Git: Best Practises

|  |  |
| --- | --- |
| Author: | Richard Stevenson |
| Document Name: | Git: Best practice concrete examples |
| Document Location: | \\lukpc210\Projects\Product Projects\Operations\Vitic 2017.1\Project Documents\POC\VXRW-272-git-best-practice-concreate-examples.docx |
|  |
| Date Created: | 10/03/2017 |
| Date Last Modified: | 10/03/2017 |
| Status: | Draft |
| Confidentiality: | <Select/enter confidentiality status> |
| Circulation: | <Detail circulation list> |
| Document Version: | 0.1 |

Contents

[1. Overview 3](#_Toc477170059)

[2. Cloning a Repository 4](#_Toc477170060)

[2.1 Git Bash 4](#_Toc477170061)

[2.2 Eclipse Plugin 5](#_Toc477170062)

[3. Create a branch on the remote 7](#_Toc477170063)

[3.1 Git Bash 7](#_Toc477170064)

[3.2 Eclipse Plugin 8](#_Toc477170065)

[4. Commit code using different commits and then squash 10](#_Toc477170066)

[4.1 Command Line 10](#_Toc477170067)

[10](#_Toc477170068)

[4.2 Eclipse Plugin 13](#_Toc477170069)

[5. Commit code with git amend 16](#_Toc477170070)

[5.1 Command Line 16](#_Toc477170071)

[5.2 Eclipse Plugin 16](#_Toc477170072)

[6. Rebase from Main Line of Development 17](#_Toc477170073)

[6.1 Command Line 17](#_Toc477170074)

[6.2 Eclipse Plugin 18](#_Toc477170075)

[7. Merge to Main Line of Development 20](#_Toc477170076)

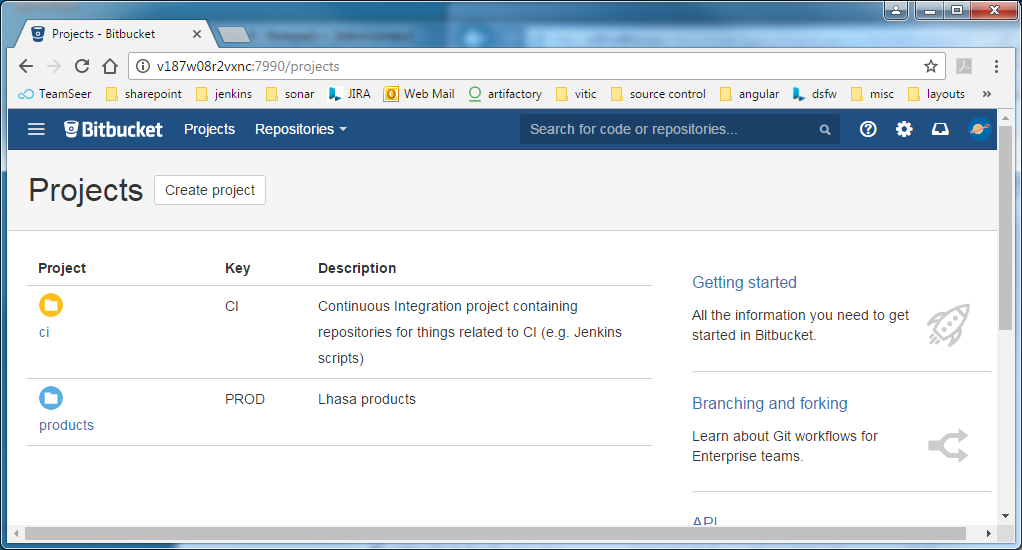
# Overview

Provide some concrete examples of how to apply git best practice using the command line and the eclipse git plugin.

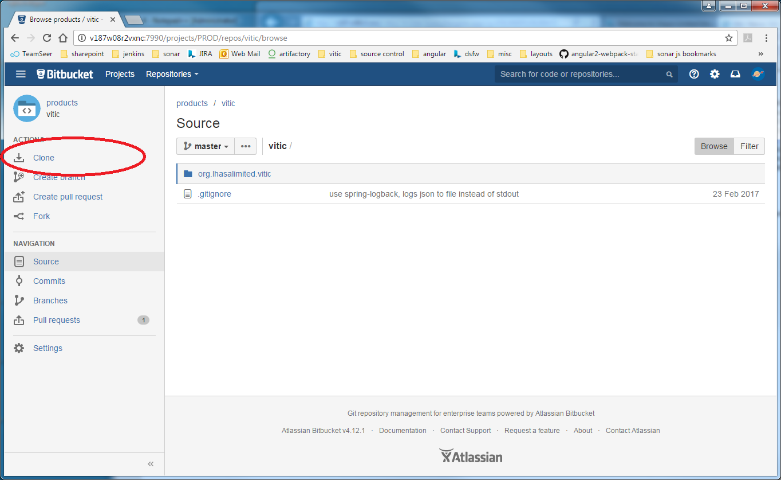
# Cloning a Repository

Get the code from source control.

Navigate to the repository you want to clone in bitbucket, in this case products > vitic

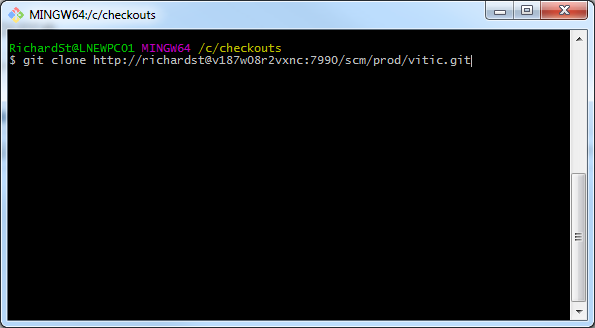


Click the clone option and you will get a url to use

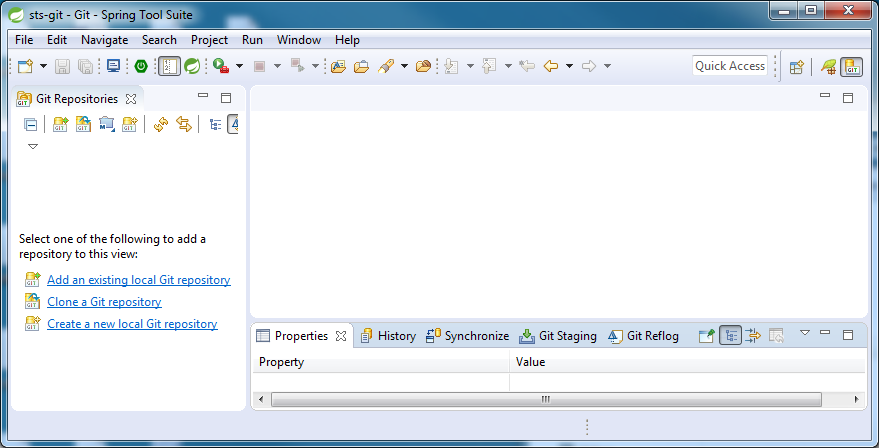


## 2.1 Git Bash

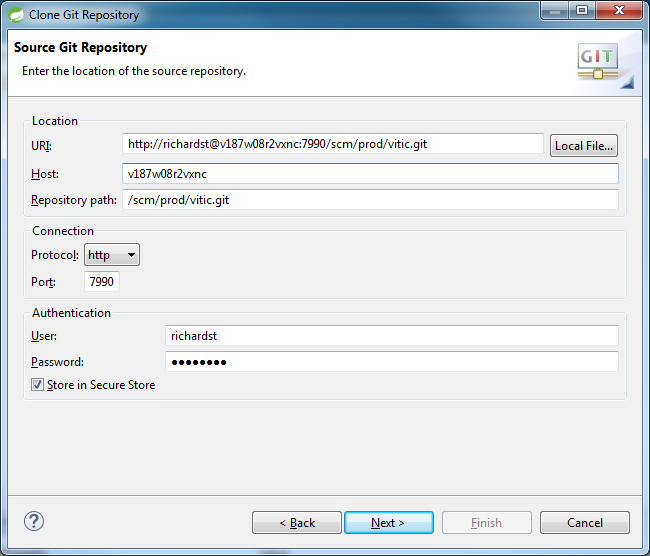
Simply type the following:

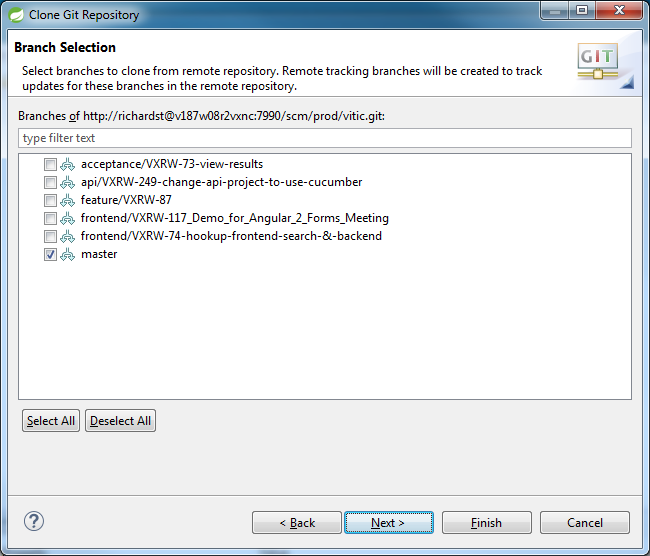
git clone <url>

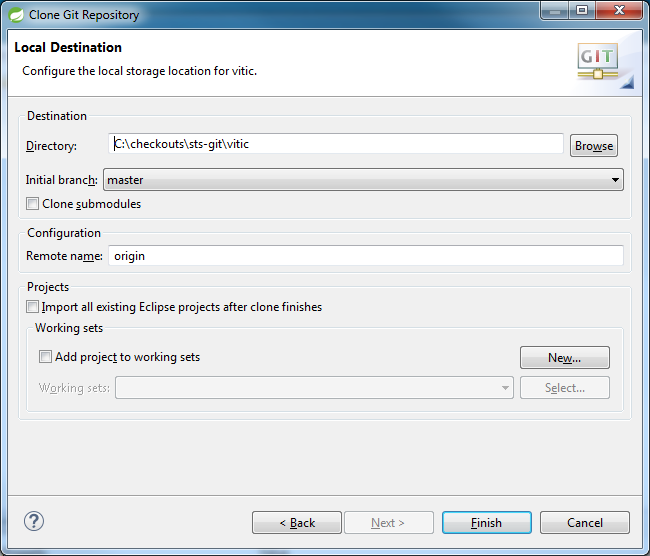
## 2.2 Eclipse Plugin

Navigate to the Git Perspective

Click on the clone icon:

enter the URI in the dialog, finding the clone URI from bitbucket and copying and pasting it into the URI on the dialog will fill in most of the dialog automatically for you, you simply have to provide a password.

Select master as the branch you are interested in:

Choose checkout directory and initial branch, leave remote as origin:

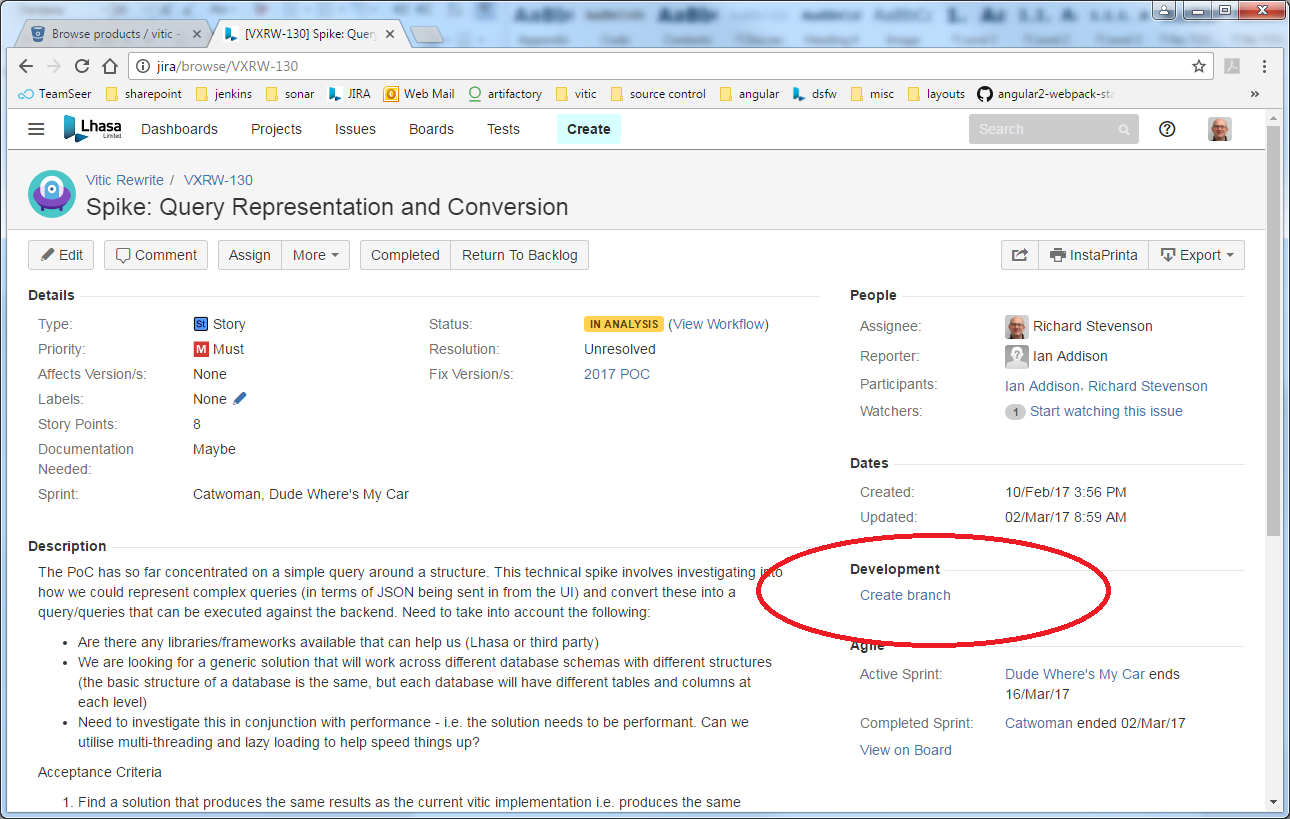
Click finish

.

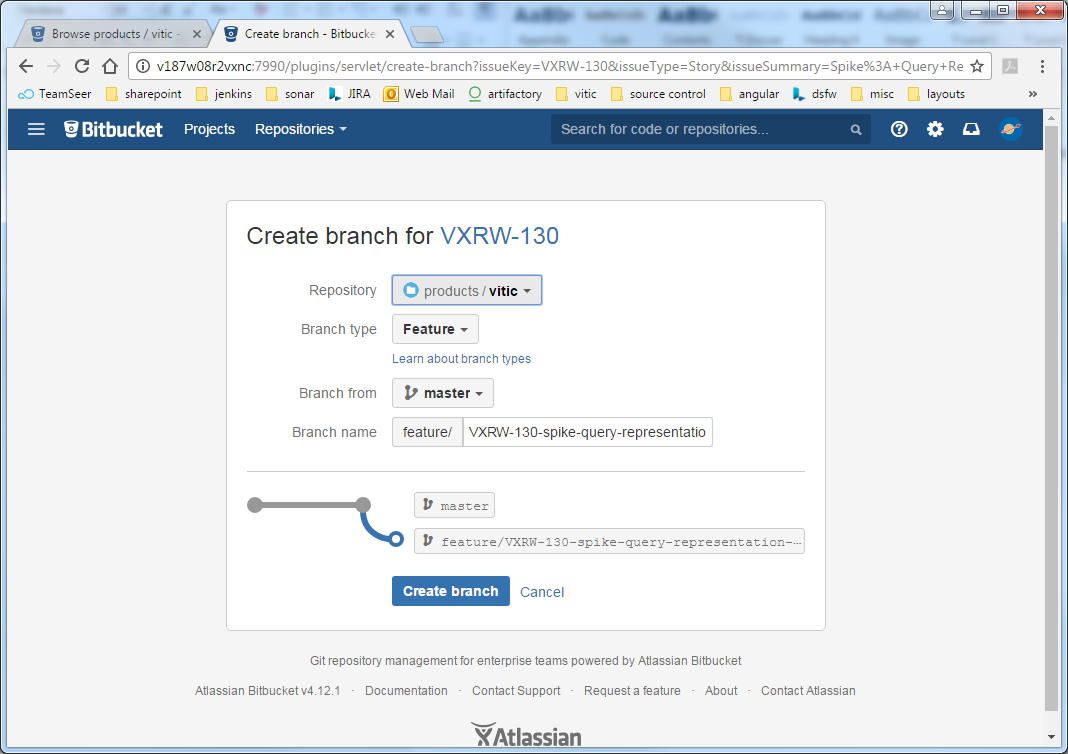
# Create a branch on the remote

Use Jira integration to create a branch on the remote then pull it to your local machine and check it out.

Navigate to your ticket in Jira and click on the create branch link



This will take you to BitBucket where you can create a branch using the form



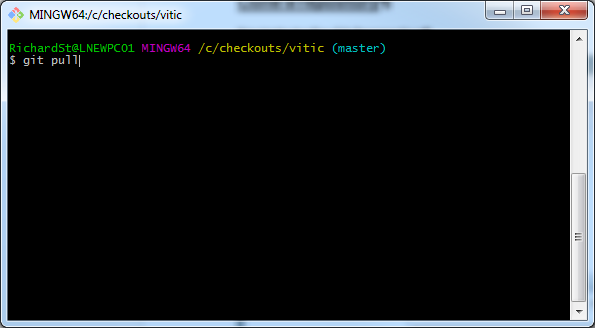
Once you have created the branch you can pull it down to your local machine and check it out

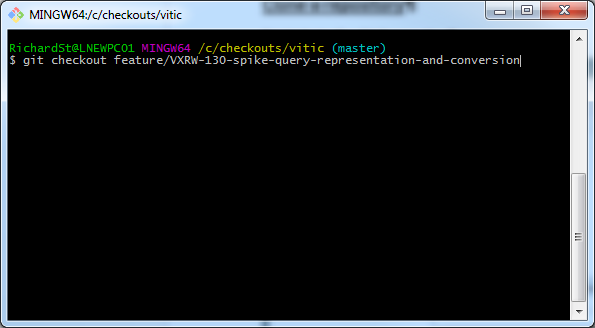
## 3.1 Git Bash

Use the following commands to pull changes from the remote to your local machine and check out your branch.

git pull

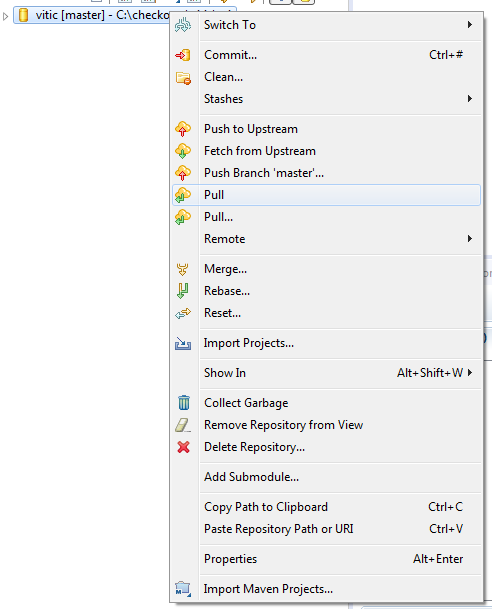
git checkout <brachname>



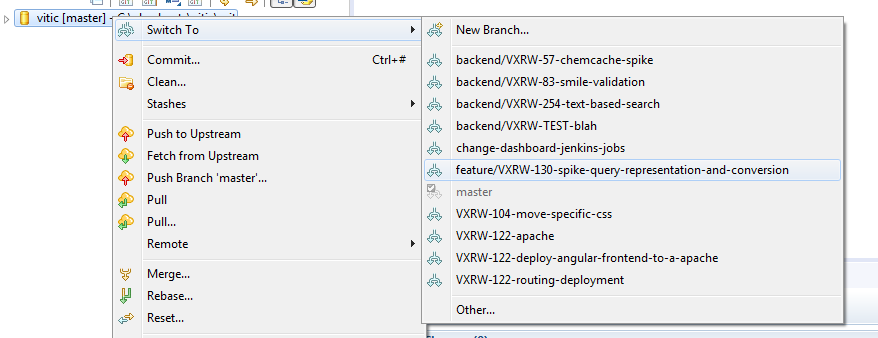


## 3.2 Eclipse Plugin

Pull the changes from the remote to your local machine by navigating to the git perspective and right clicking on the repository and selecting the Pull option



Switch to your branch



# Commit code using different commits and then squash

There are a number of ways of committing code but we want to ensure that when we merge back into the main line of development it is simple, robust and importantly it is easy to see what has happened in the past.

Commit messages should follow the standard git commit message format, max 50 characters in the summary followed by a blank line and max 72 characters in the remaining text.

## 4.1 Command Line

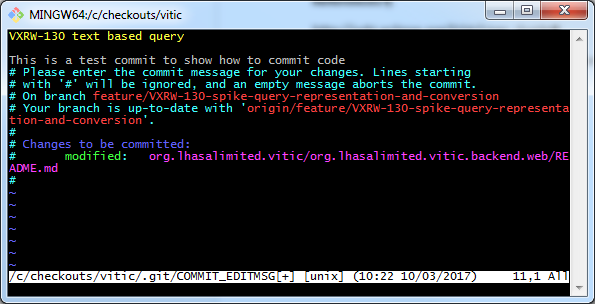
Commit code using following commands, use of git status gives us an indication of where we are, red files are un-staged and green files are staged.

git add x

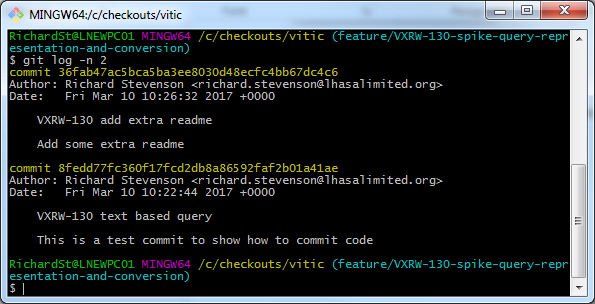
git commit

## 

When we type git commit this takes us to a vi editor where we can type our message.

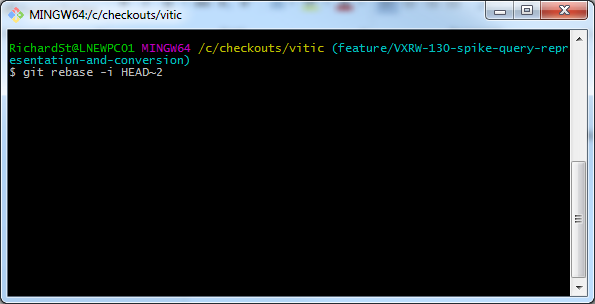
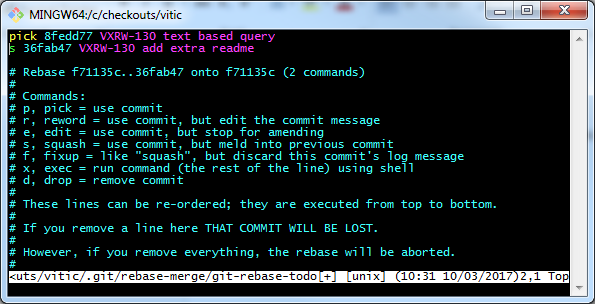


Save this file like you would any vi file and the commit will happen locally.

Once you have done this a twice you will have commits that you want to squash together, you can see these using the git log command.

Now we can squash the two commits together using the rebase command:

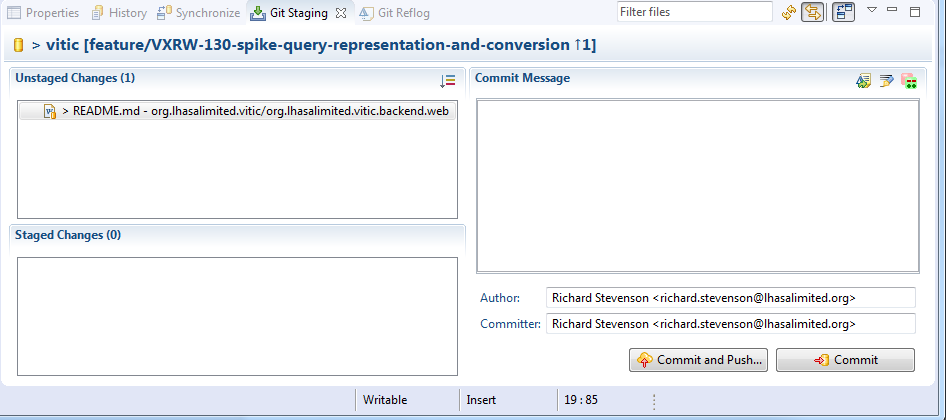
git rebase –i HEAD~<number of commits>

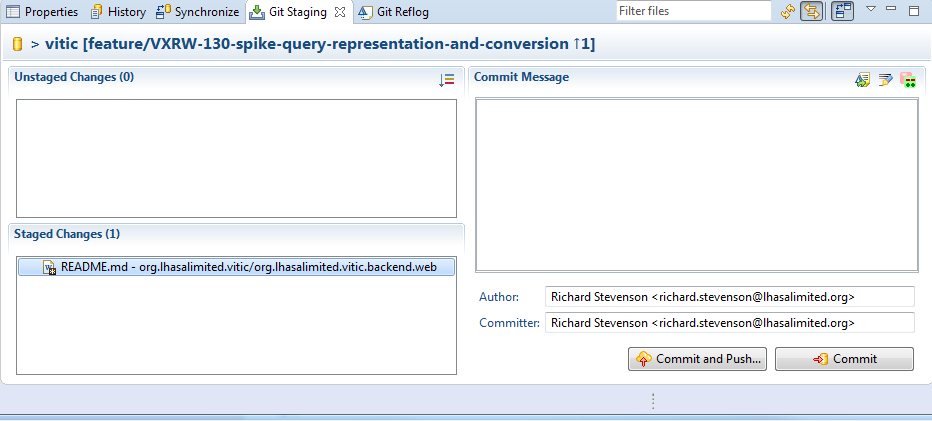
This will take you to the interactive screen which is a vi editor, here we have to pick which commits we want to squash by changing pick to s.

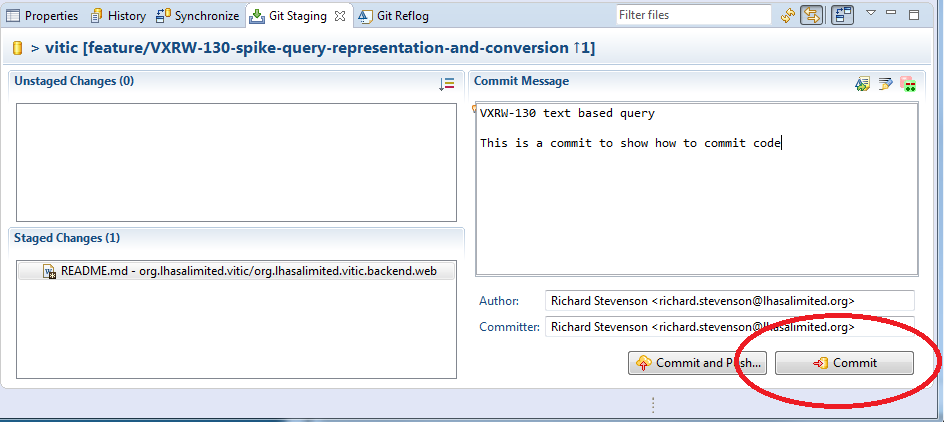
Then when we save that we have to edit the commit comments that we made for all the commits into one comment. When we save that the commits will be squashed together.

It is important to know that this rewrites the git commit objects and if your commit objects have previously been pushed to the remote you will now have diverged from the remote and you will have to force push the changes. This can be an issue if anyone else is working on your branch. NEVER do a rebase on the main line of development.

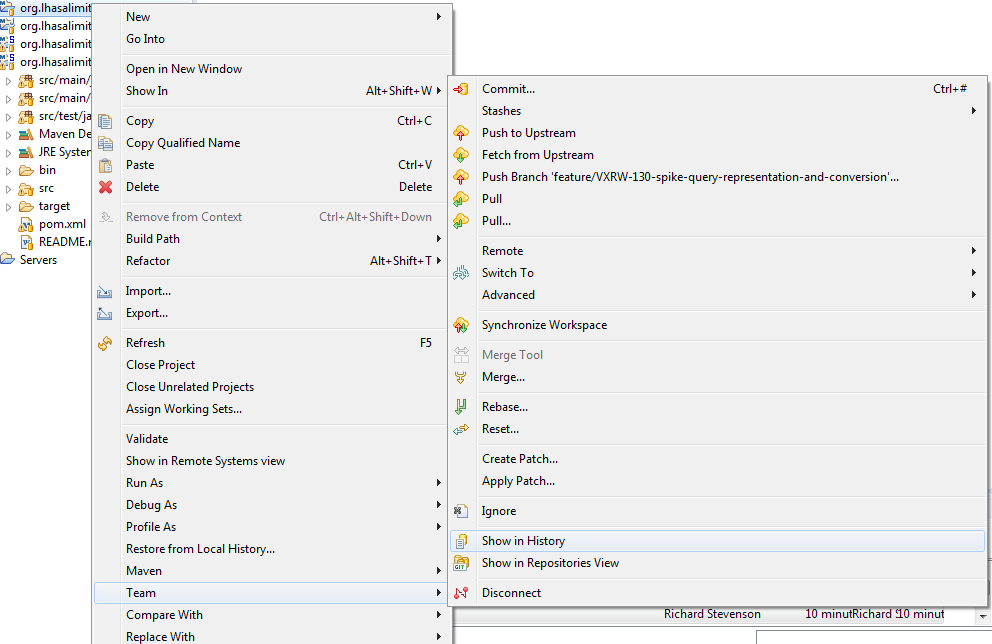
## 4.2 Eclipse Plugin

Make some changes to a file, then navigate to the git perspective and look at the staging area. Notice that there are changes in the un-staged area.

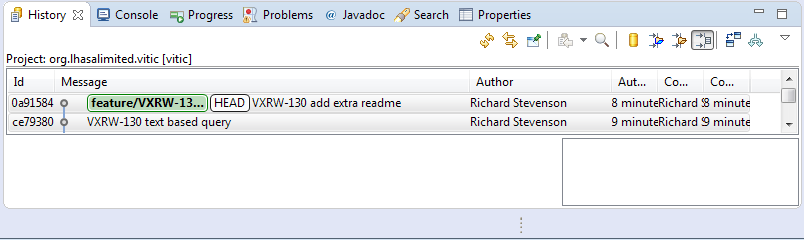
drag these changes into the staged area

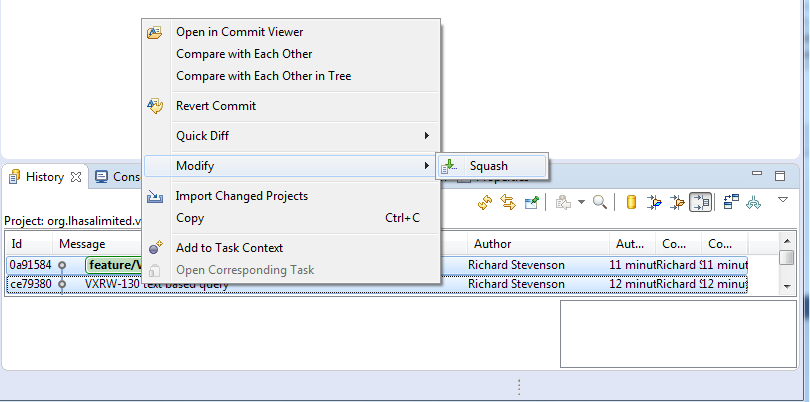
Type a commit message then hit the commit button

Once you have a few commits and you want to squash them.

Right click in navigator and select Team > Show in History

Select the commits you want to squash in the history tab



Then right click Modify -> Squash

It is important to know that this rewrites the git commit objects and if your commit objects have previously been pushed to the remote you will now have diverged from the remote and you will have to force push the changes. This can be an issue if anyone else is working on your branch. NEVER do a rebase on the main line of development.

# Commit code with git amend

If you don’t want to have many commits you can simply use git amend to add to the previous commit. For the examples below assume we already have one commit on our branch.

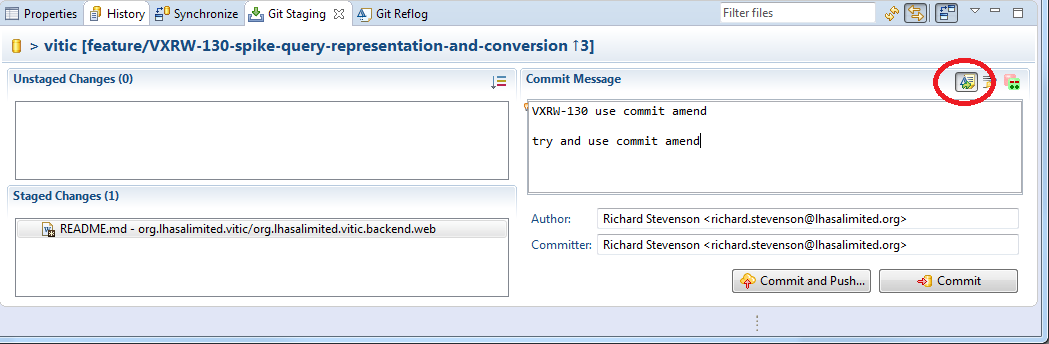
## 5.1 Command Line

Now we have made some changed that need committing but we want to add it to our previous commit.

git commit –amend

This will add anything we have in the staged area to your previous commit whilst allowing you to change the commit message. NOTE you have to be careful if doing this with any commits that have been pushed to the remote.

## 5.2 Eclipse Plugin

To amend the previous commit using the eclipse plugin click the Amend (Edit Previous Commit) checkbox as shown below, just like the command line this will add anything we have in the staged area to your previous commit whilst allowing you to change the commit message. NOTE you have to be careful if doing this with any commits that have been pushed to the remote.

# Rebase from Main Line of Development

It is good practice to rebase from the main line of development for a number of reasons:

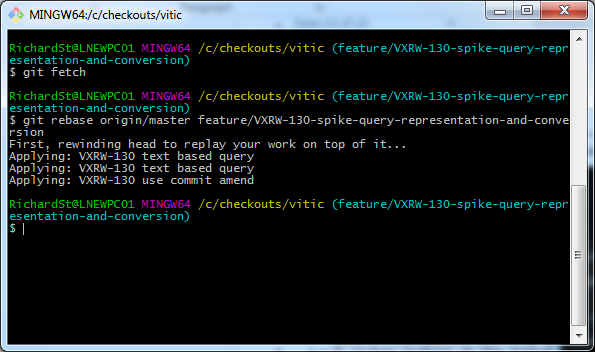
* It ensures that you always run all the code on your branch before committing removing the possibility that you merge your code into the main line and cause the build to fail because the code is incompatible with what other people have added after you took your branch.
* It makes looking at the history of what has happened in the past clear, commits are not interleaved with each other.
* If you are on a long running branch it ensures that you don’t get out of synch and have one large merge full of conflicts

## 6.1 Command Line

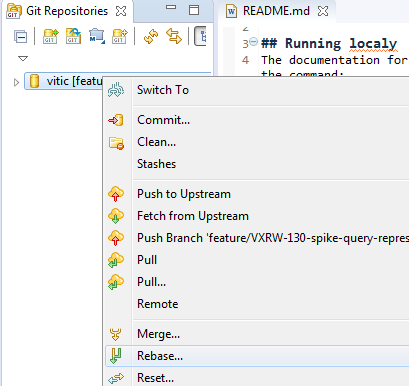
Use the following commands

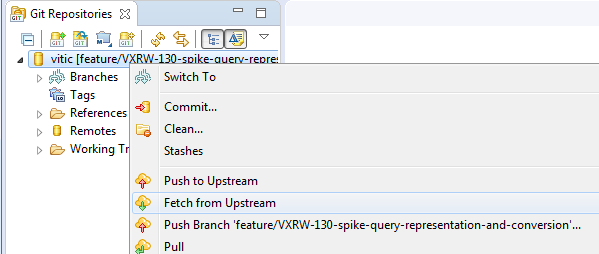
git fetch

git rebase <main line> <branch>

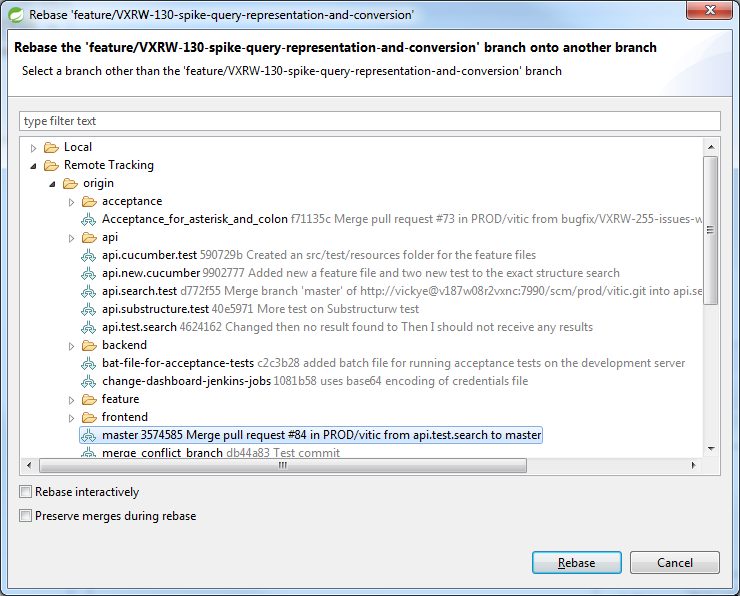
In the following example the main line of development is master and the branch is feature/VXRW-130-spike-query-representation-and-conversion. NOTE it is important to fetch all the latest code from the remote. Used the rebase command with origin/master, not my local checkout of master which could be pointing at a different commit.

## 6.2 Eclipse Plugin

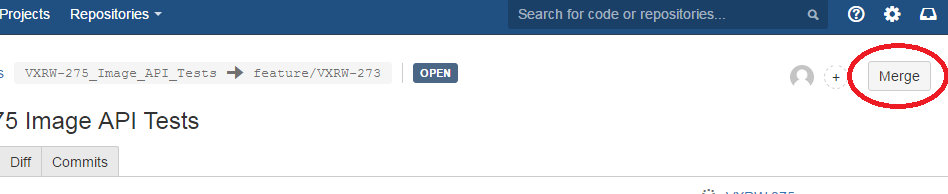
Fetch the latest from upstream, right click on repository and select “Fetch from Upstream” option

Right click on the repository in the git perspective and select rebase.

Select which branch you want to rebase from, in this case origin/master

 Click “Rebase”.

# Merge to Main Line of Development

Once you have rebased from the main line of development you know that there are only your additive changes on your branch to add, here we can use the merge button in your pull request to merge the changes into the main line of development. Navigate to your pull request and click on the merge button.