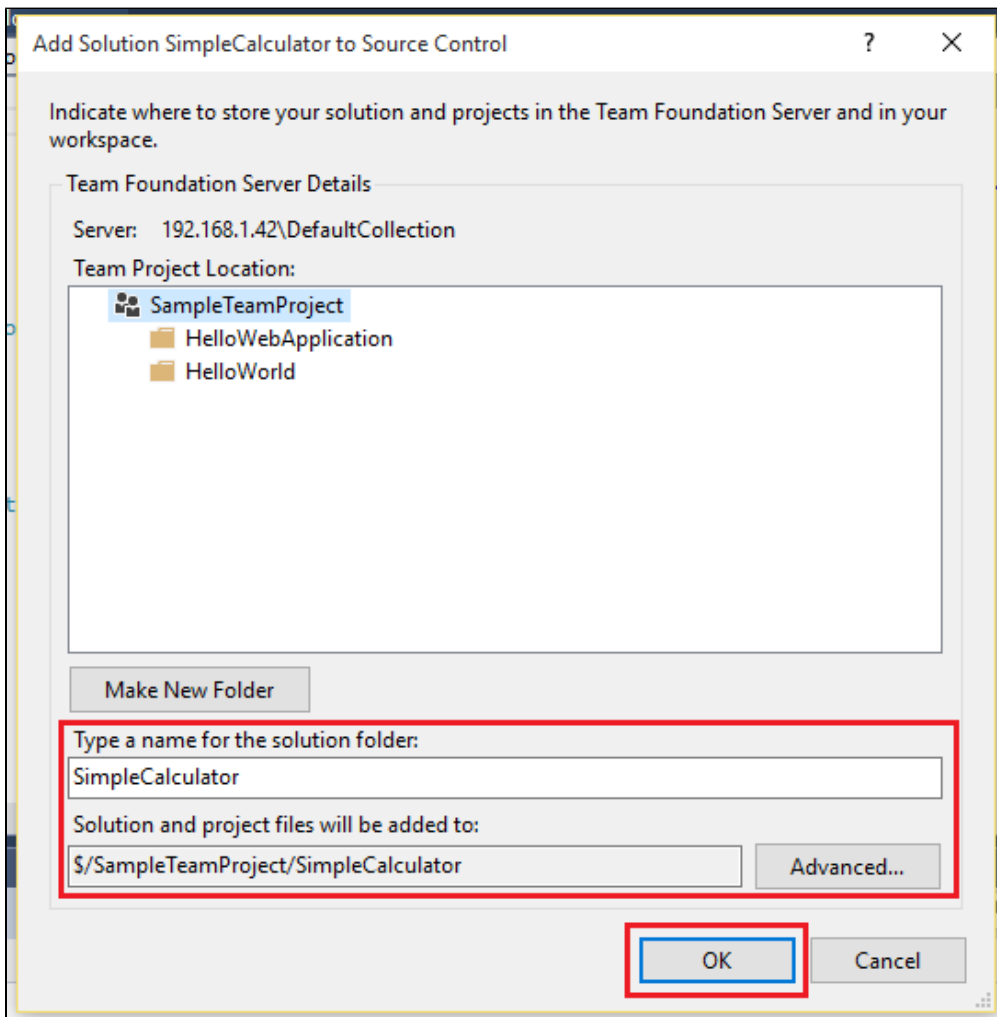
In **Solution Explorer**, right-click on the **SimpleCalculator** solution. Then go to **Add Solution to Source Control**.



A new dialog appears: **Choose Source Control** backend. Select **Team Foundation Version Control**.



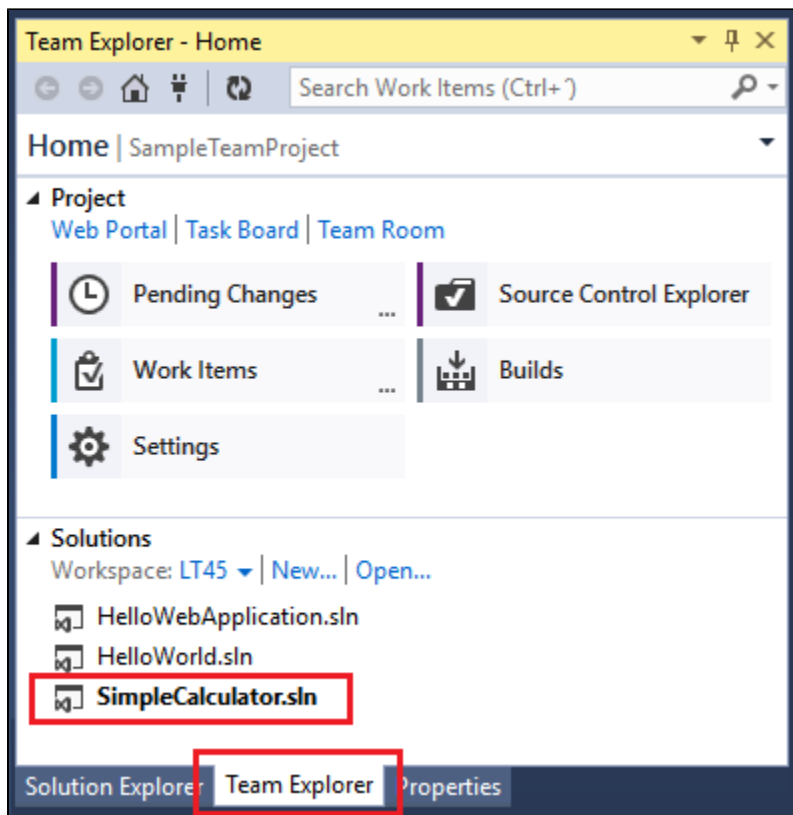
Finally, select a **TeamProject** where to place this project and create the solution folder.



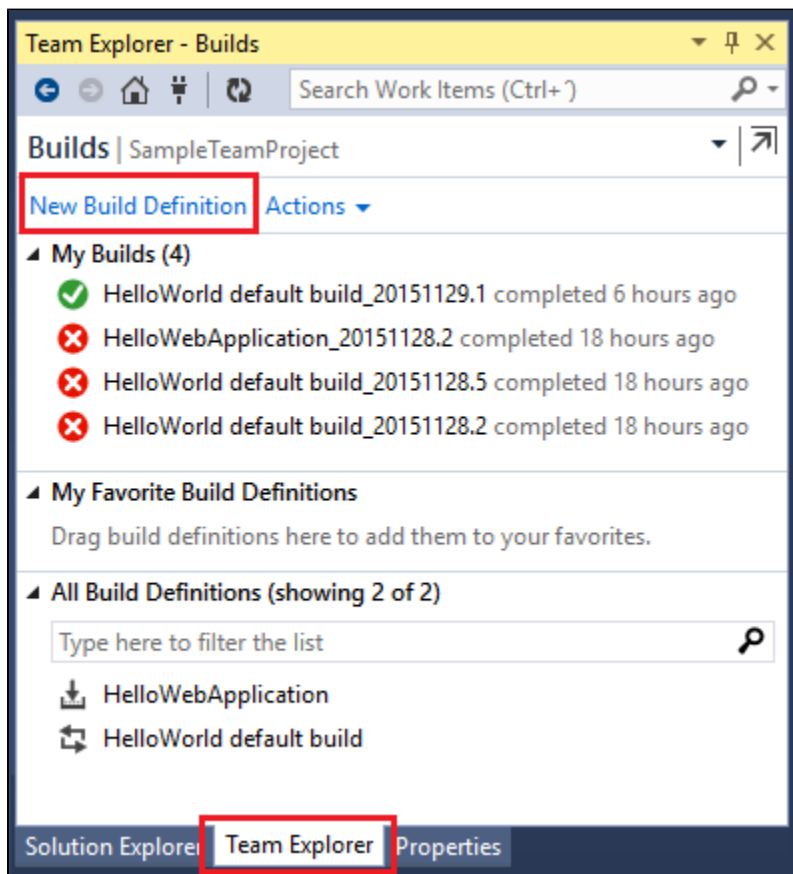
3. Build definition

Once the solution is created in TFS, define a new build process for the solution.

Open the **Team Explorer** tab.



Double click on **Builds** and select **New build definition**.



Below, the images of the dialog boxes to configure the build definition:

Build Explorer - SampleTeamProject MyCalculatorv1 SimpleCalculator -> X MainWindow.xaml.cs Simple Calculator << >>

General

Build definition name: SimpleCalculator

Trigger

Source Settings

Build Defaults

Process

Retention Policy

Description (optional):

Run Kiwan inspection each check-in.
- Build definition name will be used as kiwan application name
- Build number will be used as analysis label.

Queue processing:

☒ Enabled
Requests queued by users or triggered by the system will be added to the queue and be started in priority order.

☐ Paused
Requests queued by users or triggered by the system will be added to the queue but will not start unless the build administrator forces them to start.

☐ Disabled

Build Explorer - SampleTeamProject MyCalculatorv1 SimpleCalculator -> X MainWindow.xaml.cs Simple Calculator << >>

General

Trigger

Source Settings

Build Defaults

Process

Retention Policy

Select one of the following triggers:

☐ Manual - Check-ins do not trigger a new build

☒ Continuous Integration - Build each check-in

☐ Rolling builds - accumulate check-ins until the prior build finishes

☐ Build no more often than every minutes.

☐ Gated Check-in - accept check-ins only if the submitted changes merge and build successfully

☐ Merge and build up to submissions.

Build Explorer - SampleTeamProject MyCalculatorv1 SimpleCalculator -> X MainWindow.xaml.cs Simple Calculator << >>

General

Trigger

Source Settings

Build Defaults

Process

Retention Policy

Specify the build controller and staging location for this build definition. These selections may be modified by the person queuing the build.

Build controller: win7en32 - Controller

Description:

Staging location:

☒ This build does not copy output files to a drop folder

☐ Copy build output to the following drop folder (UNC path, such as \\server\share):

☐ Copy build output to the server

SimpleCalculator -> X Build Explorer - SampleTeamProject MyCalculatorv1 MainWindow.xaml.cs Simple Calculator << >>

General

Trigger

Source Settings

Build Defaults

Process

Retention Policy

Team Foundation Build uses a build process template defined by a Windows Workflow (XAML) file. The behavior of this template can be customized by setting the build process parameters provided by the selected template.

Build process template: Default Template [Show details](#)

Build process parameters:

> 1. TF Version Control	
v 2. Build	
1. Projects	\$/SampleTeamProject/SimpleCalculator/SimpleCalculator.sln
2. Configurations	
3. Clean build	True
4. Output location	SingleFolder
v 5. Advanced	
MSBuild arguments	
MSBuild platform	Auto
Perform code analysis	AsConfigured
Post-build script arguments	SimpleCalculator
Post-build script path	C:\scripts\tfs2kiwan.cmd
Pre-build script arguments	

Path to publish symbols

v 5. Advanced

> Agent settings Use agent where Name=* and Tags=[] (MatchExactly)

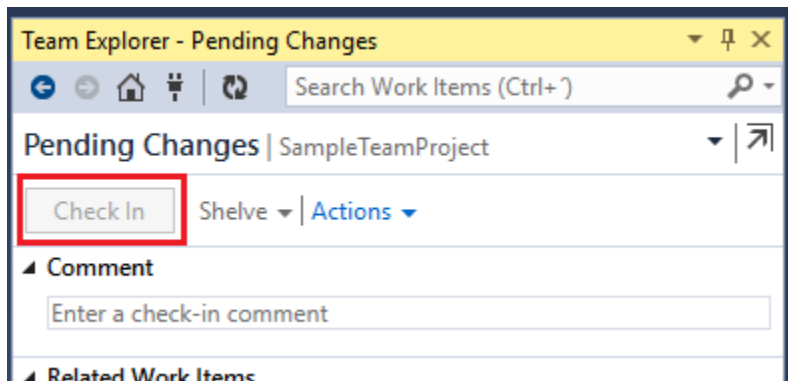
Build number format tfs ci \$(Rev:.r)

Create work item on failure True

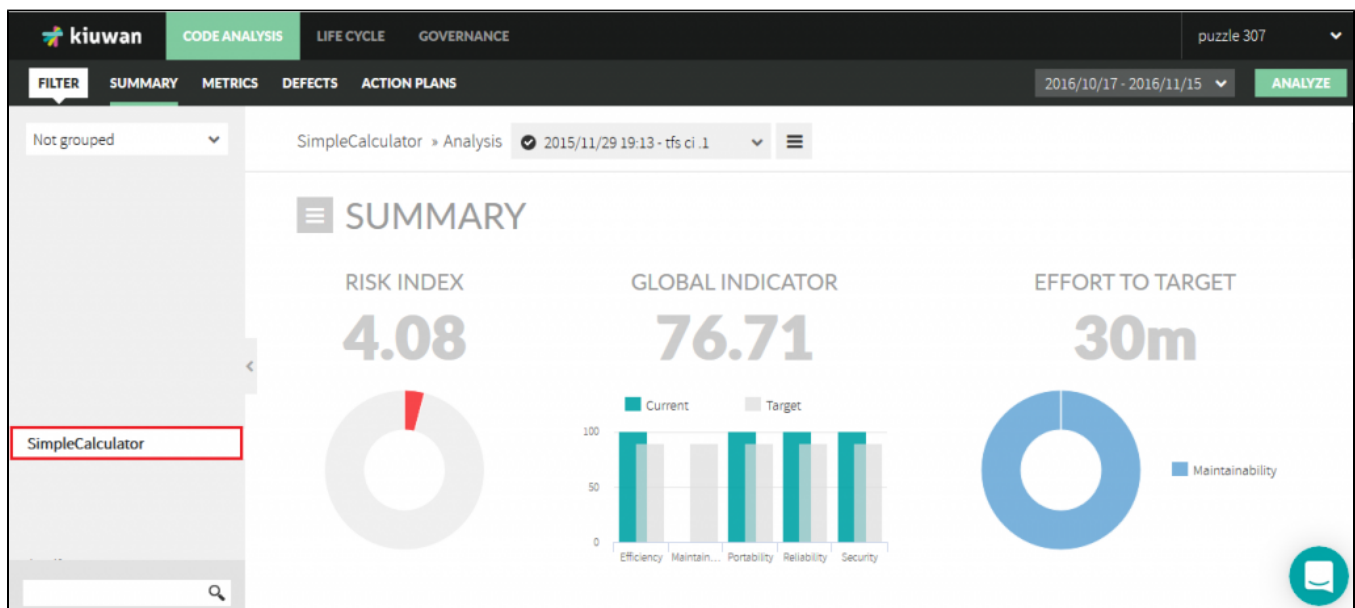
Update work items with build number True

4. Commit the changes and get the analysis results in Kiuwan

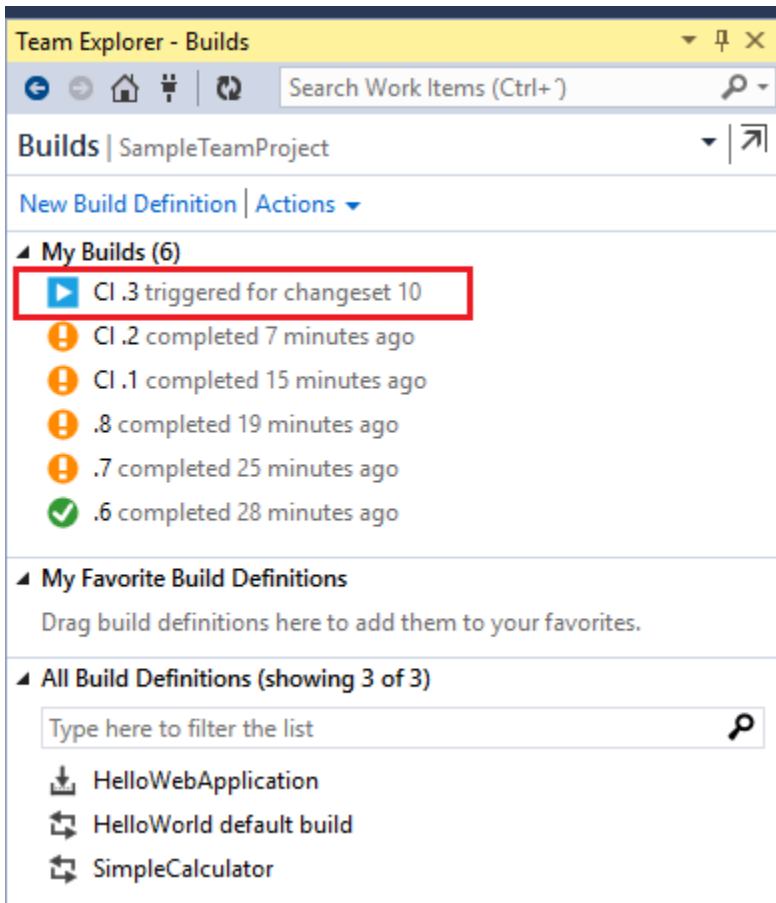
Commit the changes to the repository. A new build will be automatically triggered.



Login to kiuwan.com to see the results for your application analysis.

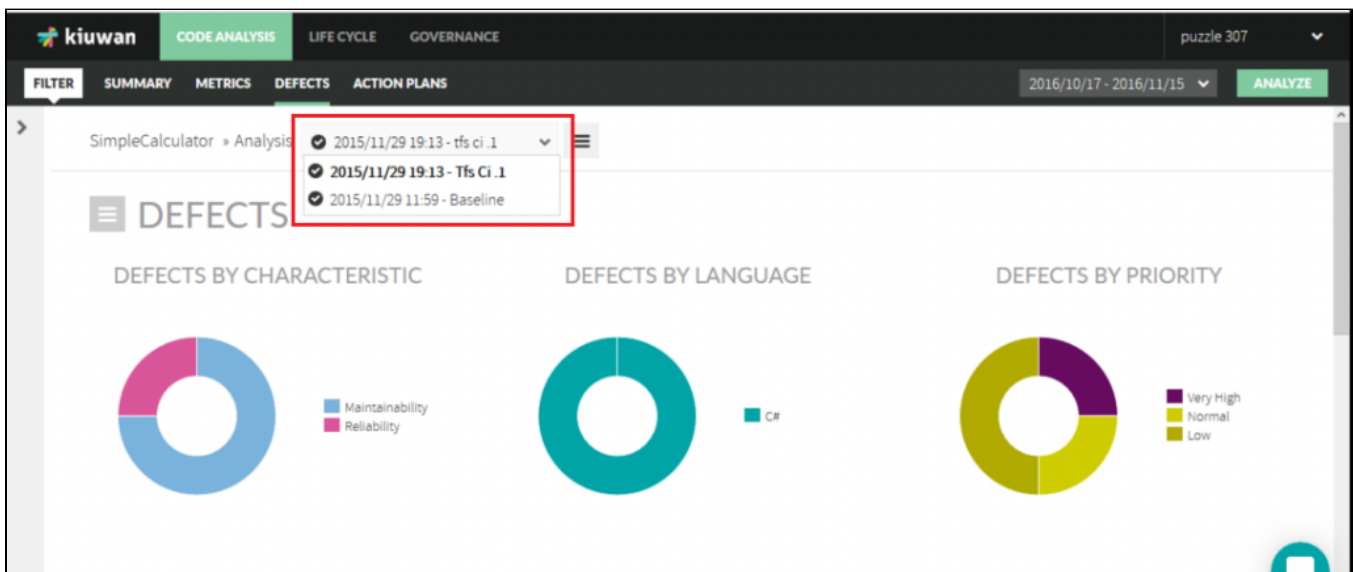


For each commit in the repository, a new build and analysis will be run automatically:

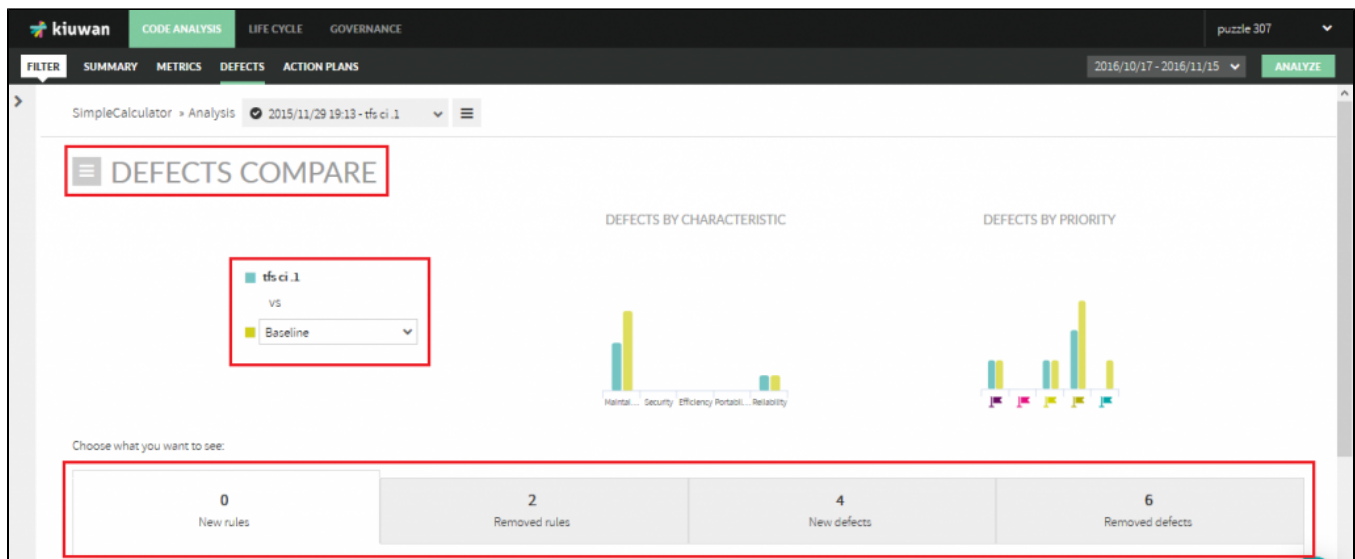


5. Compare each commit against the previous

Each analysis will generate a new version in Kiuwan. You can see all the builds in the **Analysis** drop-down list:



Click **Compare** to see a defect comparison between both analyses.



Now you have a continuous analysis of all your builds in Kiuwan.