|  |
| --- |
|  |
| Git User Guide |
| Agile DevOps Lab |
| Author: Vivek Gole  Created On: 22-May-2015  Summary: This document will guide users to use Git repositories from various client tools.  Source : git-scm.com |

Contents

[Introduction 3](#_Toc420328888)

[About Git 3](#_Toc420328889)

[Workflow 3](#_Toc420328890)

[Audience 4](#_Toc420328891)

[Prerequisites 4](#_Toc420328892)

[Git Client 5](#_Toc420328893)

[Using Git 5](#_Toc420328894)

[Git Bash 5](#_Toc420328895)

[Git GUI 10](#_Toc420328896)

[Team Foundation Server with GIT 16](#_Toc420328897)

[Prerequisites 16](#_Toc420328898)

[TFS Integration with GIT 16](#_Toc420328899)

[Global Git Setting 16](#_Toc420328900)

[Clone Git Repository 17](#_Toc420328901)

[Managing Repository from TFS 18](#_Toc420328902)

[Appendix 23](#_Toc420328903)

[References 23](#_Toc420328904)

[Study Materials 23](#_Toc420328905)

[Downloads 23](#_Toc420328906)

# Introduction

This document explains Git client tools.

## About Git

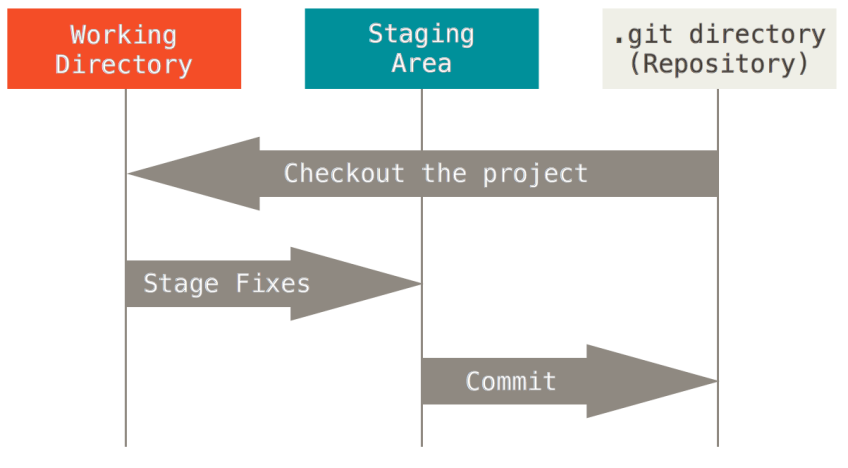
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

### Workflow

Git has three main states that your files can reside in: committed, modified, and staged.

* Committed means that the data is safely stored in your local database.
* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

This leads us to the three main sections of a Git project: the Git directory, the working directory, and the staging area.



The Git directory is where Git stores the metadata and object database for your project. This is the most important part of Git, and it is what is copied when you clone a repository from another computer.

The working directory is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

The staging area is a file, generally contained in your Git directory that stores information about what will go into your next commit. It’s sometimes referred to as the “index”, but it’s also common to refer to it as the staging area.

To summarize, the basic Git workflow goes something like this:

* You modify files in your working directory.
* You stage the files, adding snapshots of them to your staging area.
* You do a commit and push, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

## Audience

As this document only highlights the configuration and use of git client tools, we expect the intended audience should have the following skills:

* Familiar with Git version controlling.
* Familiar with Visual Studio 2013 and Team Foundation Server.

## Prerequisites

Following are the software required to configure GitBlit.

* GitBlit – This is used as git server manager.
* Git- This is used as a basic git client tool.
* Visual Studio 2013 with Team Foundation Server (TFS).

# Git Client

Git client tool is used to created/clone git repositories.

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

Choose default settings while installing the software.

## Using Git

Git for Windows focuses on offering a lightweight, native set of tools that bring the full feature set of the Git SCM to Windows while providing appropriate user interfaces for experienced Git users and novices alike.

After installation you should see Git Bash and Git GUI in start menu>>program files.

* Git Bash- Git for Windows provides a BASH emulation used to run Git from the command line.
* Git GUI- As Windows users commonly expect graphical user interfaces, Git for Windows also provides the Git GUI, a powerful alternative to Git BASH, offering a graphical version of just about every Git command line function, as well as comprehensive visual diff tools.

### Git Bash

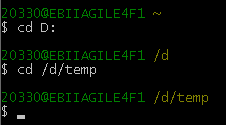
This topic explains basic use of bash console for managing git repository. Refer git help command or site documentation section for more details information.

#### Accessing Bash

Git bash can be accessible from startup menu, selecting Bash option from Git GUI, or if you have installed Git with allow Simple context menu, then it can be accessed from context menu.

You can change the targeted folder location for command execution using cd statements similar to unix commands.

Here we are redirecting to D:/Temp folder.



#### Help

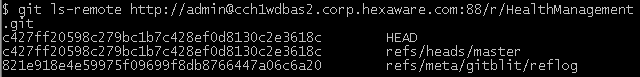
Command: *git help*



#### Validate Remote URL

Command: *git ls-remote URL*

Enter the credentials to access the URL if asked.



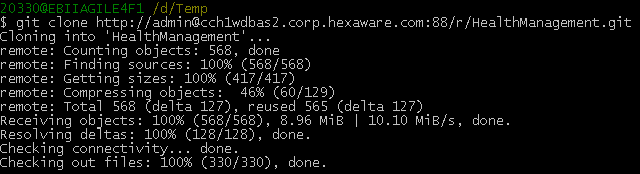
If command returns any error, most probably cause would be network access rights when accessing URL host server.

#### Clone Remote Repository

Command: *git Clone URL*

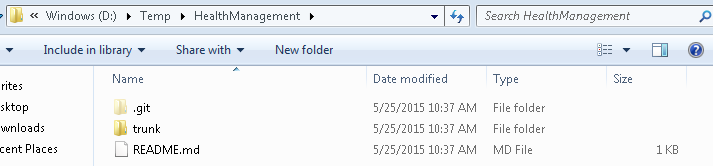
Enter the credentials to access the URL if asked.

It will clone and copy the contents from URL in location where Bash command is executed.



Repository cloned at location D/Temp folder.

The .git folder inside HealthManagement maintains all git changes/status for local cloned repository.



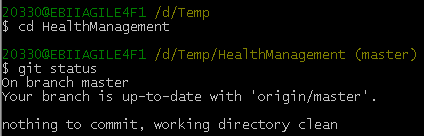
#### Check Status

Command: *git status*

Error as .git folder was not found in location d/Temp.



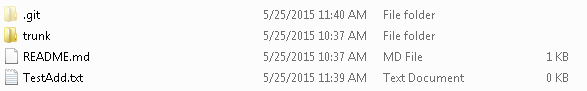
After moving inside HealthManagement folder were .git folder is maintained. Git status returns the current status for local repository.

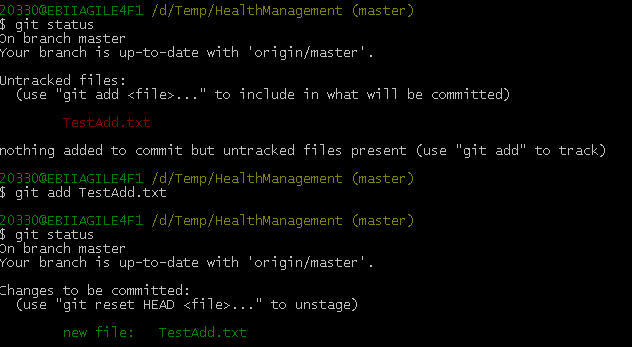


#### Add new files

Command: *git add filelocation*

Here added a new file TestAdd.txt in root folder. After adding the file **git status** shows newly added file which is marked as **Untracked**. After using **git add** command to add the file, the status of the file is changed to **Changes to be Committed** (ready to be commit to server).





Similarly all the files which are untracked needs to be added to using Git add command.

#### Remove files

Command: *git rm file*

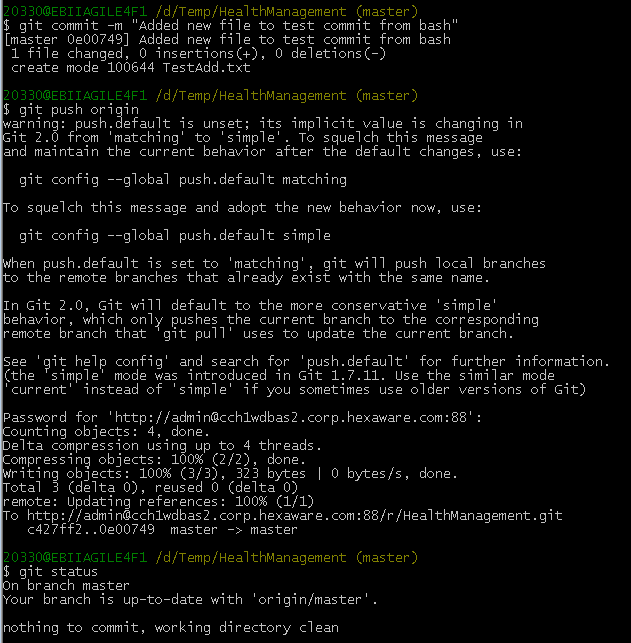
Follow similar steps as git add.

#### Commit & Push

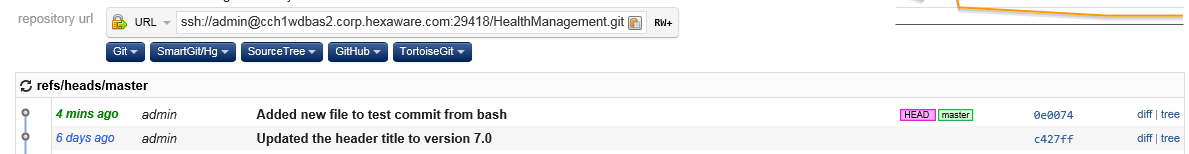
Command: *git commit –m MESSAGE*

Command: git Push <remote URL>

Enter credentials if asked.



After commit and push, changes are reflected in GitBlit site.



#### Other commands:

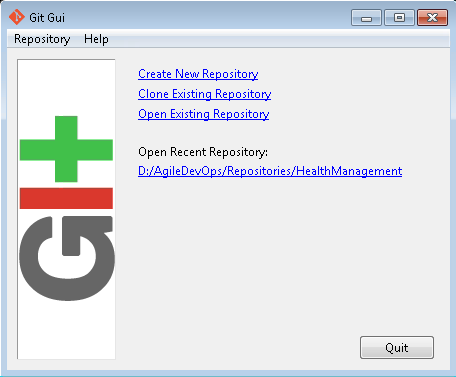
For more commands refer documentation section from git-scm.com.

### Git GUI

This topic explains basic use of Git GUI windows for managing git repository.

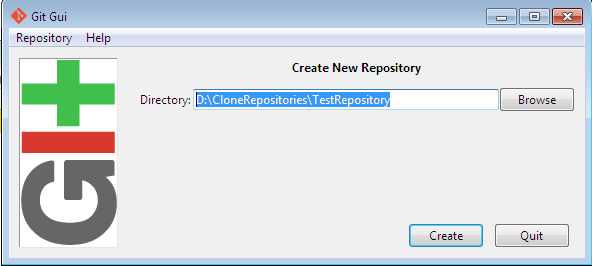
#### Default Page

Here it allows users to create new repository, clone existing repository and open repository.

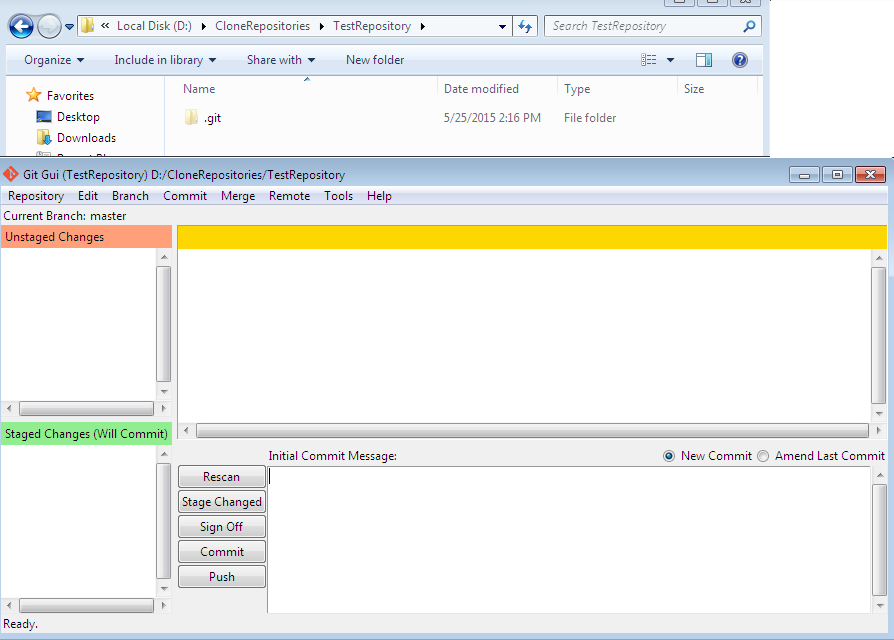


#### Create Repository

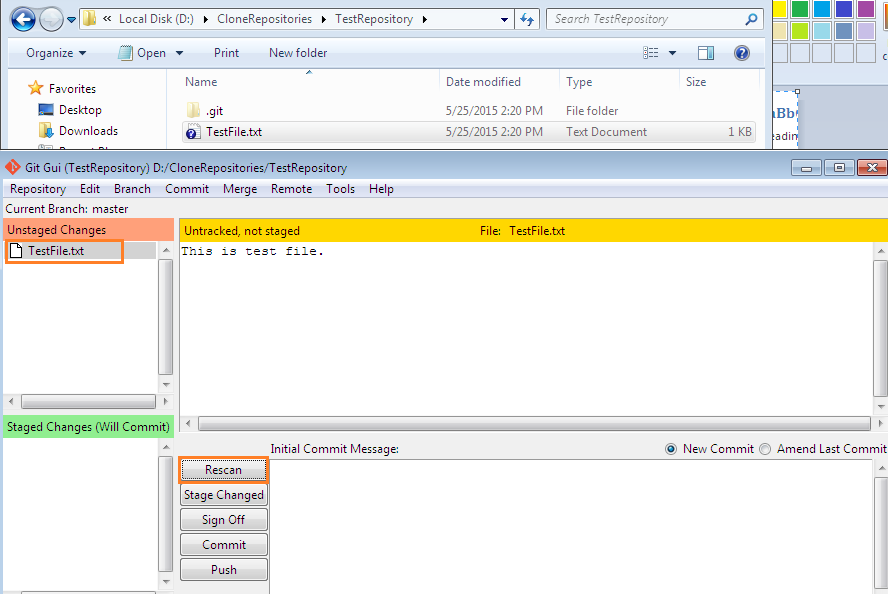
Click Create New Repository. Enter directory/folder location where you want to create the new git repository.



A new folder “TestRepository” is created in provided location with hidden .git folder. Currently no files are versioned under this git repository.

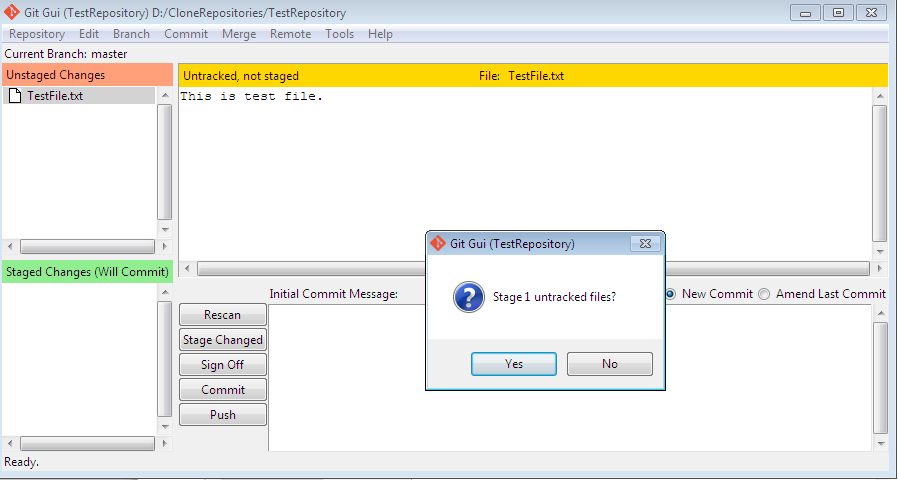


Add new file “TestFile.txt” to repository location. Click Rescan in Git GUI to see the newly added file in Unstaged Changes section. On selecting the file from Git GUI, it show the content inside it in right side panel.

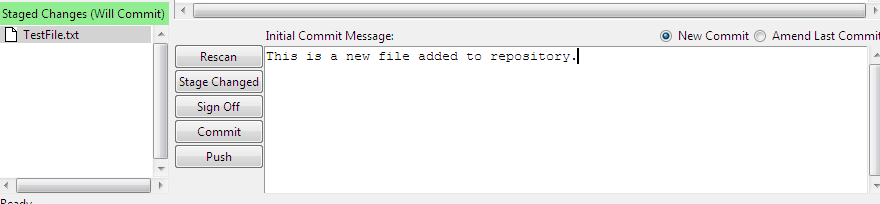


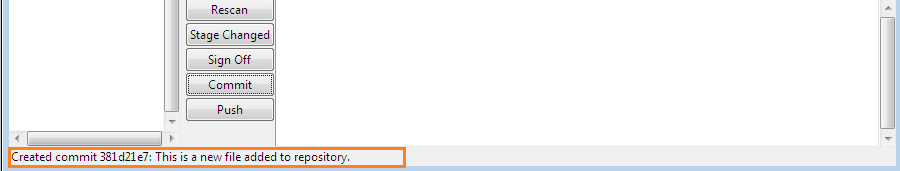
Before committing, this file needs to be moved to Staged Changes section (Tracked files). Do this by click on button Stage Changed.

It shows message box, if you select No, it will move only selected files to Staged Changed section, and otherwise it will move all the unstaged files.



After staging the file, add comment and commit the changes by clicking button Commit.



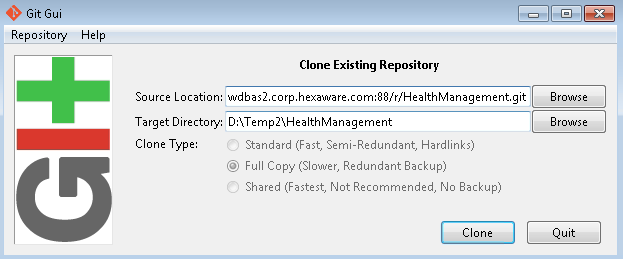
After commit, a message will be displayed displaying the status of the commit. In case of remote repository click on Push button to push the changes to remote repository. 

#### Clone Repository

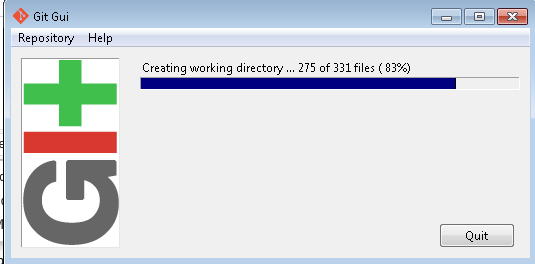
With this option you can clone the exiting repository with its remote URL and get the content of that repository.

From Git GUI click on Clone Repository. Enter the remote URL for repository which you want to clone. Here we have used HealthManagement git repository from GitBlit site.

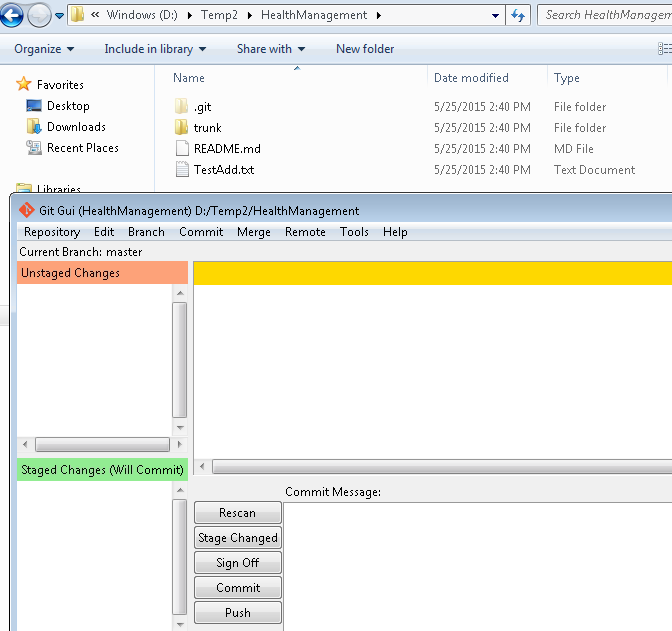
Enter the targeted directory and click clone.



On click of Clone button, Git GUI will start downloading the content from remote repository to targeted directory location.



After completely downloading, it will open the cloned repository in Git GUI similar as seen while creating repository. It also created a targeted directory with .git folder to manage the versioning with remote repository.



Rest all functionality for managing the cloned repository will be similar as we tried while creating new repository in previous section.

# Team Foundation Server with GIT

## Prerequisites

Below tools should be installed on development servers.

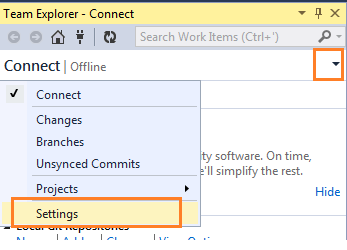
* Visual Studio 2013 with Team Foundation Server (TFS)
* Visual Studio Extension- Visual Studio Tools for Git
* Git

## TFS Integration with GIT

Open Team Explorer window to connect GIT repository from TFS.

### Global Git Setting

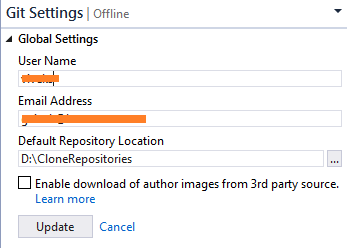
1. Click dropdown arrow on top of window and select Settings



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.

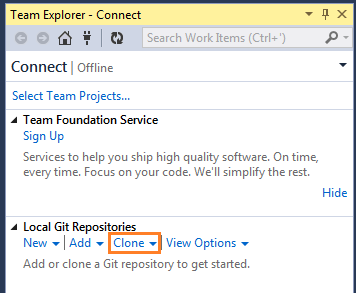


### Clone Git Repository

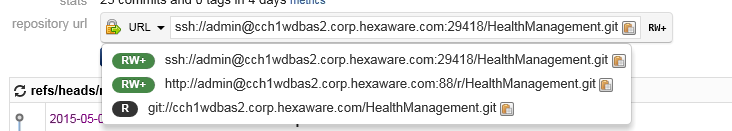
1. Clone the Git repository from server.

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

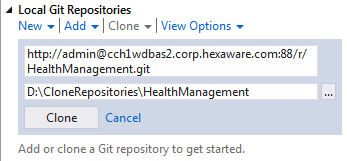
Click on Clone link from Local Git Repositories section.



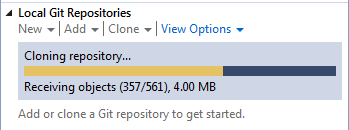
1. Get the repository URL from GitBlit



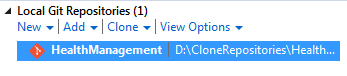
1. Enter the Git repository URL having RW permissions and the location where to pull the repository contains. Click Clone button to download the repository contains.



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.

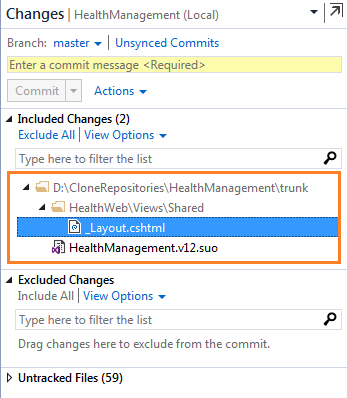
Solution Section shows all the visual studio project solution files from the Git repository.

#### 

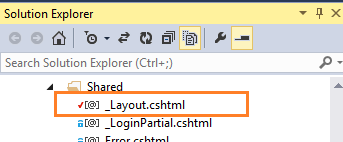
#### Open Solution

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

#### Changes- Push to Git Repository



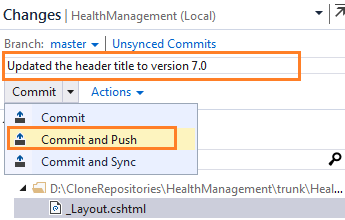
Modified files are shown as TICK in Solution Explorer.



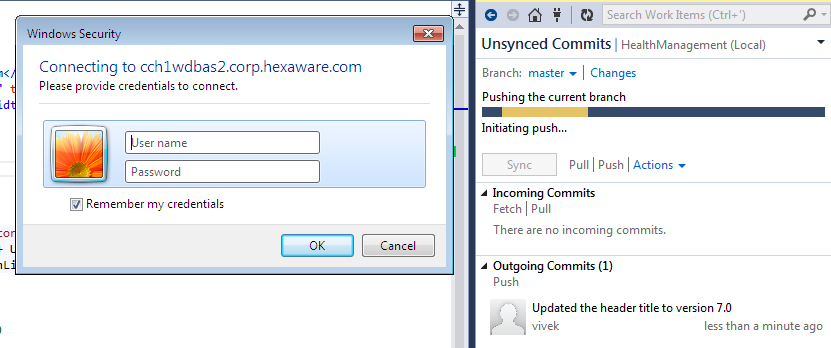
Compare history shows change in version number from 6.0 to 7.0 in Header.



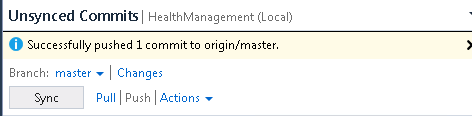
To push the modified changes to Git repository at server, add comment which describes the change and click Commit and Push.



Enter the credentials for server Git repository



Successfully pushed to server.

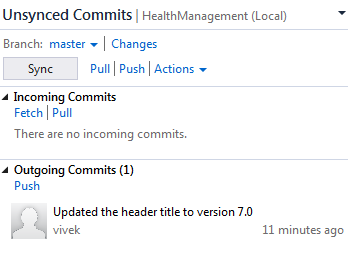


#### Unsynced Commits

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

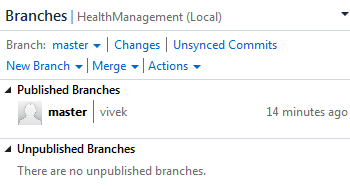
Pull option will get the Incoming Commits changes from server.

Push option will push the Outgoing uncommits to the server.



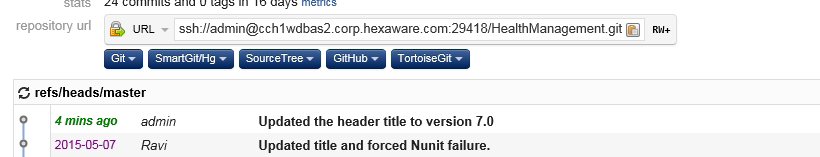
#### Branches

This section allows managing the branches for server Git repository.

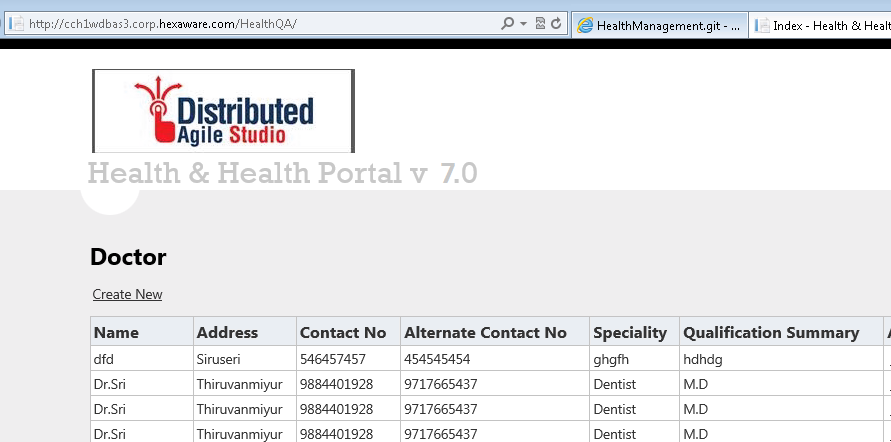


#### Results

Recent Pushed history is displayed on GitBlit site.



Header title is changed from 6.0 to 7.0.



# Appendix

## References

### Study Materials

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

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

### Downloads

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

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

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

Extension-Visual Studio tool for Git: <https://visualstudiogallery.msdn.microsoft.com/abafc7d6-dcaa-40f4-8a5e-d6724bdb980c>