|  |
| --- |
|  |
| Jenkins CICD for .Net Project with example |
| Agile DevOps Lab |
| Author: Vivek Gole  Created On: 13-May-2015 |

Contents

[Overview 5](#_Toc419808638)

[Plan & Requirements 6](#_Toc419808639)

[Prerequisites 6](#_Toc419808640)

[Aim 6](#_Toc419808641)

[Objective 7](#_Toc419808642)

[Git 8](#_Toc419808643)

[Installation 8](#_Toc419808644)

[GitBlit 8](#_Toc419808645)

[Installation 8](#_Toc419808646)

[Configuration 8](#_Toc419808647)

[Directory Storage 8](#_Toc419808648)

[Properties 8](#_Toc419808649)

[Create Certificate 9](#_Toc419808650)

[Configure Gitblit 11](#_Toc419808651)

[Install Windows Service 11](#_Toc419808652)

[Browse Gitblit 11](#_Toc419808653)

[Repositories 12](#_Toc419808654)

[Nunit 14](#_Toc419808655)

[Installation 14](#_Toc419808656)

[FxCop 14](#_Toc419808657)

[Installation 14](#_Toc419808658)

[Jenkins 15](#_Toc419808659)

[Installation 15](#_Toc419808660)

[Plugins 15](#_Toc419808661)

[Configurations 15](#_Toc419808662)

[Directory Storage 15](#_Toc419808663)

[Git 16](#_Toc419808664)

[MsBuild 16](#_Toc419808665)

[FxCop 17](#_Toc419808666)

[Email Notification 18](#_Toc419808667)

[Jenkins Project: HealthManagement 19](#_Toc419808668)

[Configurations 19](#_Toc419808669)

[Prerequisites 19](#_Toc419808670)

[Project Name 20](#_Toc419808671)

[Source Code Management- Git 20](#_Toc419808672)

[Build Triggers 21](#_Toc419808673)

[Build – MS Build 21](#_Toc419808674)

[Build – Nunit 21](#_Toc419808675)

[Build – FxCop 22](#_Toc419808676)

[Post-Build Actions – Publish Nunit result 22](#_Toc419808677)

[Post-Build Actions – FxCop result 23](#_Toc419808678)

[Post-Build Actions – Call Deployment Jenkin Build 23](#_Toc419808679)

[Post-Build Actions – Send email notifications 23](#_Toc419808680)

[Reports 24](#_Toc419808681)

[Project Health 24](#_Toc419808682)

[Test Results Analyzer 24](#_Toc419808683)

[Violation 27](#_Toc419808684)

[Jenkins Project: HealthManagementDeployQA 28](#_Toc419808685)

[Configurations 28](#_Toc419808686)

[Project Name 28](#_Toc419808687)

[Source Code Management- None 28](#_Toc419808688)

[Build – Clean Previous Publish from Jenkins 28](#_Toc419808689)

[Build – MSBuild create publish code 29](#_Toc419808690)

[Build – Archive previous deployment 29](#_Toc419808691)

[Build – Deploy 30](#_Toc419808692)

[Post-Build Actions – Run selenium tests 30](#_Toc419808693)

[Post-Build Actions – Send email notifications 31](#_Toc419808694)

[Reports 32](#_Toc419808695)

[Project Health 32](#_Toc419808696)

[Team Foundation Server with GIT 33](#_Toc419808697)

[Prerequisites 33](#_Toc419808698)

[TFS Integration with GIT 33](#_Toc419808699)

[Global Git Setting 33](#_Toc419808700)

[Clone Git Repository 34](#_Toc419808701)

[Managing Repository from TFS 35](#_Toc419808702)

[Appendix 41](#_Toc419808703)

[Scripts 41](#_Toc419808704)

[Nunit PowerShell Script 41](#_Toc419808705)

[References 41](#_Toc419808706)

[Study Materials 41](#_Toc419808707)

[Downloads 41](#_Toc419808708)

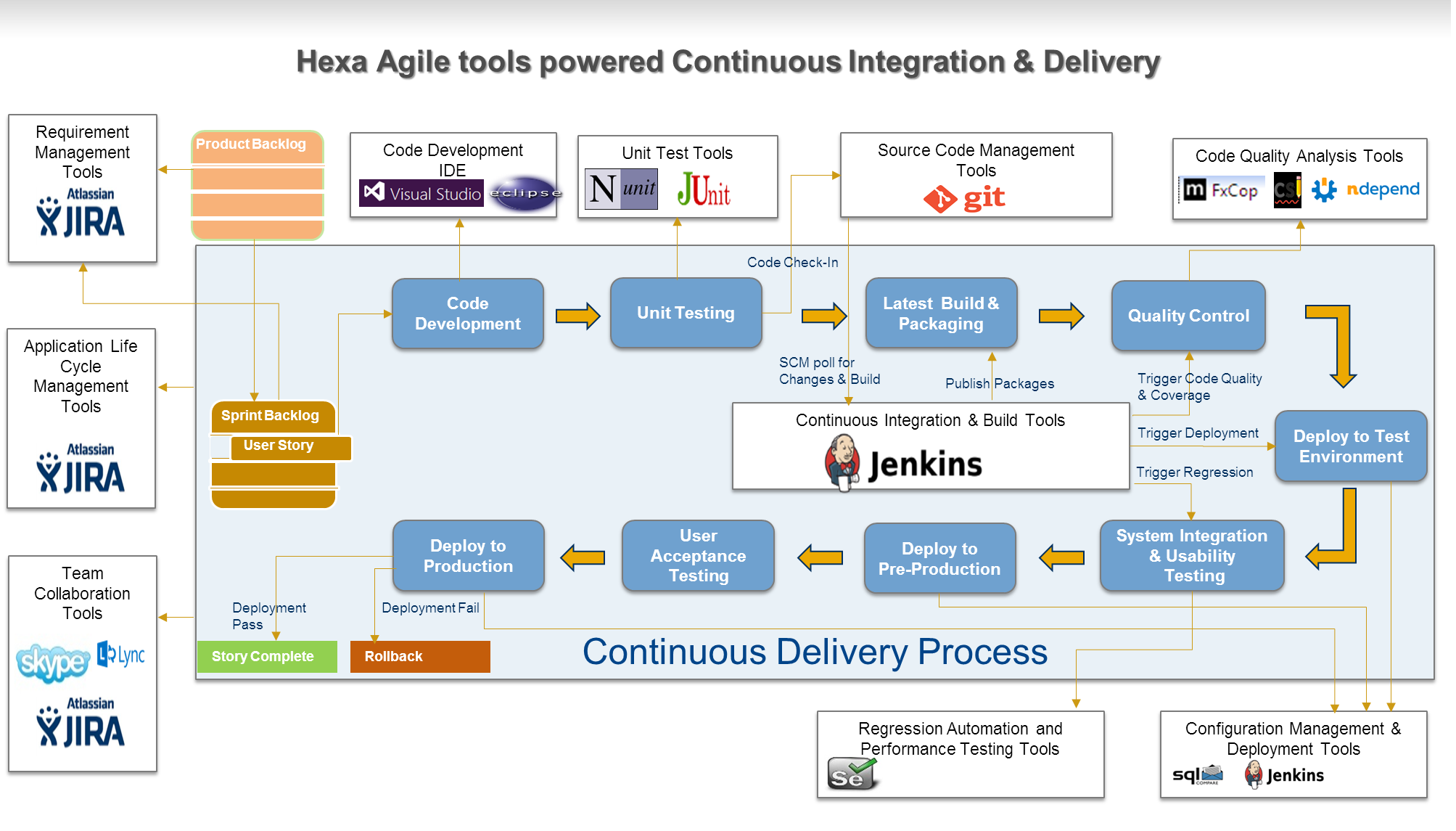
[Example 41](#_Toc419808709)

[Application Links 41](#_Toc419808710)

# Overview

This document highlights important key tools used to setup Agile DevOps lab. Main focus of this document is to show how Continuous Integration and Continuous deployment is configured using various tools.

The chart below gives a complete overview of Hexaware Agile DevOps practices.



# Plan & Requirements

This topic summarizes all prerequisites tools used and how these tools are integrated to achieve our aim, which is Continuous Integration and Continuous deployment.

## Prerequisites

Following are the software used to set up Agile DevOps lab for a sample .Net application.

Git, GitBlit, Jenkins, Nunit, FxCop are the freeware tools which are highly efficient, scalable and extensible tools.

* **Visual Studio 2013** – IDE to create a sample .Net application.
* **SQL Server 2012** - Database
* **Git**- Tool to create git repositories.
* **GitBlit**- Git server tool to manage git repositories.
* **Jenkins**- Continuous Integration and Continuous Delivery tool.
* **Nunit**- Application Unit testing tool.
* **FxCop**- Application Code analysis tool.
* **Selenium**- Automation tool.
* **Powershell**- Scripting tool.
* **JIRA**- Tracking tool.

## Aim

The specific goals of a DevOps approach span the entire delivery pipeline. They include improved deployment frequency, which can lead to faster time to market, lower failure rate of new releases, shortened lead time between fixes, and faster mean time to recovery in the event of a new release crashing or otherwise disabling the current system. Simple processes become increasingly programmable and dynamic, using a DevOps approach, which aims to maximize the predictability, efficiency, security, and maintainability of operational processes. Very often, automation supports this objective.

DevOps integration targets product delivery, quality testing, feature development, and maintenance releases in order to improve reliability and security and provide faster development and deployment cycles. Many of the ideas (and people) involved in DevOps came from the enterprise systems management and Agile software development movements.

DevOps aids in software application release management for an organization by standardizing development environments. Events can be more easily tracked as well as resolving documented process control and granular reporting issues. Companies with release/deployment automation problems usually have existing automation but want to more flexibly manage and drive this automation — without needing to enter everything manually at the command-line. Ideally, this automation can be invoked by non-operations employees in specific non-production environments. The DevOps approach grants developers more control of the environment, giving infrastructure more application-centric understanding.

## Objective

Below flow chart shows what we are trying to achieve with the sample .Net application. For every change to Git Repository- Jenkins will validate the code by running MS build and other tools. After all the build steps are passed, Jenkins will deploy the code changes to targeted deployment server and run automated selenium test cases to validate the build.

Git Repository

Jenkins

MS Build

Nunit Testing

FxCop

Call Deployment Jenkin Project

Send Mail to Configured Recipients

JIRA

Developer

Developer Check In changes to **Git** Repository

Jenkin service pulls changes from **Git** Repository

If MS Build failed-

Fail the Jenkins build

If MS Build Pass- Execute Nunit testing

If Nunit Script failed-

Fail the Jenkins build

If any Nunit test case fails- Log a ticket in JIRA

If Nunit Script Pass-

Execute FxCop

If Fxcop pass

If FxCop failed-

Fail the Jenkins build

QA – Deployment Server

Step1- Deploy changes to Targeted server - QA

Step2- If Deploy is pass –

Run Automated Selenuim test

To validate recent deployed changes.

If deployment Jenkin Project failed-

Fail the Jenkin build

# Git

## Installation

Git tool is used to created/clone git repositories. This tool will be installed on Repository server, Jenkins server and all developer’s machines.

Download the windows version for Git from <http://www.git-scm.com/>

Choose default settings while installing the software

# GitBlit

## Installation

Gitblit is an open-source, pure Java stack for managing, viewing, and serving [Git](http://git-scm.com/) repositories. This tool will be installed on repository server.

Download the windows version for Gitblit from <http://www.gitblit.com/>

Follow the installation steps mentioned in <http://www.gitblit.com/setup_go.html>

It is recommended to use secured certificate while configuring gitblit. If you want to use normal httpport, change the value for key “Server.HttpPort” from “data/gitblit.properties” to any open port other than default 80.

## Configuration

We followed below steps to configure the gitblit. For more details on configuration refer <http://www.gitblit.com/setup_go.html>.

### Directory Storage

The location where Gitblit files are extracted (this is called baseFolder): *E:\GitBlit*

### Properties

Properties/Configuration settings are defined in file *E:\GitBlit\data\gitblit.properties*

If Gitblit is installed as windows service, then the service needs to be restarted after any changes to the properties file.

Following are the properties modified with modified values.

1. server.httpPort = 88
2. server.certificateAlias = cch1wdbas2.corp.hexaware.com
3. web.siteName = cch1wdbas2.corp.hexaware.com
4. server.storePassword = gitblit

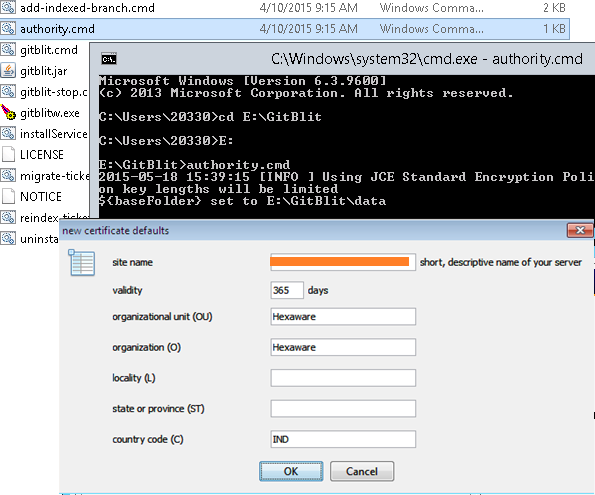
### Create Certificate

Gitblit provides a batch file inside baseFolder *authority.cmd* which can be used to create SSL certificate.

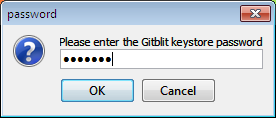
The Authority is a Swing GUI application. Use of this tool is not required as Gitblit GO will startup and create SSL certificates itself, BUT use of this tool allows you to control the identification metadata used in the generated certificates. Skipping this step will result in certificates with default metadata.

1. From baseFolder execute *authority.cmd* or *java -cp gitblit.jar com.gitblit.authority.Launcher --baseFolder data* from a command-line
2. Enter the details:

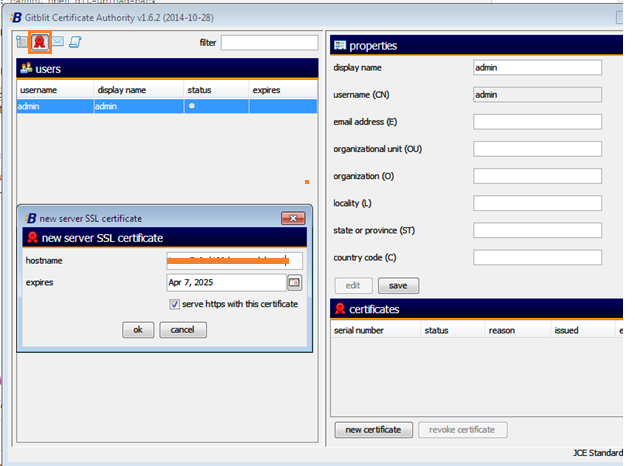
Name of website: *cch1wdbas2.corp.hexaware.com*



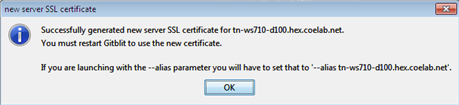
1. Enter password from gitblit.properties- property *server.storePassword = gitblit*



1. Create new server certificate- click icon in orange box.
   1. Enter host name - *cch1wdbas2.corp.hexaware.com*
   2. *Enter expiry date*
   3. *Enter other details.*



1. Save the details and this will create new certificate.



### Configure Gitblit

To install the gitblit configurations from the baseFolder execute *gitblit.cmd* or *java -jar gitblit.jar --baseFolder data* from a command-line.

### Install Windows Service

Gitblit provides two batch files to install and un-install gitblit service from windows services.

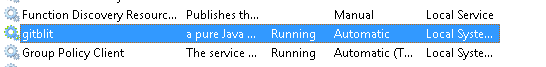
* installService.cmd – To install as windows service
* uninstallService.cmd – To uninstall from windows service

Before executing these services make sure before setting are proper as per server machine architecture.

Edit both files in notepad (notepad++), and update the ARCH property as per server architecture.

In our case it was updated as “*SET ARCH=x86*”

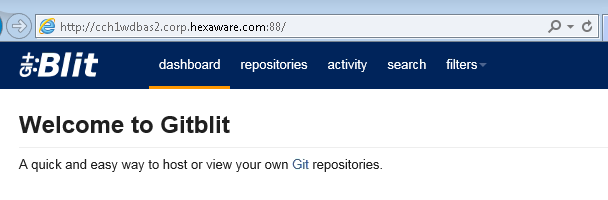
Check windows services and start gitblit service in automatic mode.



### Browse Gitblit

Access the gitblit from browser with provided website/hostname.

If you are accessing the site without SSL then enter the server.httpPort provided in gitblit.properties.



### Repositories

Repositories tab shows all the repositories from GitBlit.

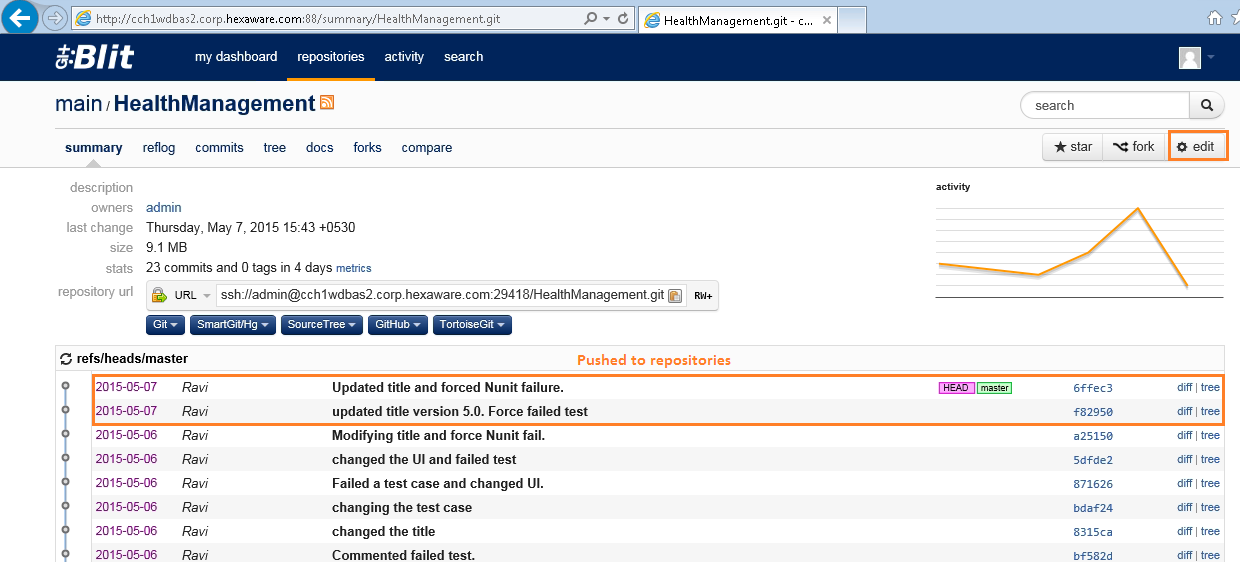
Click on *Repositories tab>> new repository* to create new repository.

1. Enter the name for repository- This will be used to access the repository.
2. Select the appropriate access policy.
3. Keep the other settings as default and click create button to create new repository.

To modify the repository, select the repository from repositories tab.

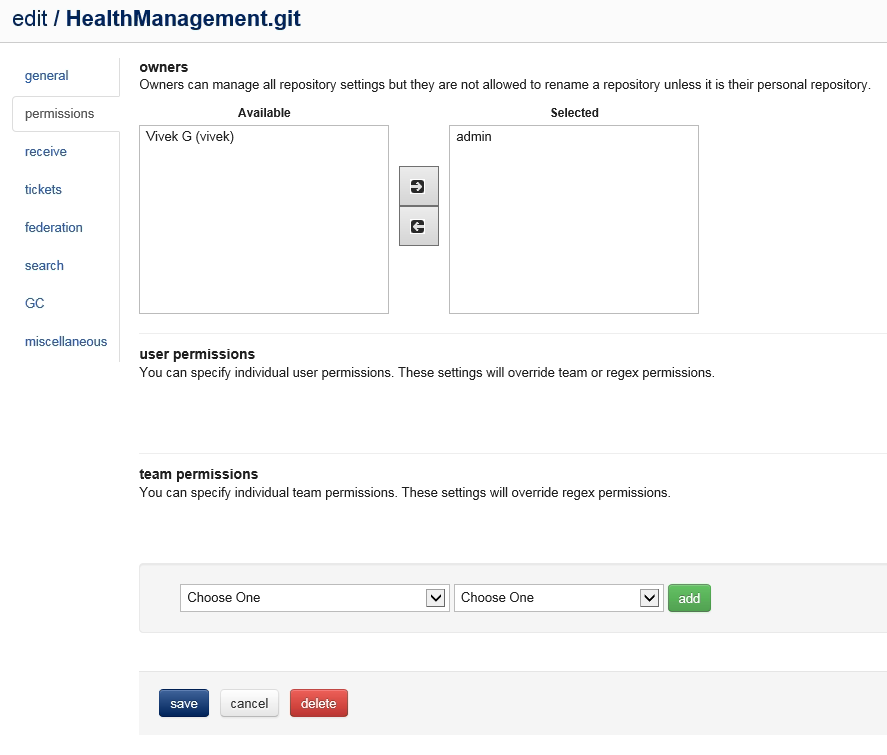
Below image shows dashboard for HealthManagement repository. It list all the changes pushed to the repository.

Highlighted in orange box shows the changes pushed to repository. To see more details about the push changes, click on the comment.



To edit the repository setting click Edit *(Highlighted in orange box in above image)*.

Here you can update the repository name, manage permissions for users and teams, create pre and post receive hooks.

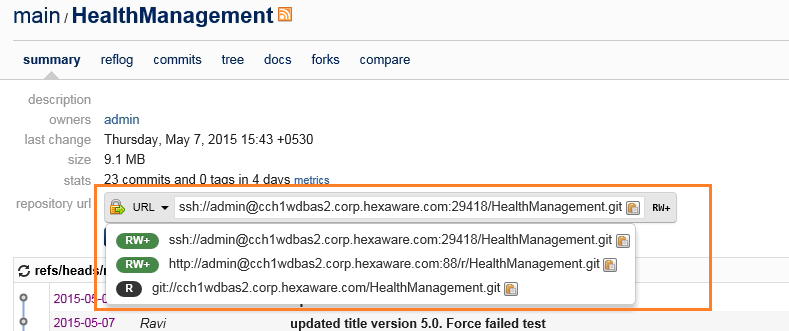


Repository URLs can be accessed with SSL or without SSL with normal httpport.

URLs are of 2 types,

1. RW- URL with read/write permissions.
2. R- URL with only read permissions.

Use URL with RW if you want to push changes to repository. USE URL with only read right if when application only needs to access the repository changes.



# Nunit

## Installation

Nunit tool is used to create unit test to validate application code. This tool will be installed on Jenkins server and all developer’s machines.

Download the windows version for Nunit from <http://nunit.org/index.php?p=download>

# FxCop

## Installation

FxCop tool is used as a code analysis and quality check tool. This tool is an integrated part of Visual Studio 2013; hence all .Net developer’s machine will have this preinstalled. This tool will be installed on Jenkins server.

Download FxCop from <http://www.microsoft.com/en-gb/download/details.aspx?id=6544>

# Jenkins

## Installation

Jenkins CI is the leading open-source continuous integration server. Built with Java, it provides **1009** plugins to support building and testing virtually any project.

Download stable windows version of Jenkins from <http://mirrors.jenkins-ci.org/windows/latest>

Note: Install Jenkins as windows service and make sure to run this windows service with admin user privileges. This can be done by using admin user account for Log On property of Jenkins service.

## Plugins

Jenkins plugin manager provides vast range of plugins which can be used as per requirement for configuration.

To install missing plugins refer plugins from Manage Jenkins>> Manage Plugin >>Available (Tab)

Following are the plugins installed for this demo.

* [Credentials Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Credentials+Plugin)
* [GIT plugin](http://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin)
* [MSBuild Plugin](http://wiki.jenkins-ci.org/display/JENKINS/MSBuild+Plugin)
* [NUnit plugin](http://wiki.jenkins-ci.org/display/JENKINS/NUnit+Plugin)
* [FxCop Runner plugin](https://wiki.jenkins-ci.org/display/JENKINS/FxCop+Runner+Plugin)
* [Project Health Report](https://bitbucket.org/henriklynggaard/project-health-report-plugin/wiki/Home)
* [Test Results Analyzer Plugin](http://wiki.jenkins-ci.org/display/JENKINS/Test+Results+Analyzer+Plugin)
* [Violations plugin](http://cch1wdbas1.corp.hexaware.com:8080/pluginManager/wiki.jenkins-ci.org/display/JENKINS/Violations)

## Configurations

To configure global settings for all the builds, refer link Manage Jenkins>>Configure System.

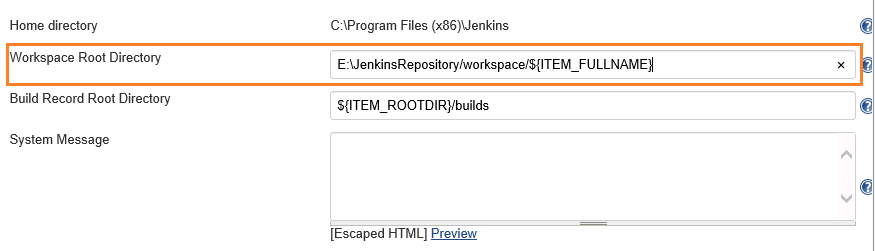
Below we are highlighting all settings used to configure this demo.

### Directory Storage

**Home directory**- This is the place where Jenkins is installed.

**Workspace Root Directory**- Since the size of Jenkin project will keep on increasing, it better to change the Workspace Root Directory location to data disk drive.

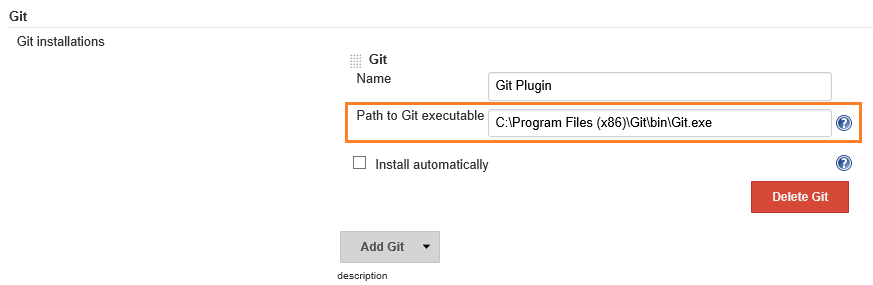
**Build Record Root Directory**- This directory location can also be changed as per needs. By default Jenkins project build configurations are saved in project build directory under Home directory.



### Git

A configuration for git is displayed after installing git plugin.

Pre-requisites: Git client (to clone repositories), Git plugin.

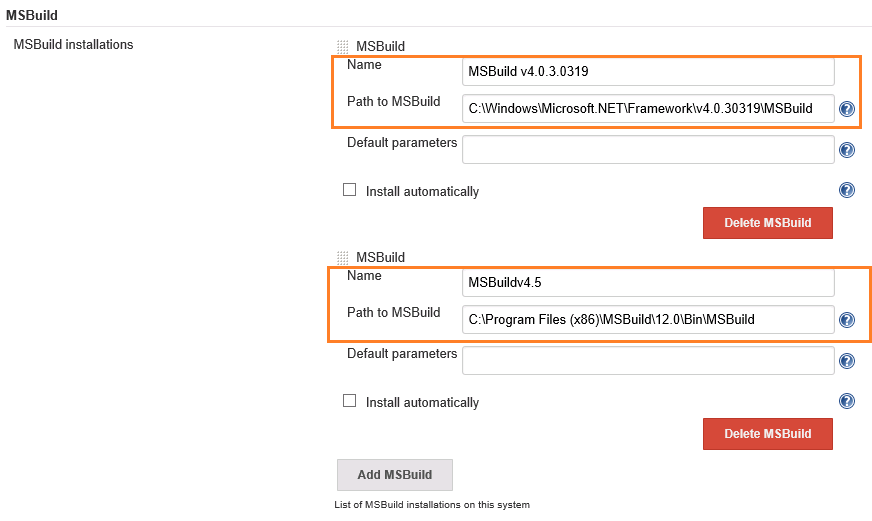


### MsBuild

A configuration for Msbuild is displayed after installing Msbuild plugin.

Pre-requisites: Framework referred for Msbuild should be installed on server and Msbuild plugin.

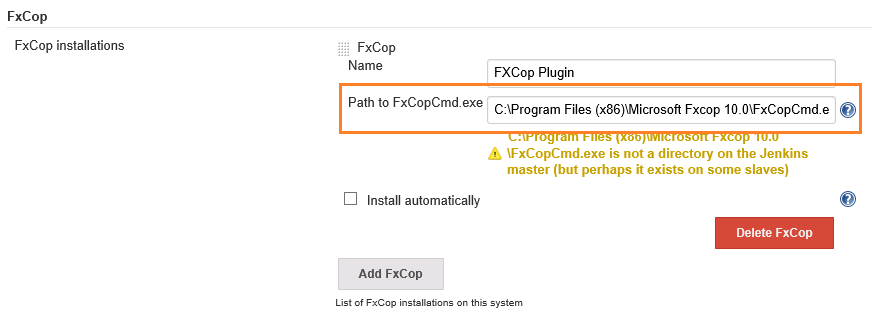
Here we have configured 2 different MSbuilds to compile and build .Net applications. In most cases it is recommended to use C:\Program Files (x86)\MSBuild\12.0\Bin\Msbuild path.



### FxCop

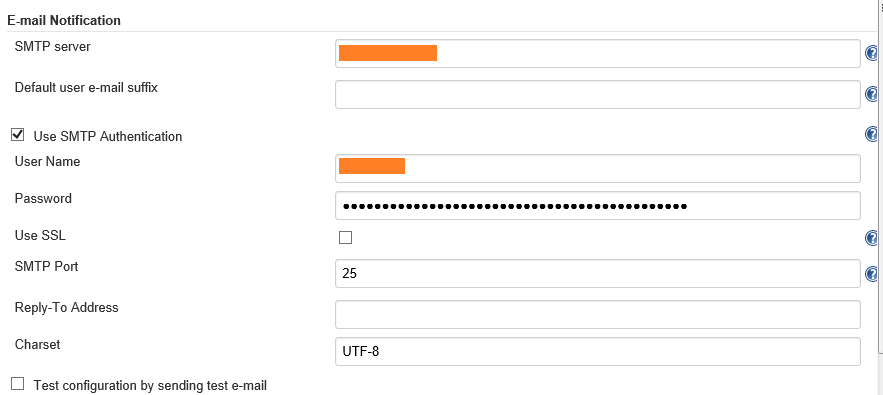
A configuration for FxCop is displayed after installing FxCop runner plugin.

Pre-requisites: FxCop should be installed on server and FxCop runner plugin.



### Email Notification

Configure this to enable mail send engine for sending alerts for failed or unstable Jenkins build.



# Jenkins Project: HealthManagement

This Jenkins freestyle project/build is created for continuous integration of a sample .Net application created in MVC4.5.

For every change in its source Git repository, this project will get the latest code, compile and test it using Nunit and FxCop. Any build failure will immediately be notified to concern developer and team to rectify the error.

## Configurations

Note: Path used in the windows command batch will be changed for any changes in existing referred folder paths.

### Prerequisites

To keeps all build artifacts organized; we are following specific folder structures to preserve the results/output files.

{**Workspace Root Directory**}\**JenkinBuild**  - This is the master folder to keep all artifacts for each Jenkins project/build.

{**Workspace Root Directory**}\**JenkinBuild\ MsBuildResults**- This folder stored Msbuild compile output files for each build. Files are named to have Jenkins build number. This helps in troubleshooting and identifying results for each build.

{**Workspace Root Directory**}\**JenkinBuild\ FxCopResults**  - This folder stored Fxcop output files for each build. Files are named to have Jenkins build number. This helps in troubleshooting and identifying results for each build.

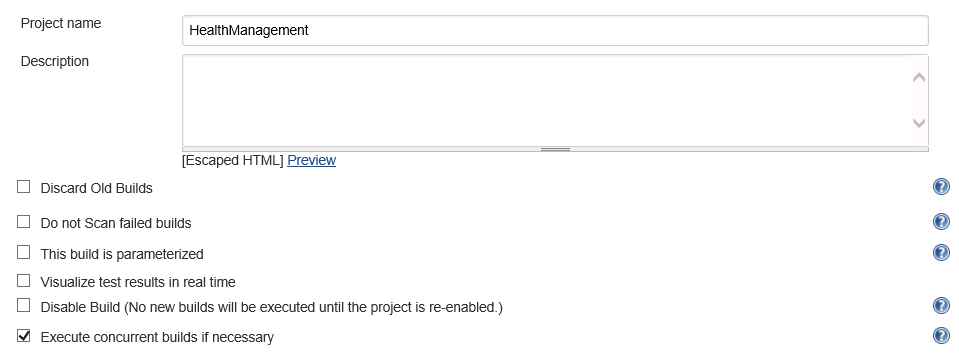
{**Workspace Root Directory**}\**JenkinBuild\ NunitResults**- This folder stored Nunit test result files for each build. Files are named to have Jenkins build number. This helps in troubleshooting and identifying results for each build.

{**Workspace Root Directory**}\**JenkinBuild\ PowershellScripts**- This folder stored all the PowerShell scripts used in Jenkins project. Run logs for PowerShell scripts

{**Workspace Root Directory**}\**JenkinBuild\ Logs**- This folder stored run logs for PowerShell scripts. This will be used for troubleshooting PowerShell scripts.

{**Workspace Root Directory**}\**JenkinBuild\ PublishCode**- This folder stored MS build published code, which will be deployed on targeted deployment server. This folder is used in Jenkins Project- HealthManagementDeployQA.

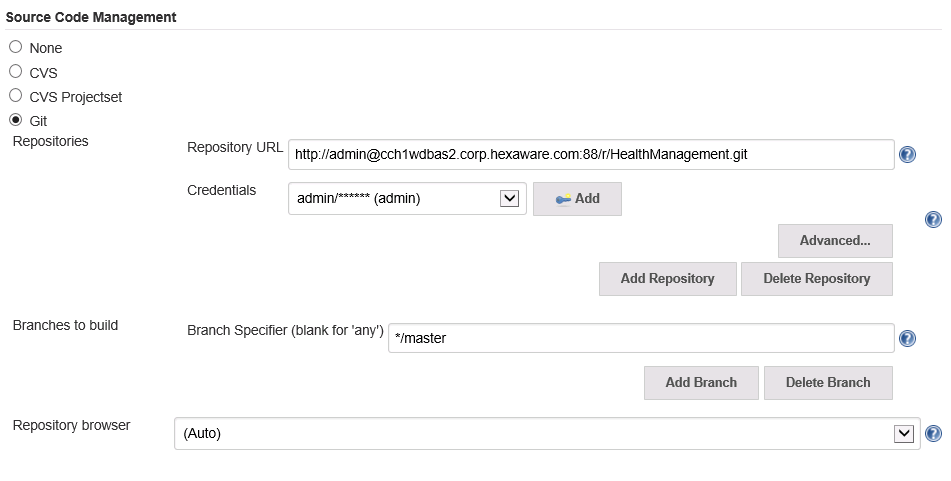
### Project Name



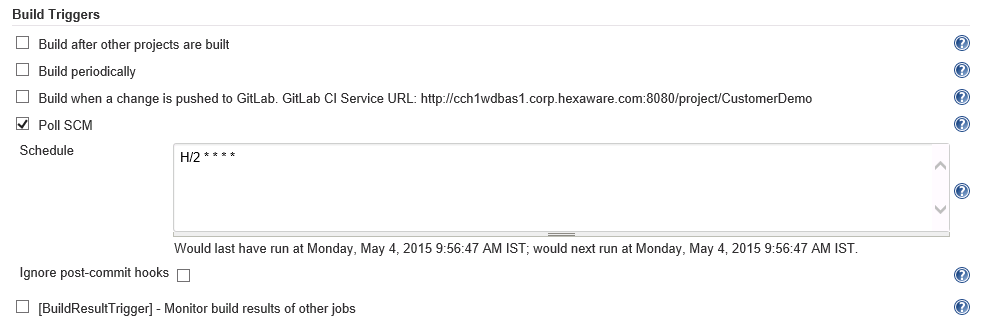
### Source Code Management- Git

URL: [*http://admin@cch1wdbas2.corp.hexaware.com:88/r/HealthManagement.git*](http://admin@cch1wdbas2.corp.hexaware.com:88/r/HealthManagement.git)

Credentials: *admin/admin*



### Build Triggers

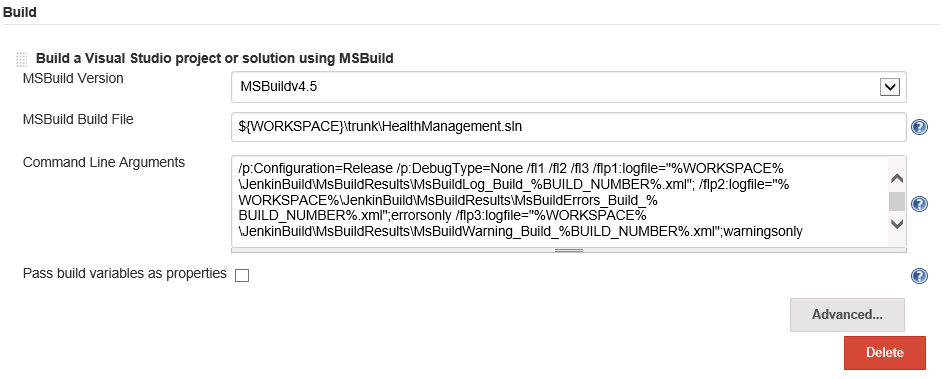


### Build – MS Build

MSBuild Build File: *${WORKSPACE}\trunk\HealthManagement.sln*

Command Line Arguments:

*/p:Configuration=Release /p:DebugType=None /fl1 /fl2 /fl3 /flp1:logfile="%WORKSPACE%\JenkinBuild\MsBuildResults\MsBuildLog\_Build\_%BUILD\_NUMBER%.xml"; /flp2:logfile="%WORKSPACE%\JenkinBuild\MsBuildResults\MsBuildErrors\_Build\_%BUILD\_NUMBER%.xml";errorsonly /flp3:logfile="%WORKSPACE%\JenkinBuild\MsBuildResults\MsBuildWarning\_Build\_%BUILD\_NUMBER%.xml";warningsonly*

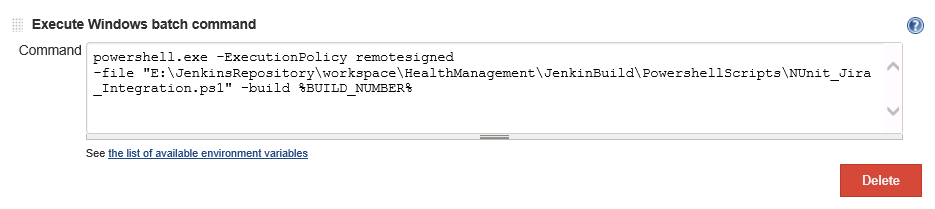


### Build – Nunit

Powershell script reads the settings from configuration file. It will compile the Nunit test case and will preserve the test result in given destination. It will log a bug in JIRA for failed test cases.

Source for Powershell script is mentioned in the Appendix section.

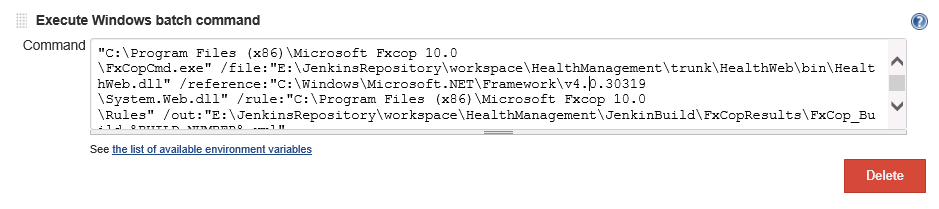
Command:

*powershell.exe -ExecutionPolicy remotesigned -file "E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\PowershellScripts\NUnit\_Jira\_Integration.ps1" -build %BUILD\_NUMBER%*

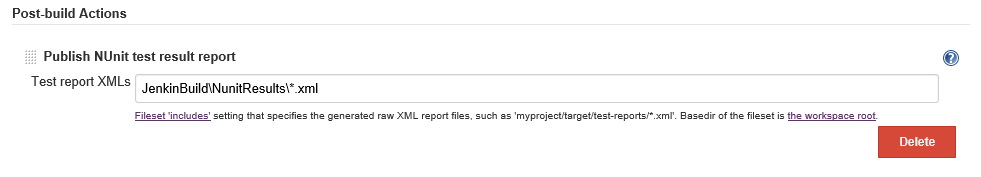
### Build – FxCop

Command:

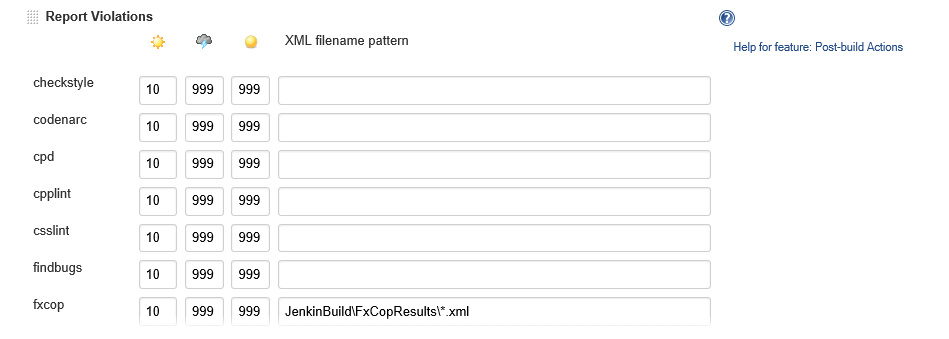
*"C:\Program Files (x86)\Microsoft Fxcop 10.0\FxCopCmd.exe" /file:"E:\JenkinsRepository\workspace\HealthManagement\trunk\HealthWeb\bin\HealthWeb.dll" /reference:"C:\Windows\Microsoft.NET\Framework\v4.0.30319\System.Web.dll" /rule:"C:\Program Files (x86)\Microsoft Fxcop 10.0\Rules" /out:"E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\FxCopResults\FxCop\_Build\_%BUILD\_NUMBER%.xml"*



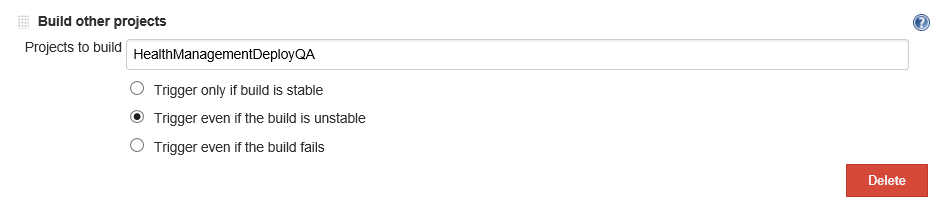
### Post-Build Actions – Publish Nunit result



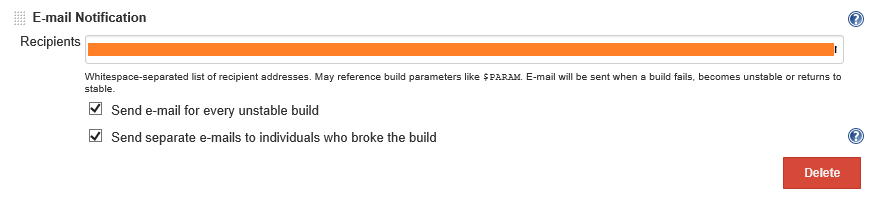
### Post-Build Actions – FxCop result



### Post-Build Actions – Call Deployment Jenkin Build



### Post-Build Actions – Send email notifications



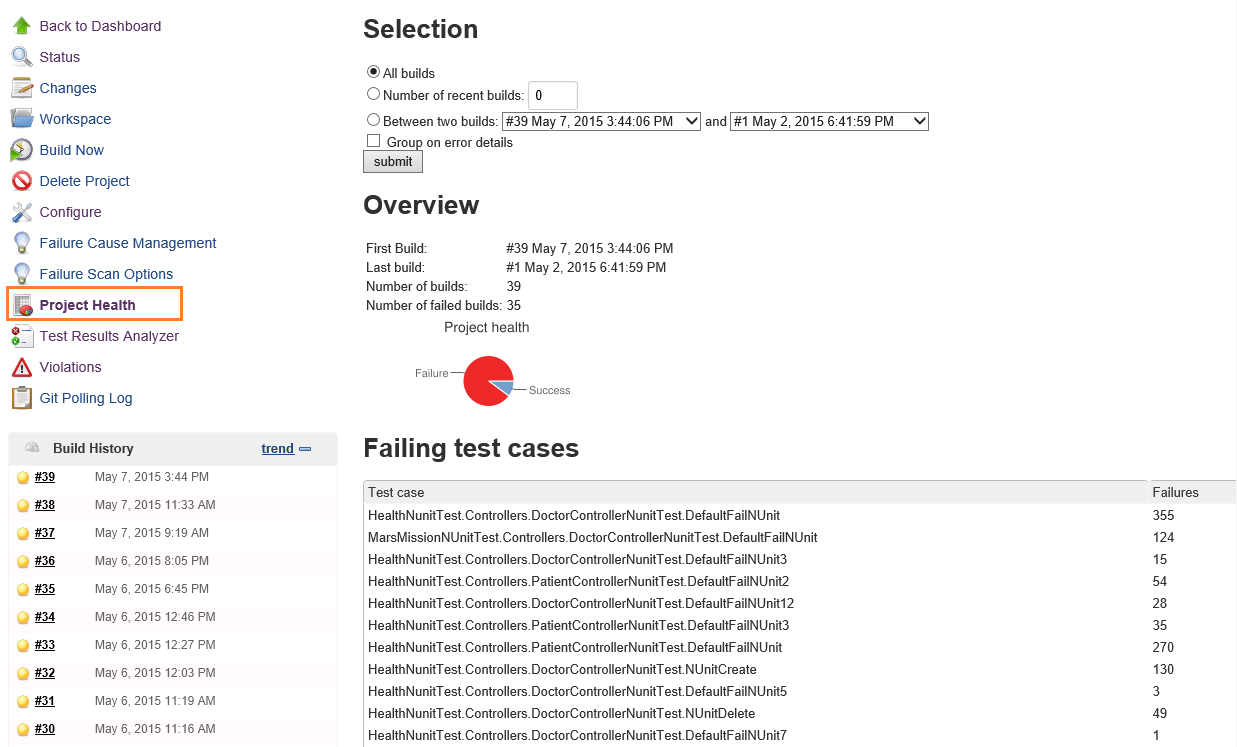
## Reports

Open the dashboard for HealthManagement Jenkins project to see all the reporting options. By default it will show Test result trends and FxCop trends.

### Project Health

This report is displayed after installing Project Health Plugin.

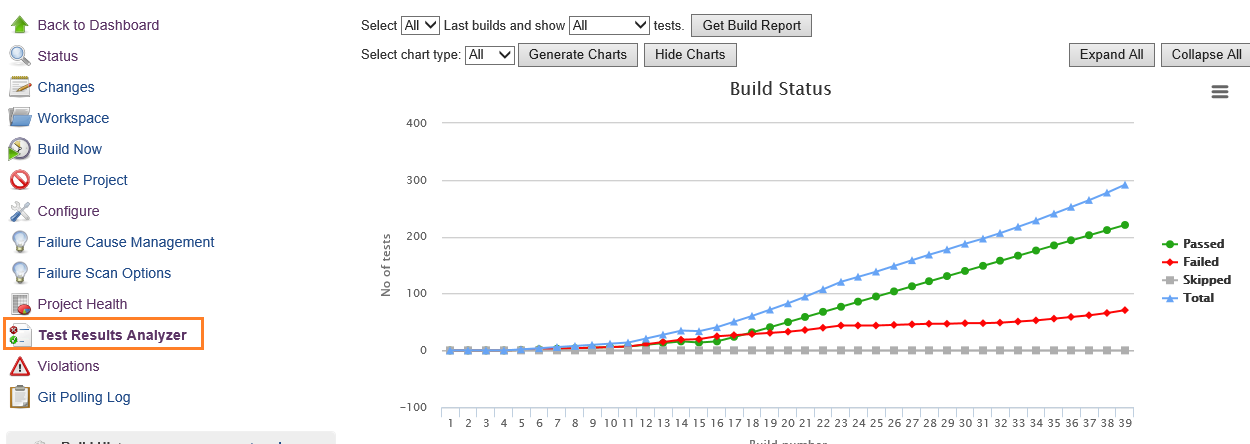
Project Health dashboard shows the overview of builds as per selection in graphical chart. It also shows summary of all failed Nunit test cases.

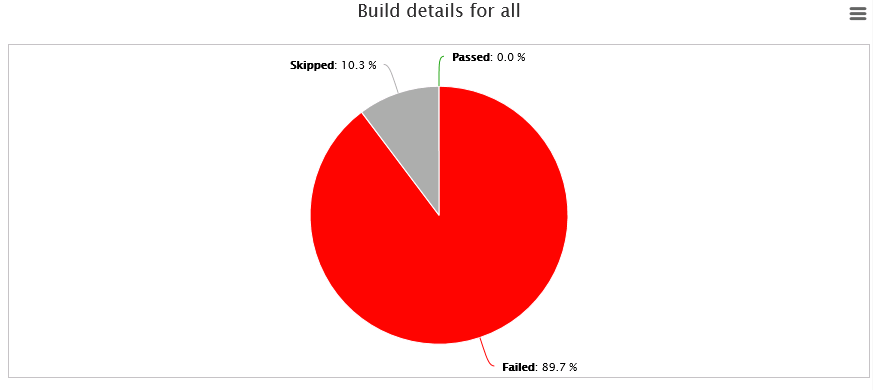


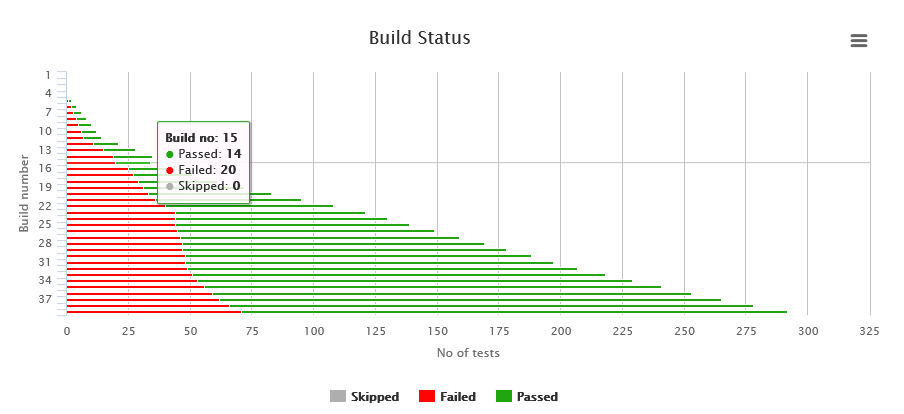
### Test Results Analyzer

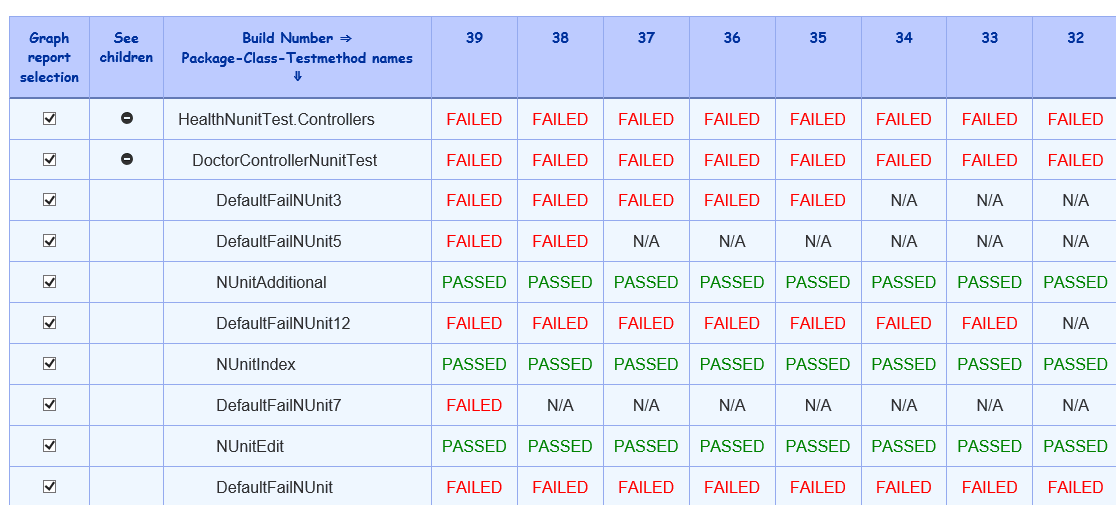
This reporting is displayed after installing Test Results Analyzer Plugin.

Test Results Analyzer dashboard allows to generate report as per selection criteria’s. It provides various graphical charts to summarize the result.







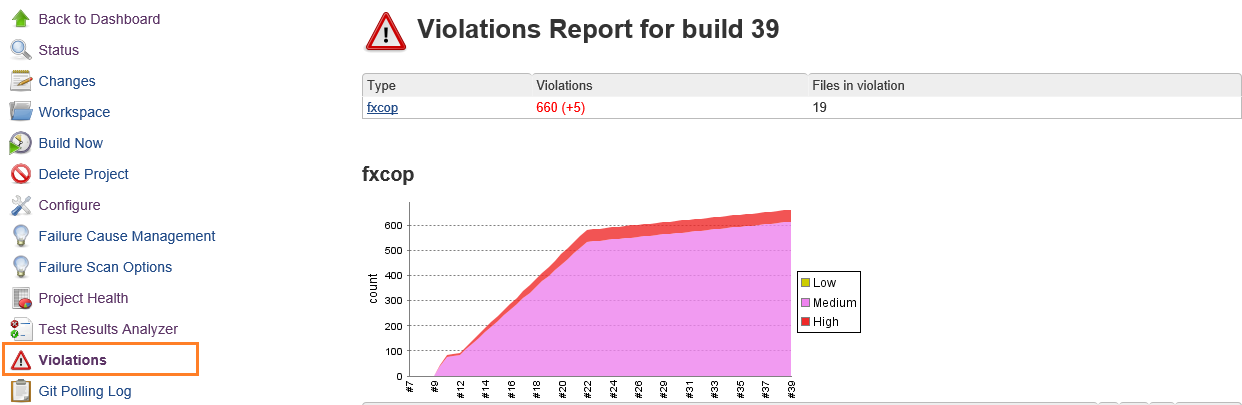


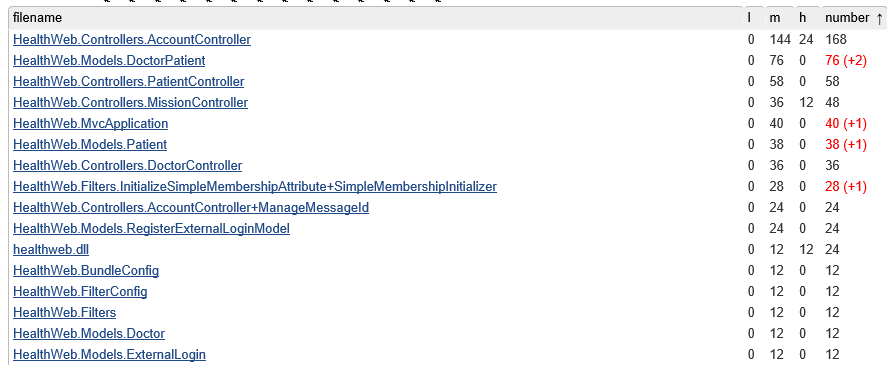
### Violation

This reporting is displayed after installing Violation Plugin.

Violation dashboard displays code violations from various tools like FxCops, stylechecks etc.

For this demo we have configured Violation to show FxCop result analysis.



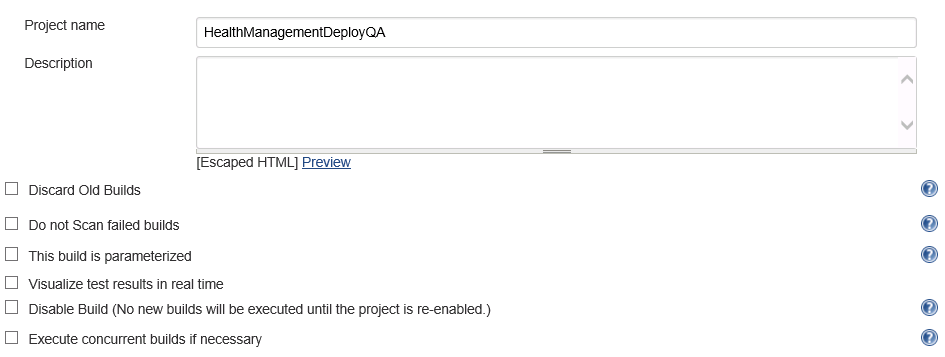


# Jenkins Project: HealthManagementDeployQA

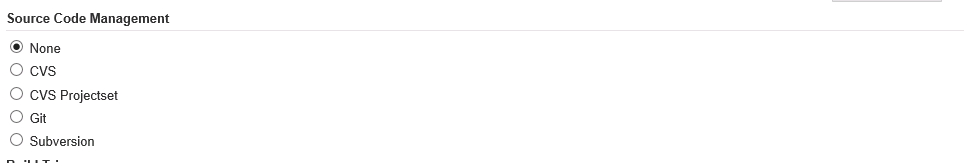
This Jenkins freestyle project/build is created for continuous deployment of a sample .Net application build and validated by Jenkins build HealthManagement.

## Configurations

### Project Name



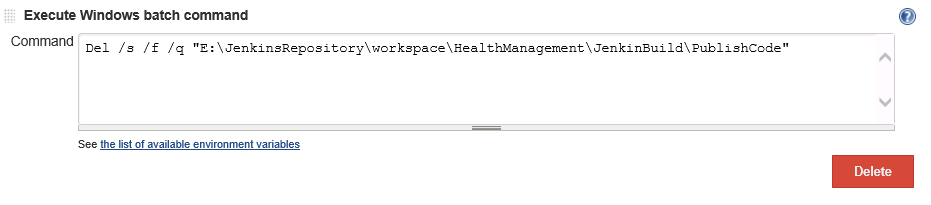
### Source Code Management- None



### Build – Clean Previous Publish from Jenkins

Command:

*Del /s /f /q "E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\PublishCode"*



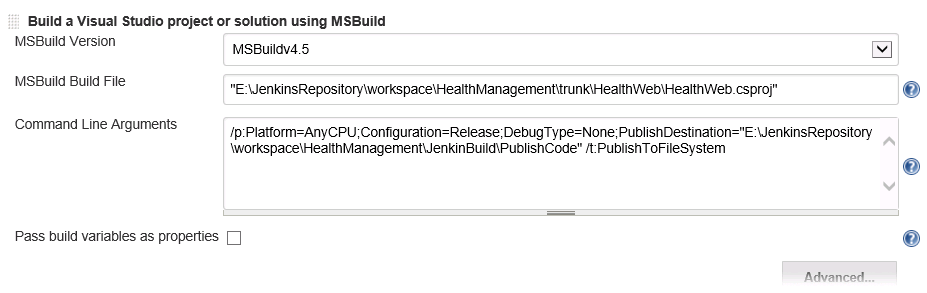
### Build – MSBuild create publish code

MSBuild Build File:

*"E:\JenkinsRepository\workspace\HealthManagement\trunk\HealthWeb\HealthWeb.csproj"*

Command Line Arguments:

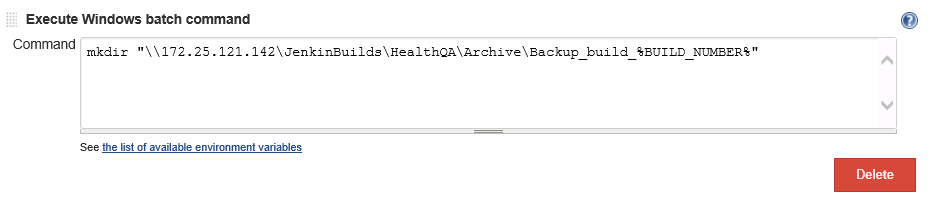
*/p:Platform=AnyCPU;Configuration=Release;DebugType=None;PublishDestination="E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\PublishCode" /t:PublishToFileSystem*



### Build – Archive previous deployment

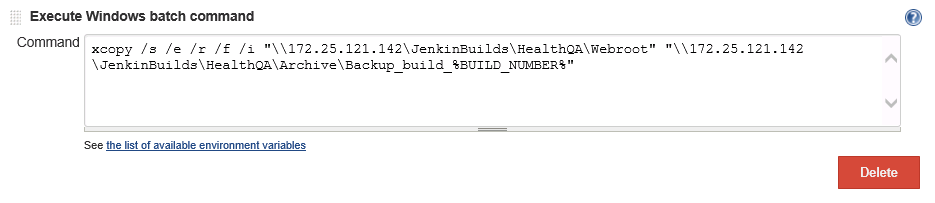
Command:

*mkdir* [*\\172.25.121.142\JenkinBuilds\HealthQA\Archive\Backup\_build\_%BUILD\_NUMBER%*](file:///\\172.25.121.142\JenkinBuilds\HealthQA\Archive\Backup_build_%25BUILD_NUMBER%25)



Command:

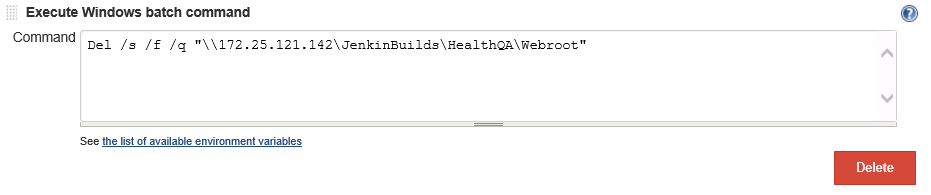
*xcopy /s /e /r /f /i "\\172.25.121.142\JenkinBuilds\HealthQA\Webroot"* [*\\172.25.121.142\JenkinBuilds\HealthQA\Archive\Backup\_build\_%BUILD\_NUMBER%*](file:///\\172.25.121.142\JenkinBuilds\HealthQA\Archive\Backup_build_%25BUILD_NUMBER%25)



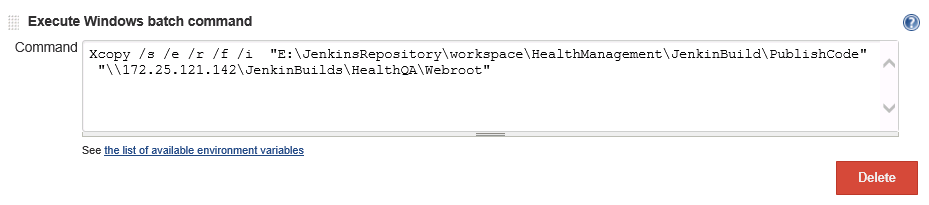
### Build – Deploy

Command:

*Del /s /f /q* [*\\172.25.121.142\JenkinBuilds\HealthQA\Webroot*](file:///\\172.25.121.142\JenkinBuilds\HealthQA\Webroot)



Command:

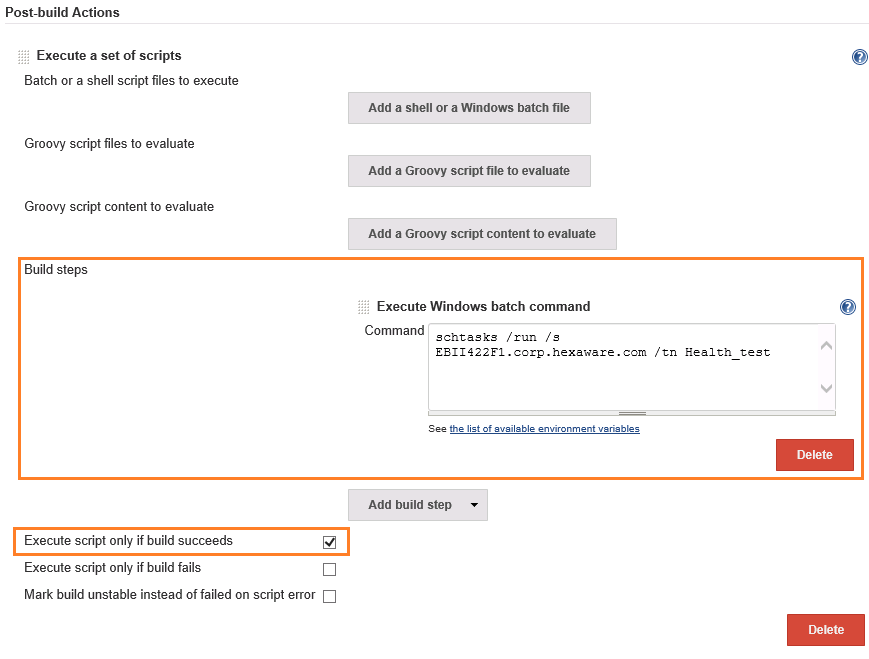
*Xcopy /s /e /r /f /i "E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\PublishCode" "\\172.25.121.142\JenkinBuilds\HealthQA\Webroot"*

### Post-Build Actions – Run selenium tests

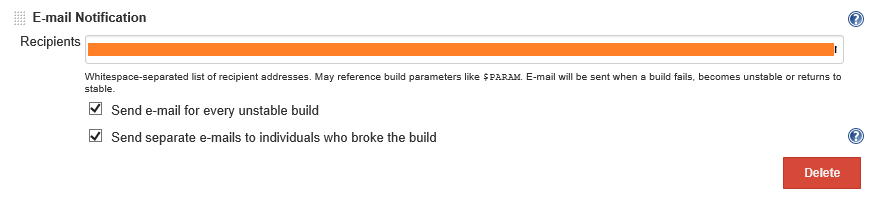
Once Jenkins build for deployment is successful, below post build action will run automated selenium test cases hosted as scheduled task on different server- *EBII422F1.corp.hexaware.com*.

Command:

*schtasks /run /s EBII422F1.corp.hexaware.com /tn Health\_test*



### Post-Build Actions – Send email notifications



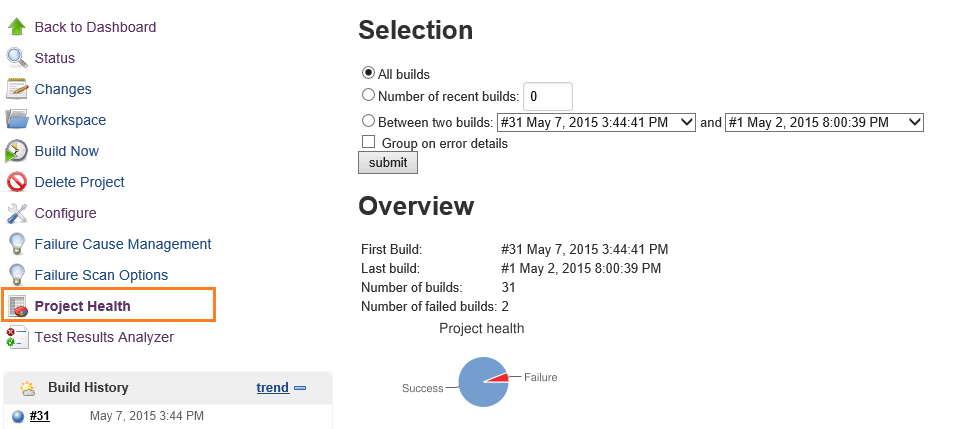
## Reports

Open the dashboard for HealthManagementDeployQA Jenkins project to see the reporting options. Since this Jenkins project objective it to deploy the code changes, only Project Health reporting option is used to show overall build status.

### Project Health

This report is displayed after installing Project Health Plugin.

Project Health dashboard shows the overview of builds as per selection in graphical chart.



# Team Foundation Server with GIT

## Prerequisites

Below tools should be installed on development servers.

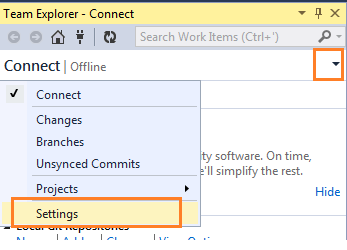
* Visual Studio 2013 with Team Foundation Server (TFS)
* Git

## TFS Integration with GIT

Open Team Explorer window to connect GIT repository from TFS.

### Global Git Setting

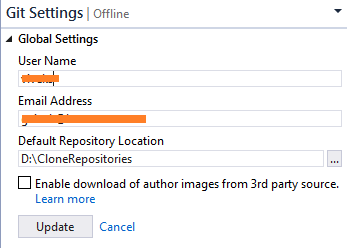
1. Click dropdown arrow on top of window and select Settings



1. Click on Git Settings link



1. Enter User Name, Email Address and Default Repository Location and Click Update button to save the default GIT settings.

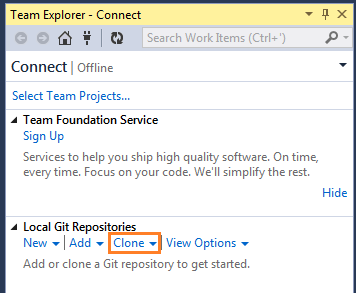


### Clone Git Repository

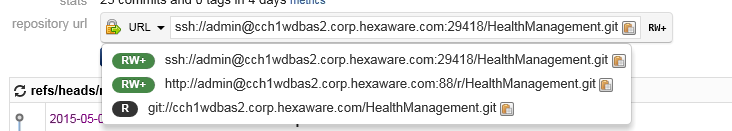
1. Clone the Git repository from server.

Click on dropdown arrow on top of window and select Projects>>Connect to Team Projects.

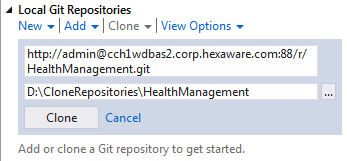
Click on Clone link from Local Git Repositories section.



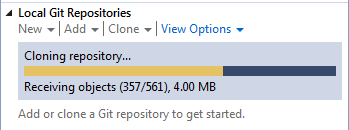
1. Get the repository URL from GitBlit



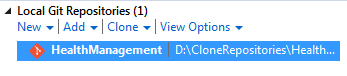
1. Enter the Git repository URL having RW permissions and the location where to pull the repository contains. Click Clone button to download the repository contains.



1. Pulling repository contains from server



1. Git repository clone created. Open the newly created clone by double click or right click and open.



### Managing Repository from TFS

Project sections show various options to manage the Git repository.

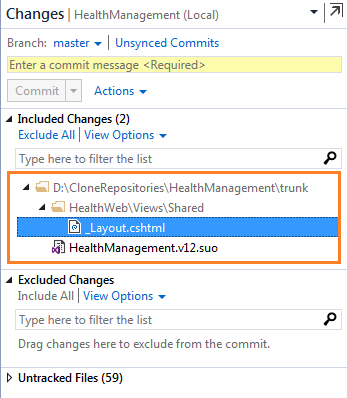
Solution Section shows all the visual studio project solution files from the Git repository.

#### 

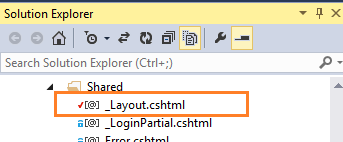
#### Open Solution

Open the git repository from Local Git Repositories section and open the solution file from Solutions section.

#### Changes- Push to Git Repository



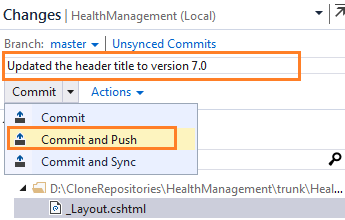
Modified files are shown as TICK in Solution Explorer.



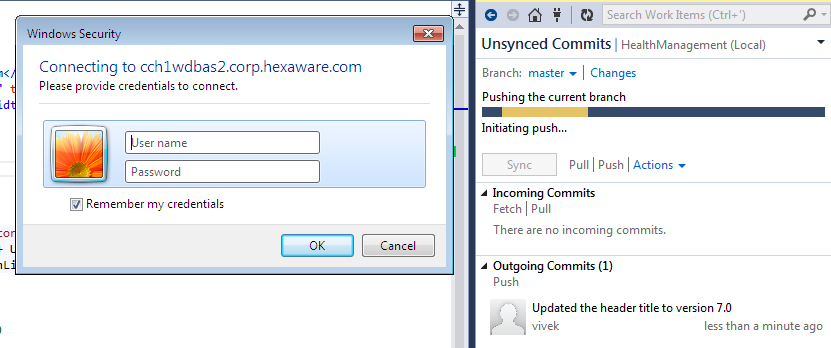
Compare history shows change in version number from 6.0 to 7.0 in Header.



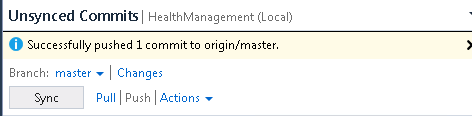
To push the modified changes to Git repository at server, add comment which describes the change and click Commit and Push.



Enter the credentials for server Git repository



Successfully pushed to server.

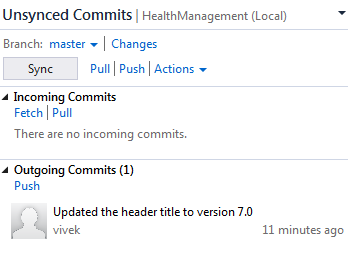


#### Unsynced Commits

This section allows to keep local repository in sync with server repository.

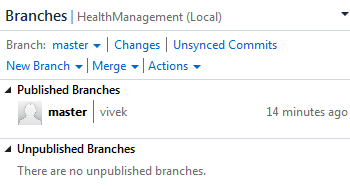
Pull option will get the Incoming Commits changes from server.

Push option will push the Outgoing uncommits to the server.



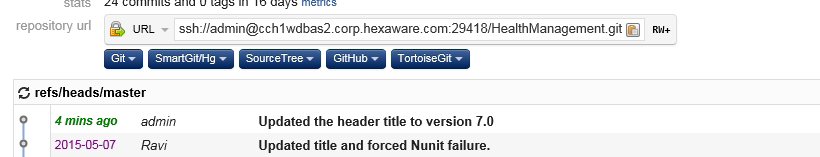
#### Branches

This section allows managing the branches for server Git repository.

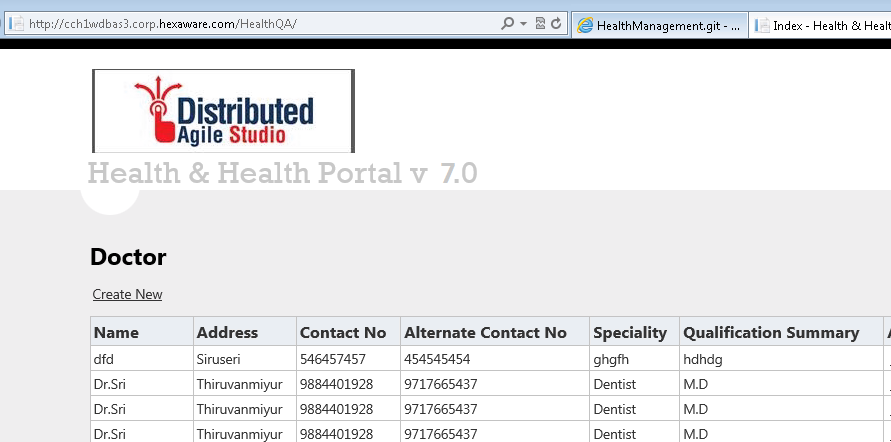


#### Results

Recent Pushed history is displayed on GitBlit site.



Header title is changed from 6.0 to 7.0.



# Appendix

## Scripts

### Nunit PowerShell Script

Below is the path for PowerShell scripts used for Nunit.

Server: cch1wdbas2

*E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\PowershellScripts\NUnit\_Jira\_Integration.config*

*E:\JenkinsRepository\workspace\HealthManagement\JenkinBuild\PowershellScripts\NUnit\_Jira\_Integration.ps1*

## References

### Study Materials

Jenkins- [http://jenkins-ci.org](http://jenkins-ci.org/)

Git- [http://www.git-scm.com](http://www.git-scm.com/)

GitBlit- [http://www.gitblit.com](http://www.gitblit.com/)

Nunit- <http://nunit.org>

FxCop- <https://msdn.microsoft.com/en-us/library/bb429476(v=vs.80).aspx>

### Downloads

Download latest stable version for software’s from below source.

Jenkins: <http://mirrors.jenkins-ci.org/windows/latest>

Git: <http://www.git-scm.com>

GitBlit: [http://www.gitblit.com](http://www.gitblit.com/)

Nunit: <http://nunit.org/index.php?p=download>

FxCop: <http://www.microsoft.com/en-gb/download/details.aspx?id=6544>

## Example

### Application Links

Jenkins: [http://cch1wdbas1.corp.hexaware.com:8080](http://cch1wdbas1.corp.hexaware.com:8080/)

GitBlit: [http://cch1wdbas2.corp.hexaware.com:88](http://cch1wdbas2.corp.hexaware.com:88/)

.Net Application: <http://cch1wdbas3.corp.hexaware.com/HealthQA>