Continuous Integration &  
Continuous Delivery

CGI | [Company address]

Setup CI & CD of Skånetrafikens CRM solution

Niklas Henricson

2016

Table of Contents

[Project Configuration 2](#_Toc446728017)

[Introduction 2](#_Toc446728018)

[Overview – Ska skrivas om och översättas till engelska 2](#_Toc446728019)

[Installation 3](#_Toc446728020)

[Setup client computer 3](#_Toc446728021)

[Overview 3](#_Toc446728022)

[Install Visual Studio 2012 3](#_Toc446728023)

[Install XRM CI Framework 3](#_Toc446728024)

[TFS Build 2015 4](#_Toc446728025)

[Windows Identity Framework 4](#_Toc446728026)

[CRM Toolkit 4](#_Toc446728027)

[CRM Templets 4](#_Toc446728028)

[Setup build server 4](#_Toc446728029)

[Overview 4](#_Toc446728030)

[Install Visual Studio 2015 4](#_Toc446728031)

[Install Team Foundation Server 2015 8](#_Toc446728032)

[Team Foundation Server Configuration Center 9](#_Toc446728033)

[When the Build Server doesn’t belong to the same domain as the TFS server 19](#_Toc446728034)

[Pre-work 20](#_Toc446728035)

[Architecture 20](#_Toc446728036)

[Process 20](#_Toc446728037)

[Ports 24](#_Toc446728038)

[Xrm CI Framwork 24](#_Toc446728039)

[Project Types 24](#_Toc446728040)

[Config Types 25](#_Toc446728041)

[Build Definitions CI 25](#_Toc446728042)

[Automated Tests 25](#_Toc446728043)

[Continuous Delivery 25](#_Toc446728044)

[Creating & Setting up XAML Build Definition 26](#_Toc446728045)

Continuous Integration & Continuous Delivery of Skånetrafikens CRM Solution

# Project Configuration

<https://support.microsoft.com/en-us/kb/555396>

# Introduction

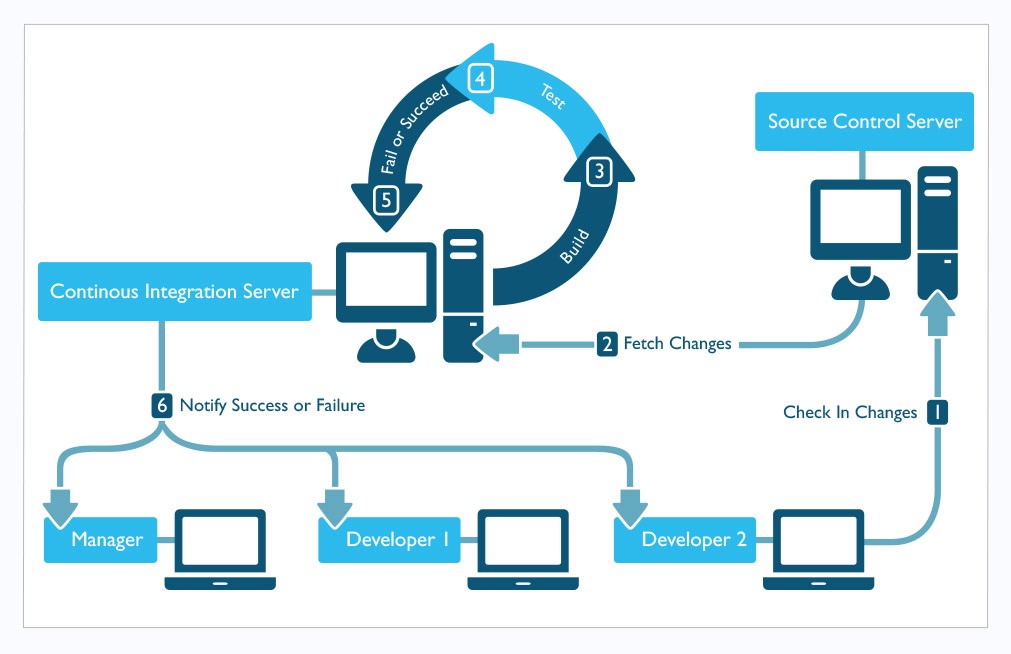
The book “Continuous Delivery with Visual Studio ALM 2015” describes Continuous Integration and Continuous Delivery as the following:

*“Continuous Integration is all about quick feedback and validation of the commit phase, and Continuous Delivery is about establishing the mindset where you can deliver features at customer demand”*

In this document we will describe the necessary steps to install and configure Continuous Integration (CI) and Continuous Delivery (CD) of Skånetrafikens CRM Solution.

The solution involves one build server (DK-BYGGSERVER) and 4 environments (Development, Test, Q, and Production).

# Overview – Ska skrivas om och översättas till engelska



# Installation

## Setup client computer

### Overview

1. Install Visual Studio 2012
2. Install XRM CI Framework
3. Install Windows Identity Framework
4. Configure Visual Studio
5. Install CRM Templates

### Install Visual Studio 2012

* Download Visual Studio 2012
* Install Visual Studio 2012

### Install XRM CI Framework

* Install Xrm CI Framework
* Place the XRM CI Framework in the “Common Library” folder in TFS.
* Turn on “Windows identification” through the Add/remove Windows features which can be found in the Control Panel (Figure 1 - Windows Identity Foundation 3.5)

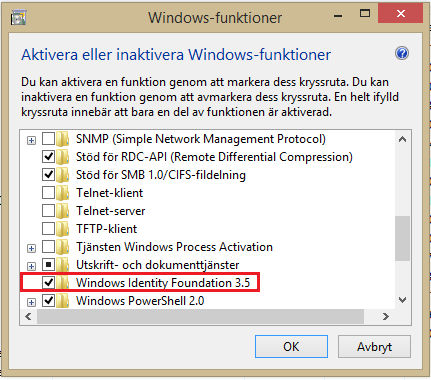


Figure 1 - Windows Identity Foundation 3.5

* Download Microsoft Dynamics CRM 2013 Software Development Kit (SDK)
* Install Visual Studio 2012 CRM Solution Templates via SDK\Templates (Figure 2 - CRMSDKTemplates for Visual Studio 2012)
* Install CrmDeveloperToolsVS12\_Installer (se Figure 3 - CrmDeveloperToolsVS12\_Installer)



Figure 2 - CRMSDKTemplates for Visual Studio 2012

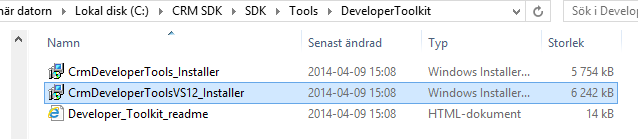


Figure 3 - CrmDeveloperToolsVS12\_Installer

TFS Build 2015

* Connect TFS with Visual Studio
* Login with your username and password in TFS through Visual Studio
* Get the latest code from “**Sekund**” and “**integrationer**”.

Windows Identity Framework

* Install Windows Identity Framework

CRM Toolkit

* Install CRM Toolkit

CRM Templets

* Install CRM Templates

## Setup build server

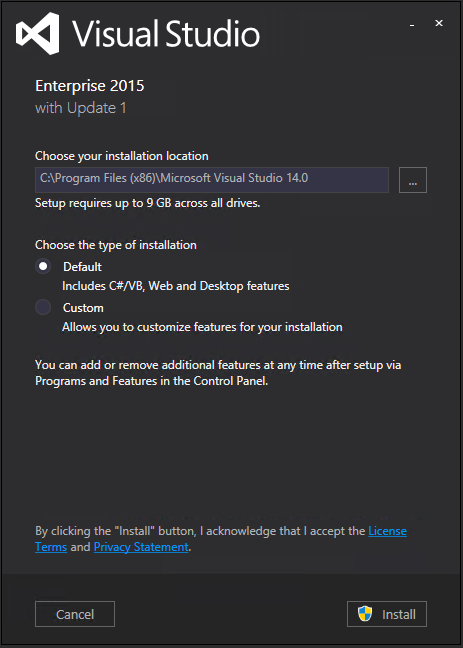
Tar upp installationsstegen för Visual Studio, Team Foundation Build Server och login-/service-lösningen

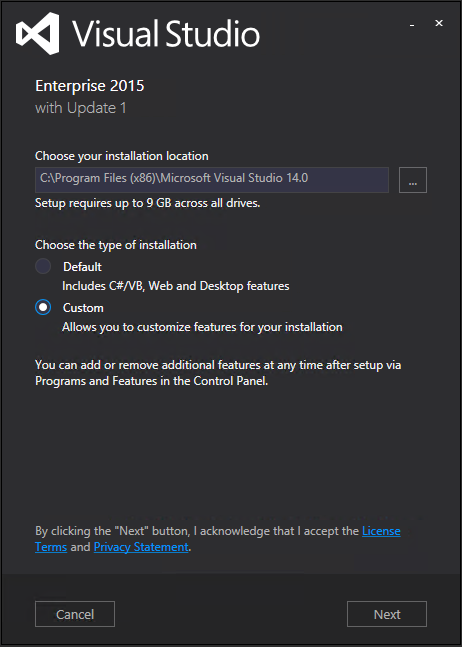
### Overview

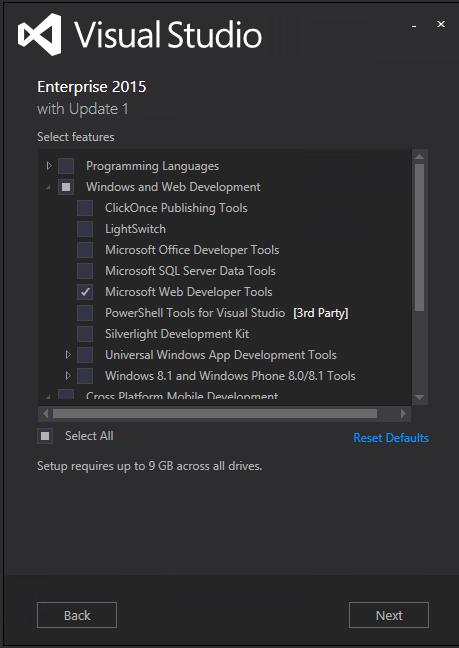
1. Install Visual Studio 2015
2. Install Team Foundation Server 2015 Update 1
3. Create local accounts for build service (sa-se-Skanetrafiken) with the same password
4. Make sure that the build service is running under the sa-se-Skanetrafiken account “Visual Studio Build Service Host 2015”
5. Open Team Foundation Server Administration Console and start the build service
6. If you receive unauthorized access error, it means the sa-se-Skanetrafiken account needs to be added as *Project Collection Build Service Accounts* in the Skanetrafiken collection (see screenshot below)

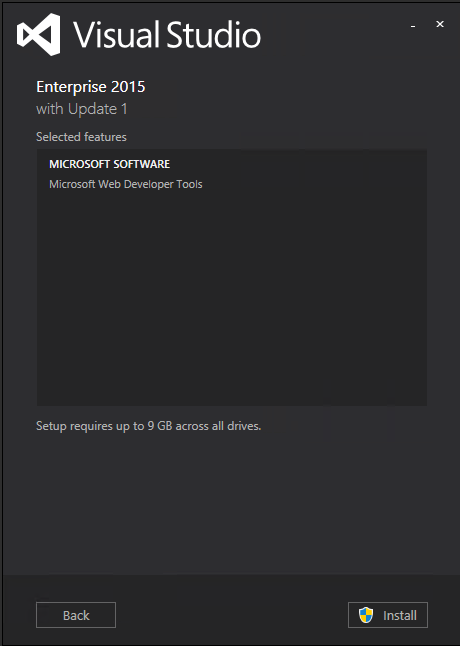
### Install Visual Studio 2015

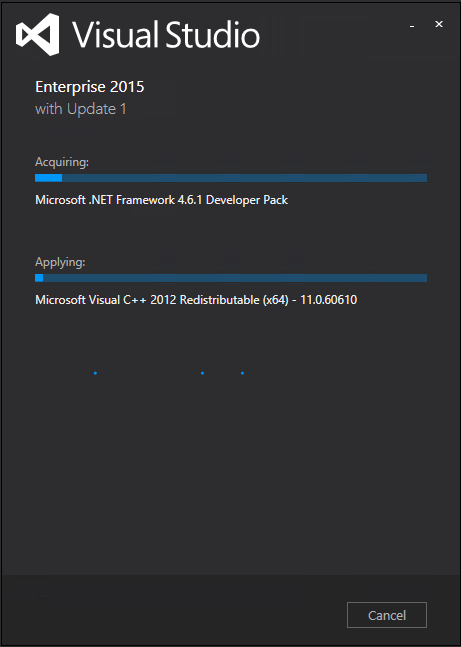
In this case we are installing Visual Studio Enterprise 2015 do enable more functionality. This is however not necessary in order to setup CI.





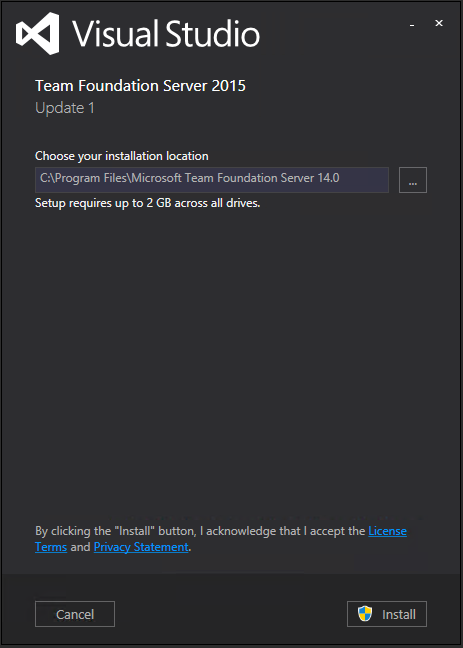


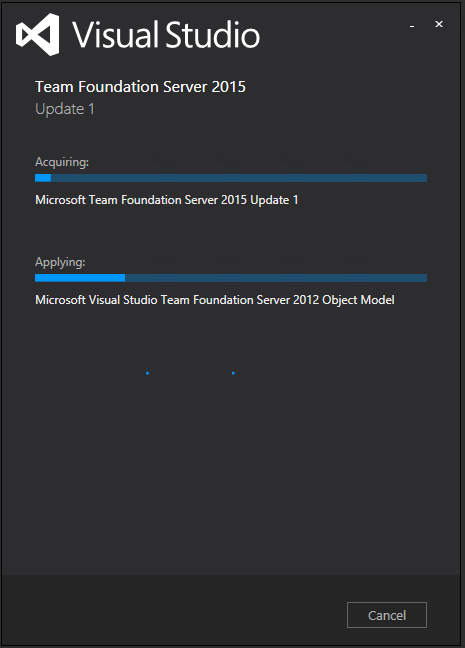




### Install Team Foundation Server 2015

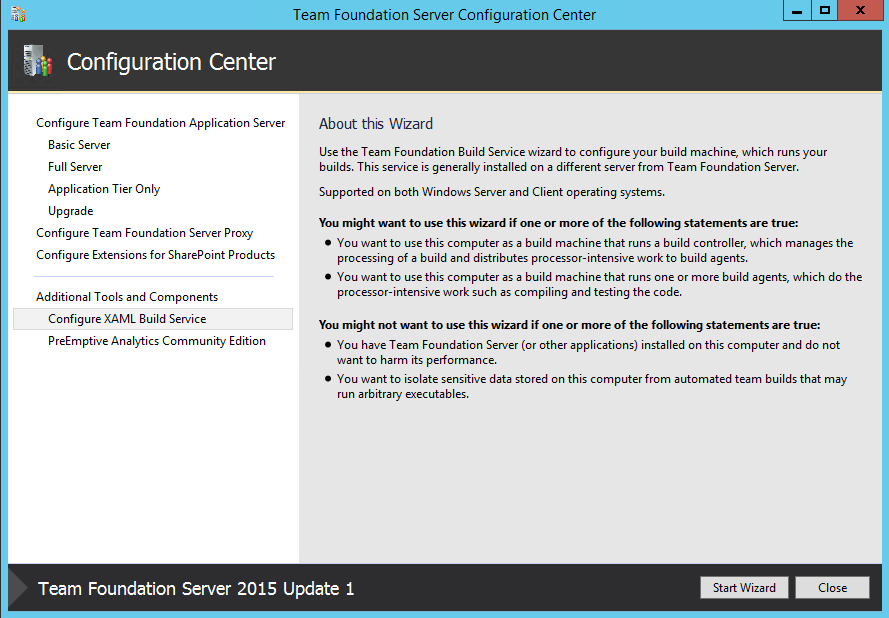
In this case we are installing Team Foundation Server 2015 with Update 1.

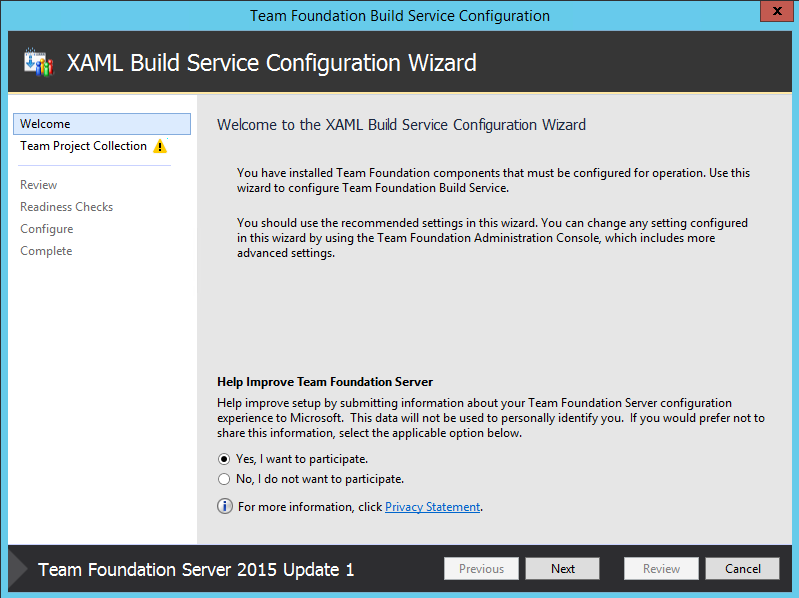




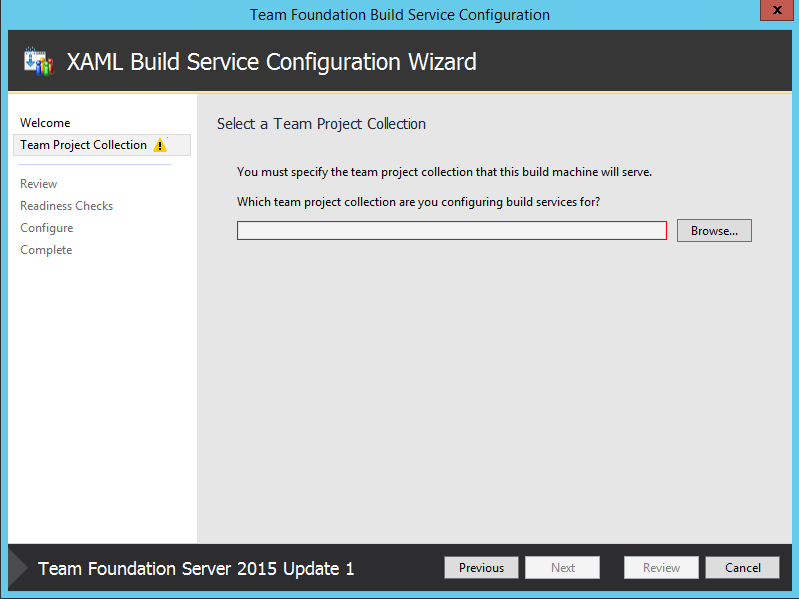
### Team Foundation Server Configuration Center

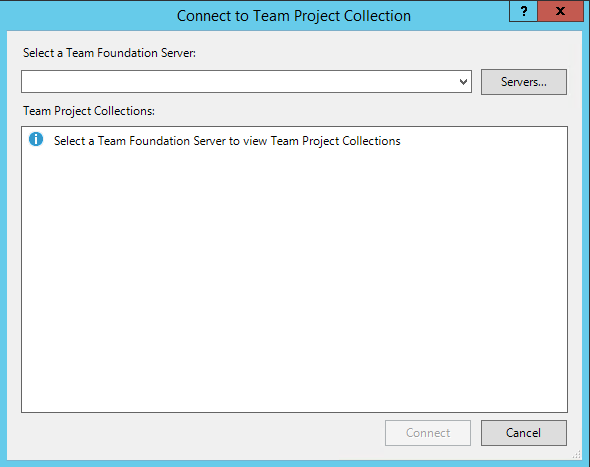
To setup a build server in order for the Continuous integration to work we have to do the following steps. The setup will show how to configure Team Foundation Server for Skånetrafikens CRM Solution.



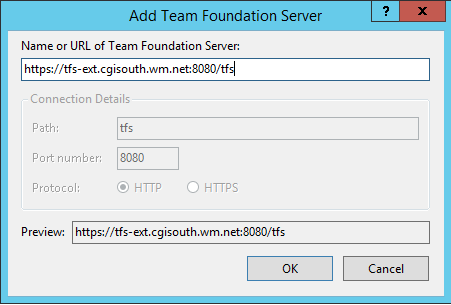


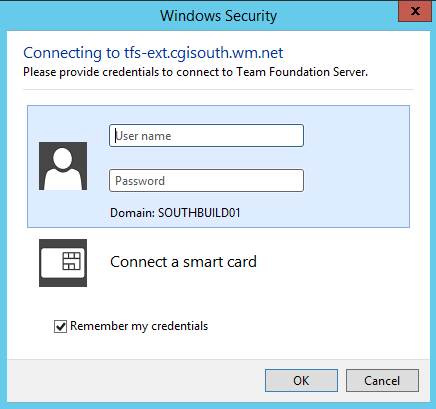
Team Project: <https://tfs-ext.cgisouth.wm.net:8080/tfs/Skanetrafiken>



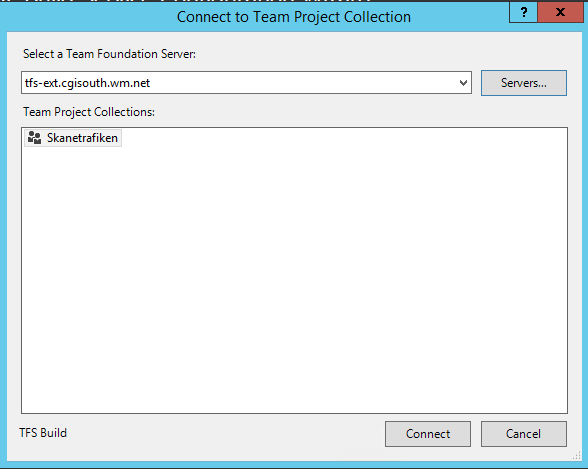


Register a TFS server (<https://tfs-ext.cgisouth.wm.net:8080/tfs>)

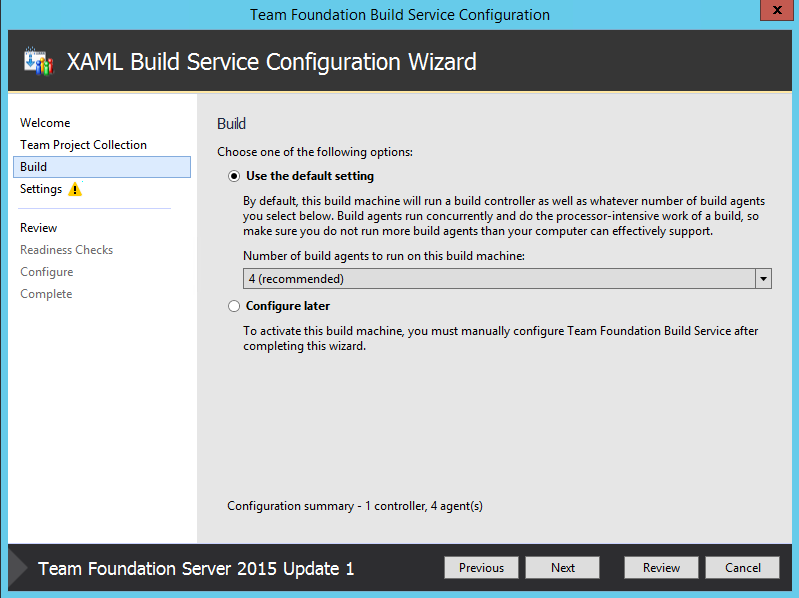




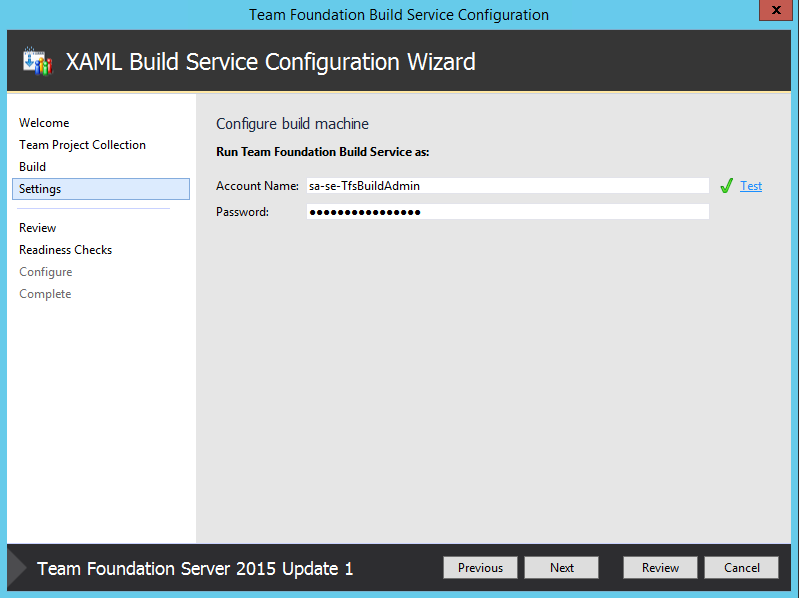
Login into the TFS server with your credentials



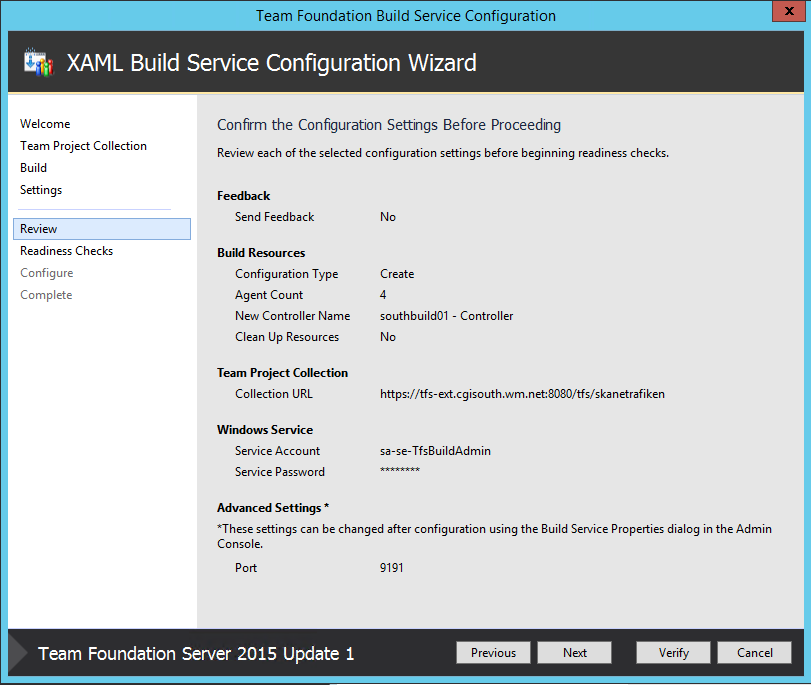
Choose collection name (Skanetrafiken)



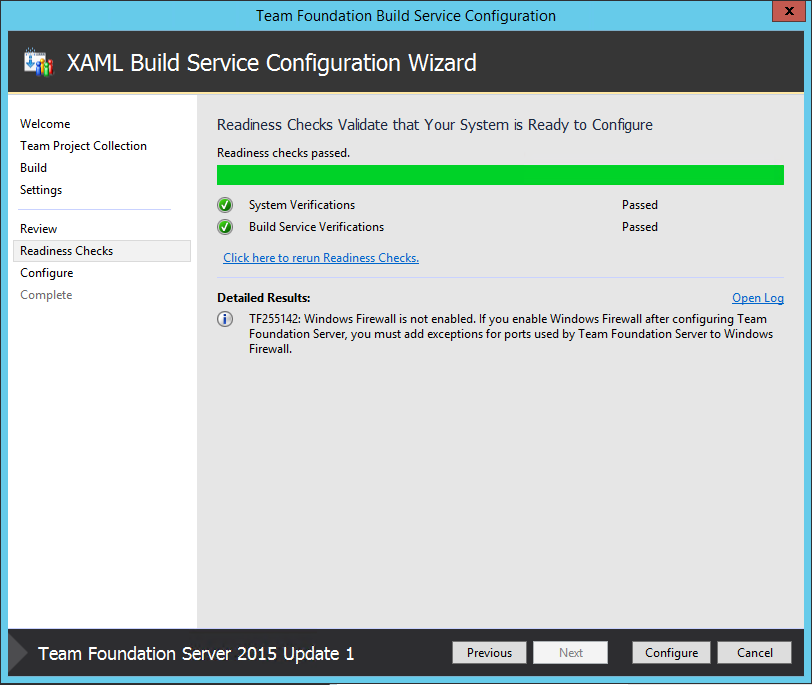
The recommended and the default number of agents is 4. Note that this will require a lot of RAM and therefore we will only choose 2 agents for Skånetrafikens CRM Solution.



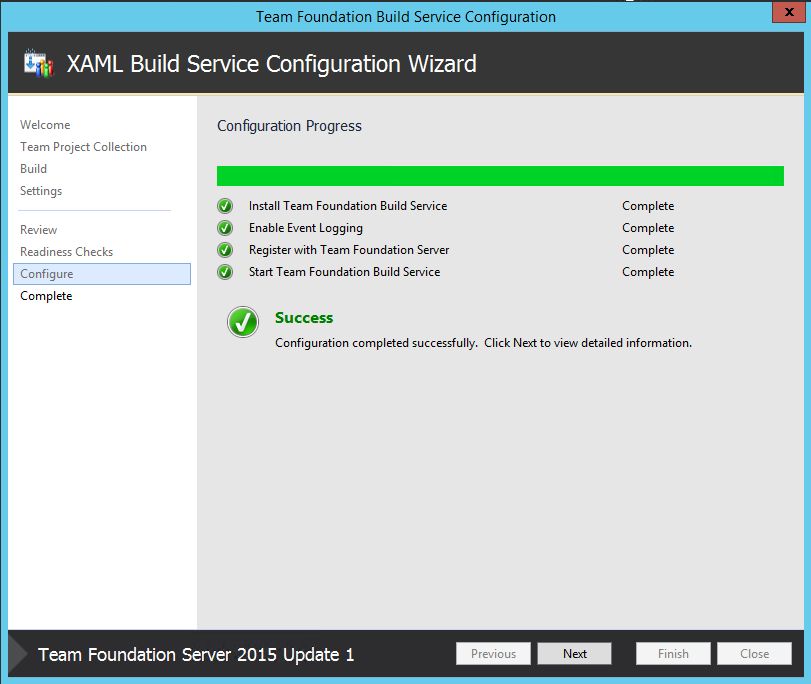
Note that the account that was used was “sa-se-Skanetrafiken” with password: “fvw6sUx2tAYkc2Zf”



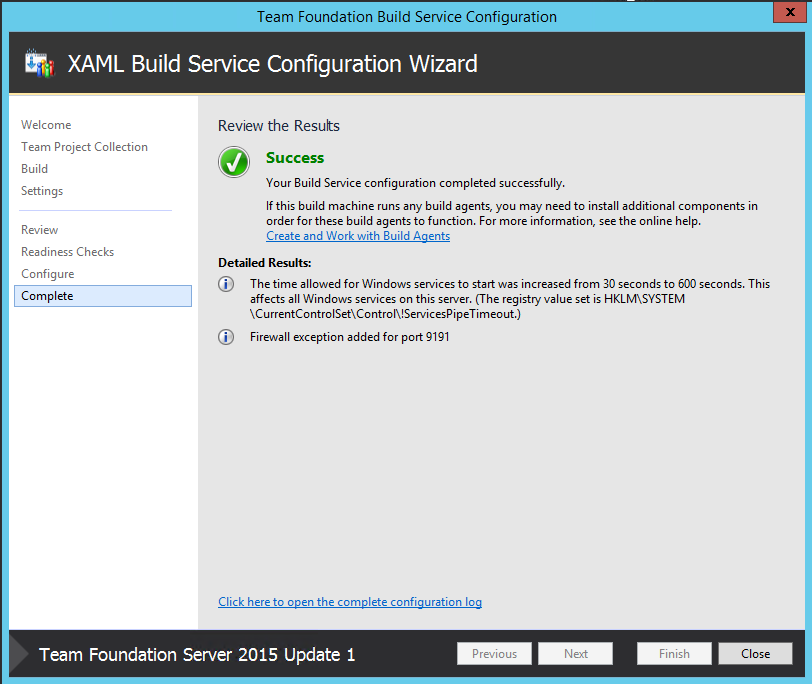
Ensure that all configuration settings are correct



Finalize the configuration by pressing ”Configure”



Press ”Next” to complete the installation

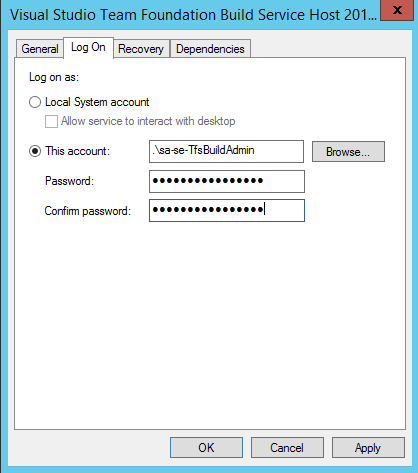


### When the Build Server doesn’t belong to the same domain as the TFS server

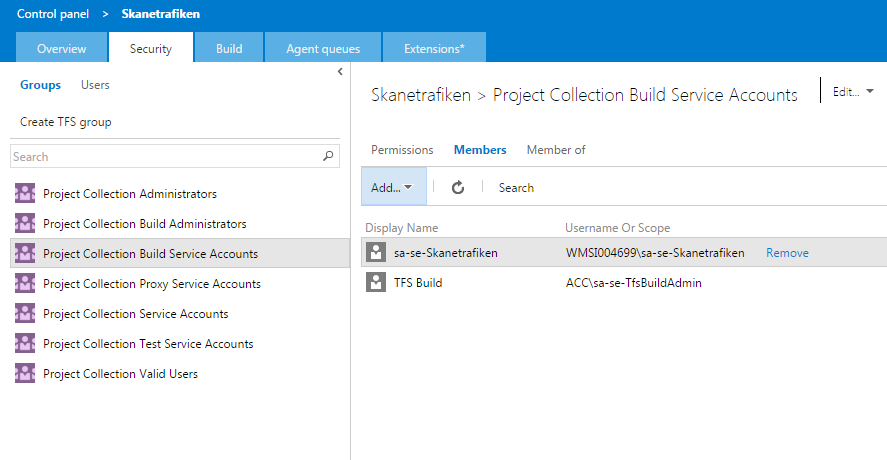
* Create local service user accounts with the same credentials on both the TFS and the Build servers.
* Open Windows services console (start > run > **services.msc**) and manually change the "Log On As" account for the "Visual Studio TFS Build Service Host 2013"



Same account must exist on both the Team Foundation Server and the TFS Build server.



Note that the account that was used was “sa-se-Skanetrafiken” with password: “fvw6sUx2tAYkc2Zf”



# Pre-work

Before applying CI and CD, the code was restructured into regions, commented code was removed, multiple classes in the same file was split into multiple files and ReSharper was used to get more structured and organized code.

# Architecture

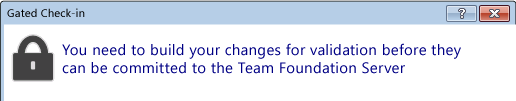
For more information about the ALM Architecture read the document “ALM for Skånetrafiken”.

## Process

The way process of which continuous integration & continuous delivery will be implemented in Skånetrafikens solution can be seen marked as green (CI) and yellow (CD) in the image below. The blue color however indicates the client side of this operation.



* 1. A developer makes modifications in the current solution project.
  2. By committing the changes to TFS the developer automatically triggers the build server to queue up a new build containing the modifications.
     + The functionality which triggers an event when code is committed is called gated check-ins and can be read in more detail in the links below).



<https://msdn.microsoft.com/en-us/library/vs/alm/build/define/triggers>

<https://msdn.microsoft.com/en-us/library/vs/alm/build/define/create>

* + - This is possible through the dedicated build agents configured on the build server whose purpose is handling build request sent by users.
  1. The developer is notified through email if the build was unsuccessful containing an error message.
  2. If the build however was successful, UnitTests are being run making sure that no computation logic errors have been made.
  3. If there are errors in the UnitTest this will be added to a detailed report.
  4. When the build is completed and all tests have been passed the solution will then be sent to

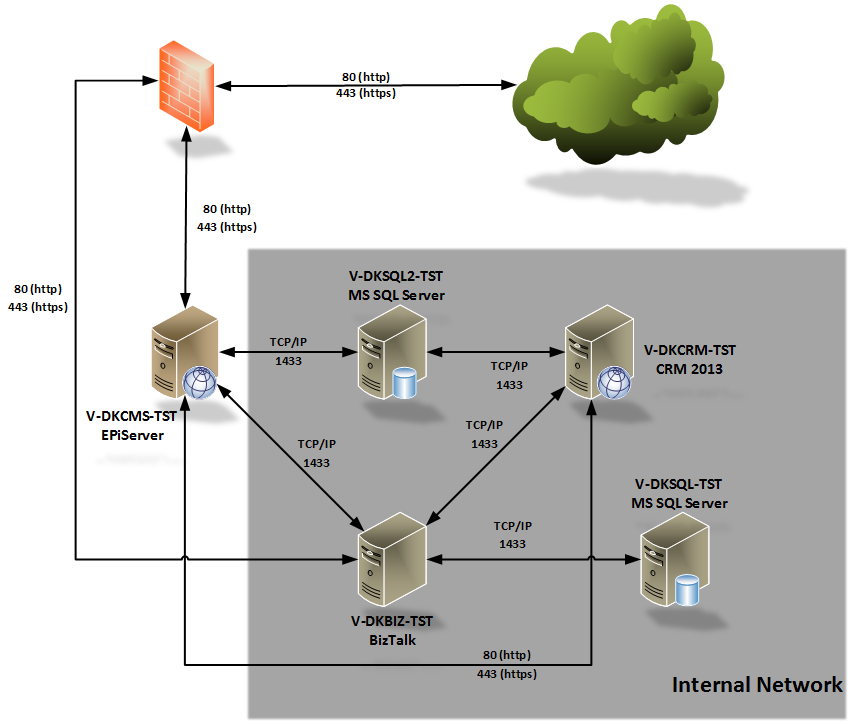
The image below shows the process of which a developer checks in new code that triggers an event for the build server to make sure that the code still compiles as usual. Note that this is a more detailed view of step 1 & 2 in the previous page.

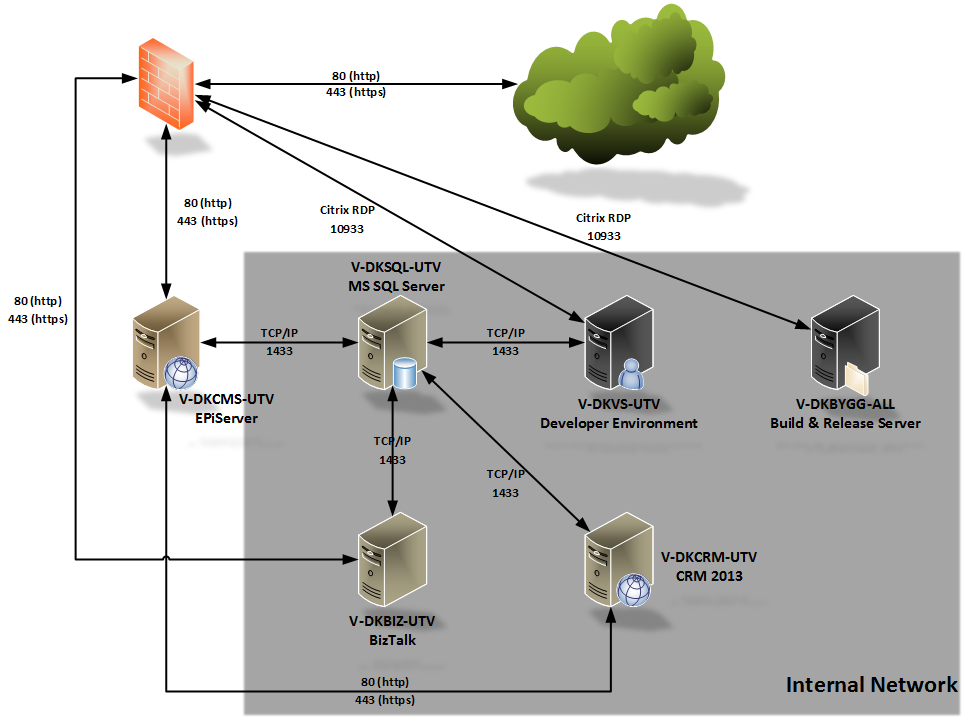
The image below shows how the Continuous Integration & Continuous Delivery process should work between all four environments.





The image below shows the schema of the TEST environment.





Visio-diagram och förklaringstext över leveransförfaranden mellan TFS-, Bygg- och olika miljöservers ur CRM-perspektivet. Detta kommer att förändras I och med att en ny byggserver kommer på plats.

## Ports

* 8000
* 8080
* 9001
* 1000

# Xrm CI Framwork

* Gated check-ins

Hur implementerar vi in Xrm CI Framework med vår lösning. Dokumentation finns på sidan där framework laddas ned. Vi tar och kombinerar med skärmdumpar som involverar vår lösning

# Project Types

**Vilka projekttyper blandar vi in i CICD? Crm, WCF, SQL-skript, etc**

* CRM components such as plugins, workflows, SSRS reports, entities, Silverlight controls and web resources.
* WCF
* SQL-script
* PowerShell-script

# Config Types

En specifikation över vilka olika configurationstyper vi har för dem olika miljöerna och vad skiljer dem åt sig.

# Build Definitions CI

Vilka olika build definitions kommer vi ha. En för WCF:arna, En för gated check in, och en för CRM bygget.

The complete project will be split into four different components:

* CRM components such as plugins, workflows, SSRS reports, entities, Silverlight controls and web resources.
* WCF
* SQL-script
* PowerShell-script

# Automated Tests

**En specning över vilka delar kommer vi skriva unit-tester för.**

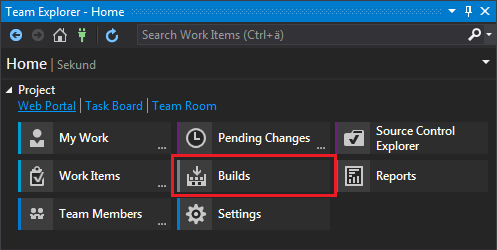
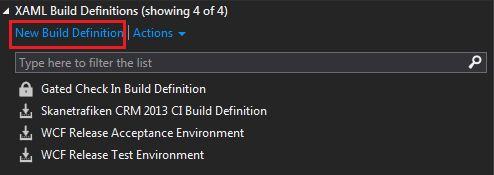
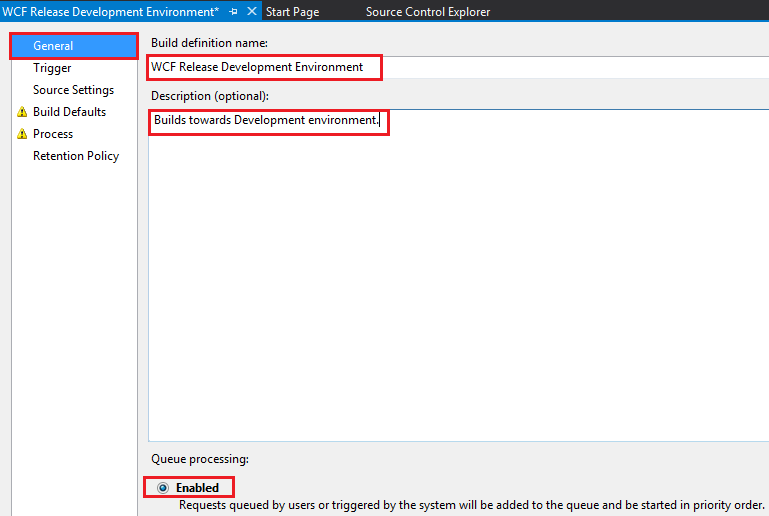
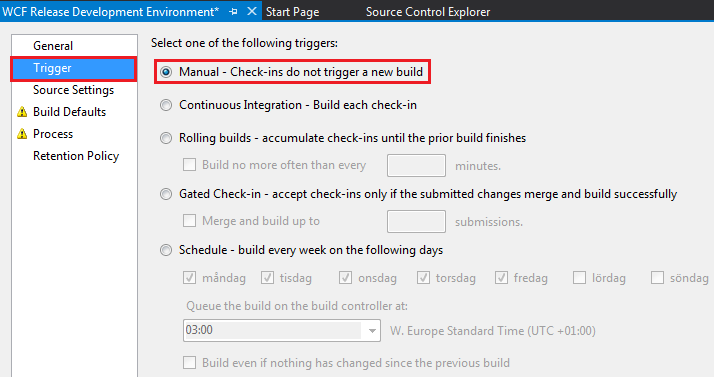
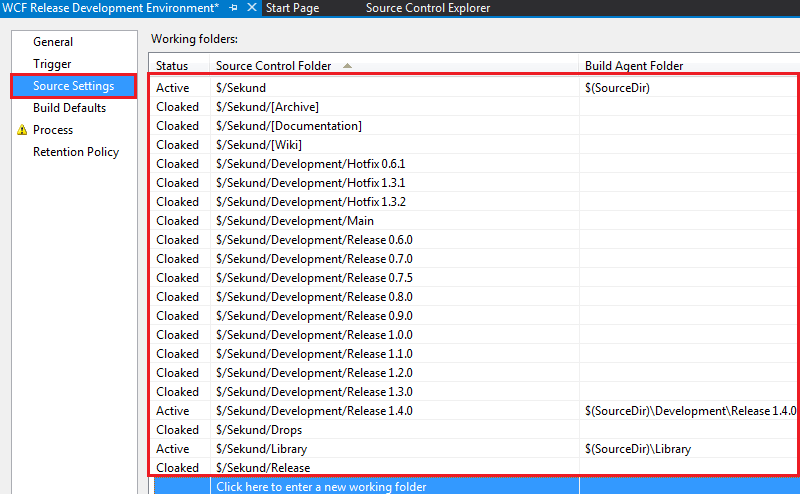
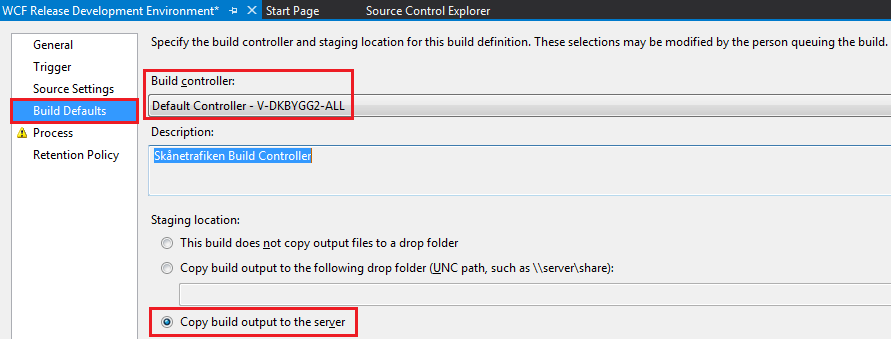
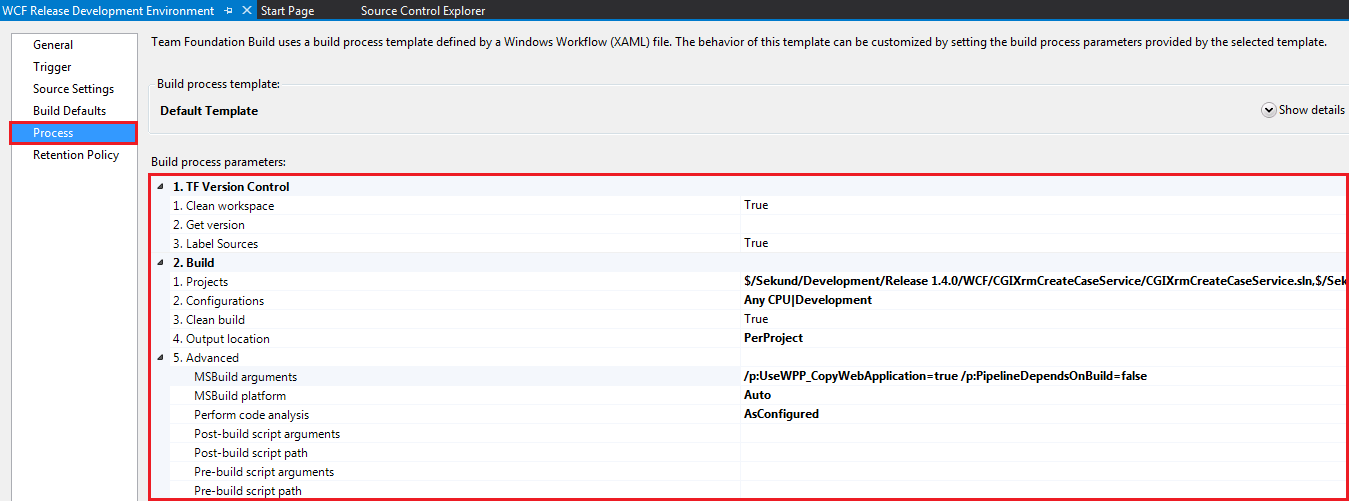
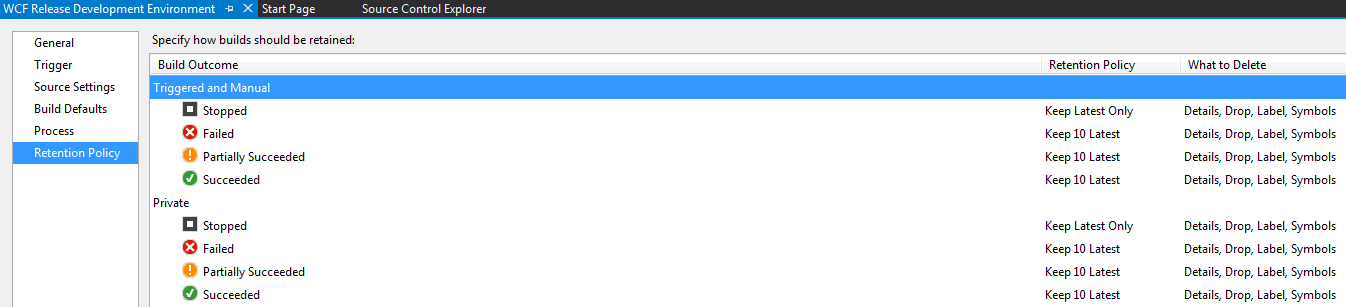
Automated tests are not required but highly recommended for Continuous Integration to work as expected. This is done to make sure that the code committed from the developers is not only compiling but also functioning correct. To achieve this, it is possible to run UnitTests on all operation which can be triggered after the build have succeeded. By doing this we can assure that the code is compiling and that we are retrieving expected results.

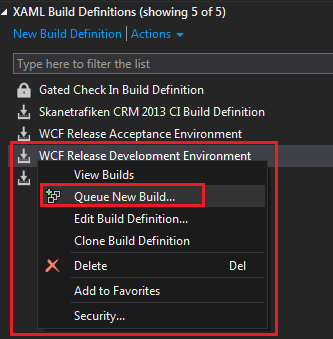
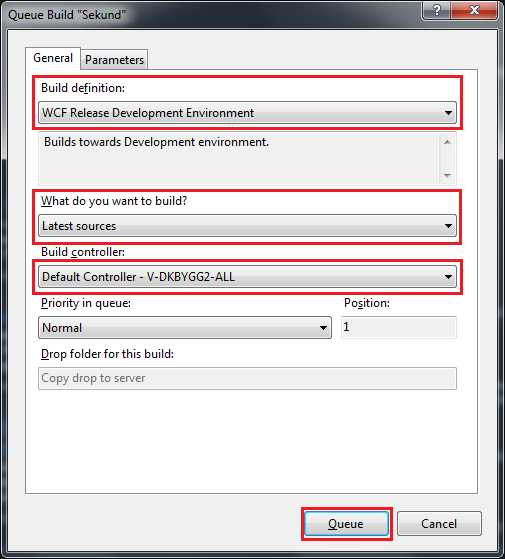
KOMMER ATT FYLLAS I ALLT EFTERSOM TESTERNA KOMMER PÅ PLATS

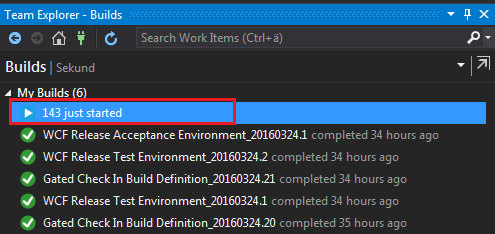
# Continuous Delivery

Beskrivning över vår lösning på continuous delivery. Tar vi in som sista biten i dokumentet.

# Creating & Setting up XAML Build Definition

1. Open Visual Studio
2. Enter Team Explorer and click on Builds button  
     
   
3. Press “New Build Definition” link button  
     
   
4. Enter the necessary information needed under the “General” session  
     
   
5. Under the “Trigger” session choose “Manual” option if this build definition is meant for releases.  
     
   
6. Enter the source control folders from TFVC and cloak unnecessary folders or branches that will not be included in the build definition. You can open a pre-existing build definition, target all and copy/paste this listing to save time. In this example below we’re creating a build from version 1.4.0 branch.  
     
   
7. In the “Build Defaults” session we’re going to define which Build Controller (TFS Build Server Controller) will work with our build definition and also where the deliverables will be dropped. In our case we choose “Copy build output to the server” which means that the deliverables will be dropped under C:\drops on the build server.  
     
   For gated check ins you might want to skip a drop folder and choose the first option in the “staging location”. Gated check ins are meant to just verify if your code changes are compatible with the code version in the repository.  
     
   
8. In the “Process” session edit the following settings:  
     
     
   * **Projects** (we add a list of solutions to be compiled from the 1.4.0 branch in this example)
   * **Configurations** (We use Config-transformation, which means that we defined our own web.config output based on which environment we’re working with)
   * **Output location** (We want each WCF-project to get its own folder in the drop folder structure)
   * **MSBuild arguments** (We add that line to force a config transformation based on the configurations setting previously)
9. In the “Retention Policy” session we are defining how many builds we keep before we start deleting finished builds, depending on the build outcome. In this case we let the default values remain. This should be monitored depending on the disk size available on the build server/destination and how often delivery/development cycles are occurring through the ALM.  
     
   

Now save your build definition and try running it once to see the results  
  
  
  


Double-click your newly queued build to see the status of your on-going build.  
  


Following you’ll retrieve a status report of your build. In the example below we haven’t added any unit tests to be included *(by the time this guide was written the TDD was an on-going process)*.  
  
It is outmost important that in Continuous Integration/Continuous Delivery processes include automated tests through continuous TDD (*Test Driven Development*) as these would be your first quality assurance tests prior deliverance/automated builds towards your environments.

