Git Lab 6
Using the SourceTree GUI Client

1 References

Some basic starter guidelines for SourceTree is available at:

https://confluence.atlassian.com/get-started-with-sourcetree

The complete knowledge base for all functionality available from SourceTree is available at:

https://support.atlassian.com/sourcetree/

2 Lab setup

Make sure you have a suitable text editor installed for your OS.

Make sure you have a valid BitBucket account.

We will be cloning a remote repo into a subfolder within the main folder where we previously created our Git repositories (in this example `G:\labs`). If are using a directory with a different name, substitute that name into all the relevant commands in this lab session.

We will be repeating all the Git operations from the previous lab in interacting with a remote repo, but this time, we will be using the SourceTree client instead of typing the commands from the Git Bash Shell.

Keep in mind that SourceTree (like any other Git GUI clients) will essentially interact with Git via the Git Bash commands that we have studied so far in previous labs. In other words, SourceTree is simply a GUI wrapper around the standard Git commands which provides a point-n-click approach to accomplish common Git functionality (instead of typing commands) and which formats the results of a Git operation in a more user-friendly form (particularly the commit history of the various branches).

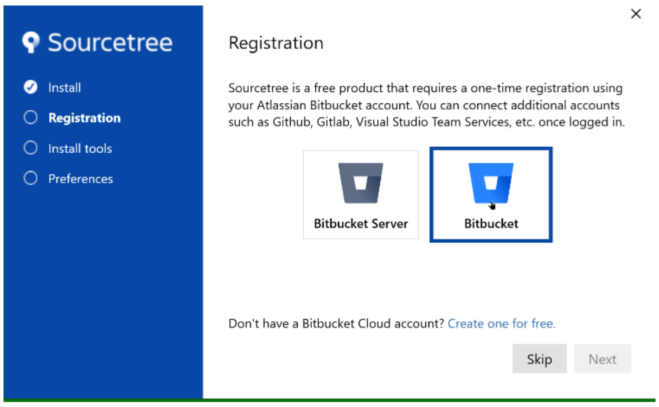
However, SourceTree DOES NOT add any extra functionality that is not already achievable via standard Git Bash commands, and sometimes you may have to issue certain more obscure commands directly in Git Bash as there is no equivalent for that in Source Tree (for e.g. `git reflog`). This is why it is more important to understand the functionality of the Git commands from the shell: once you are comfortable with them, you can learn to work with any other one of the major Git GUI clients in a very short period of time.

3 Installing SourceTree

Download SourceTree from: https://www.sourcetreeapp.com/

Before starting the installation process, ensure that you logged in to your BitBucket account on your default browser.

Double click on the Installable executable to start the installation process. You will be initially prompted on whether you wish to connect to a BitBucket or BitBucket Server service. For this lab, we will be using BitBucket Cloud, so select this:

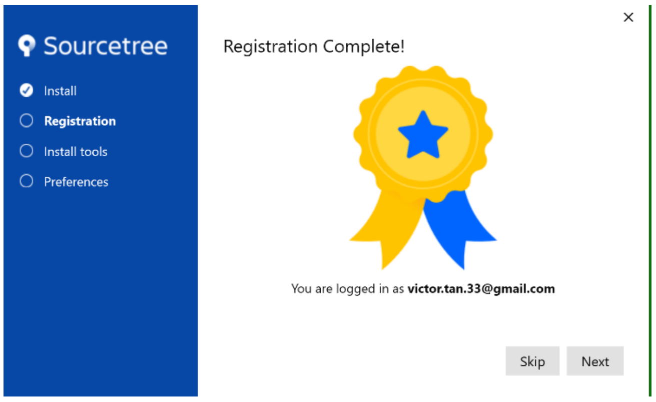


Registration

Sourcetree is a free product that requires a one-time registration using your Atlassian Bitbucket account. You can connect additional accounts such as Github, Gitlab, Visual Studio Team Services, etc. once logged in.

Don't have a Bitbucket Cloud account? Create one for free.

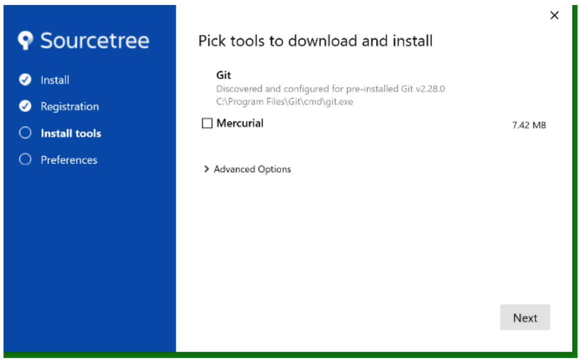
It will login to BitBucket using the current account that you are logged into on your browser. You will then be transitioned to the registration complete page.



Registration Complete!

You are logged in as **victor.tan.33@gmail.com**

You will be transitioned to a screen which identifies the tools that you need to download and install. It should identify that you already have Git installed, and there is no need to install Mercurial, so you can uncheck that checkbox and select Next.

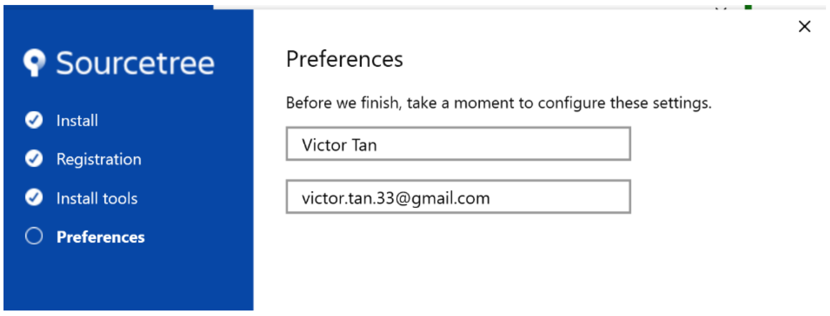


Pick tools to download and install

Git
Discovered and configured for pre-installed Git v2.28.0
C:\Program Files\Git\cmd\git.exe

☐ **Mercurial** 7.42 MB

In the preferences screen, enter the user name and email that you will use for identifying commits that you will create using Git operations issued from this client. Enter the account name and primary email for your BitBucket account:



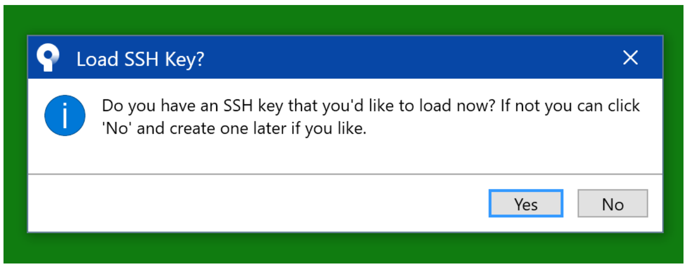
Preferences

Before we finish, take a moment to configure these settings.

Victor Tan

victor.tan.33@gmail.com

Finally, you are prompted on whether you wish to download a SSH key, and you can click No for now.

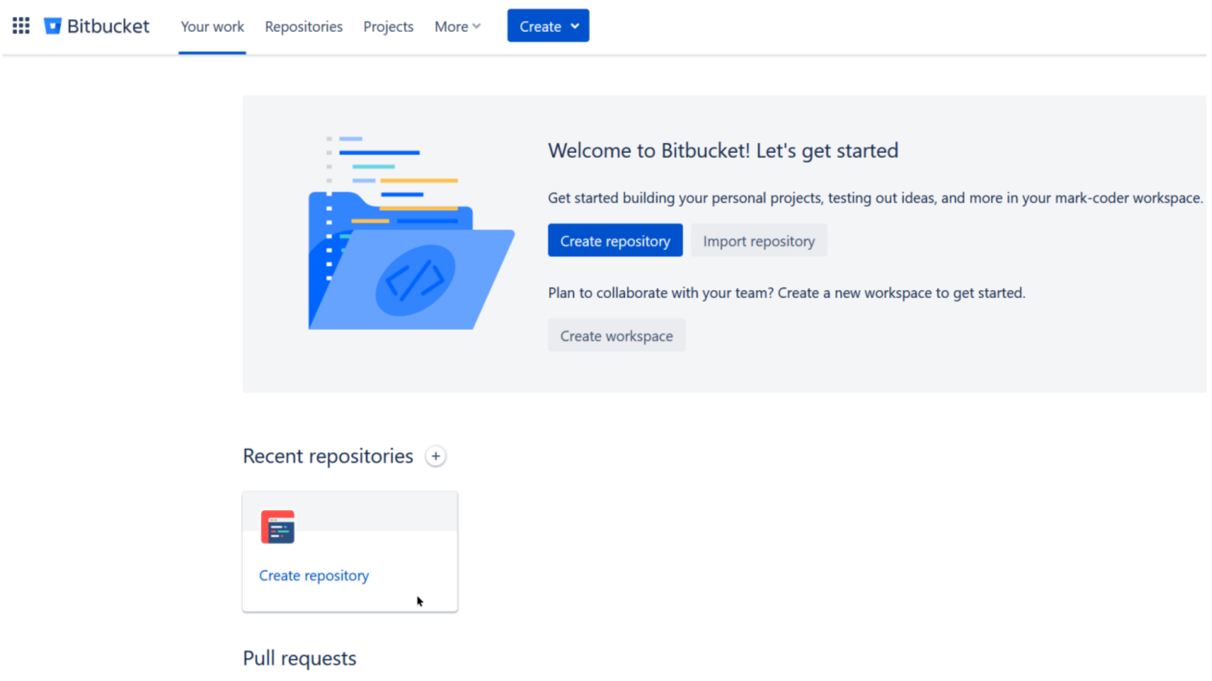


Load SSH Key?

Do you have an SSH key that you'd like to load now? If not you can click 'No' and create one later if you like.

4 Creating a private remote repo in a BitBucket account

Login to your BitBucket account. If this is a completely new account, you should see a main page that looks similar to the screen shot below.



Welcome to Bitbucket! Let's get started

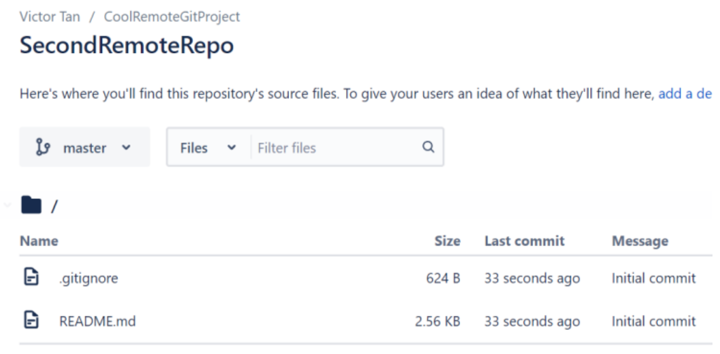
Get started building your personal projects, testing out ideas, and more in your mark-coder workspace.

Plan to collaborate with your team? Create a new workspace to get started.

We can proceed to create a new repository in this account.

Enter the values for the following fields.
Project Name: `CoolRemoteGitProject`
Repository Name: `SecondRemoteRepo`

Accept the defaults for the other fields, and click Create Repository. You will be transitioned to the Source view for the newly created repo, where you can view all the content in the project folder.



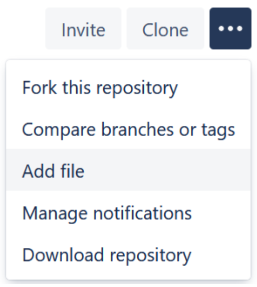
Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a de

Name	Size	Last commit	Message
.gitignore	624 B	33 seconds ago	Initial commit
README.md	2.56 KB	33 seconds ago	Initial commit

5 Adding content in new commits and new branches

We will now create a new file and add some content to it and commit these changes.

Select Add file from the vertical three dots menu option.



At the top, give the file the name: `countries.txt`

In the file edit tab, enter this new content, making sure to include an empty line at the bottom:

```
Malaysia
Singapore
```

Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

```
Initialized countries with some content
```

You are now transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

Click on Source in the left pane to return to the Source view. You should be able to see `countries.txt` in the directory listing. Click on this entry and click on Edit button to start editing in the file edit tab. Enter more new content as shown below, making sure to include an empty line at the bottom:

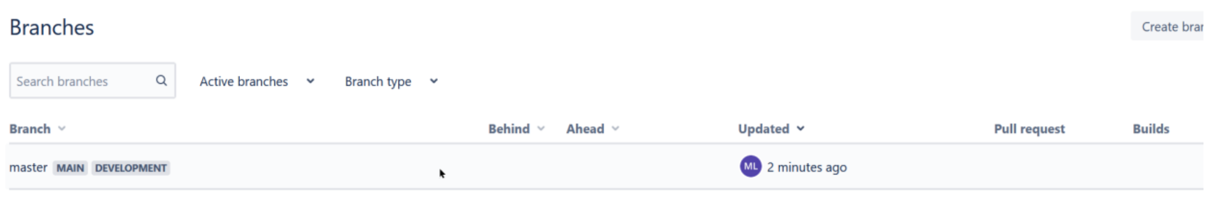
```
Brunei
Thailand
```

Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

```
Added more content to countries
```

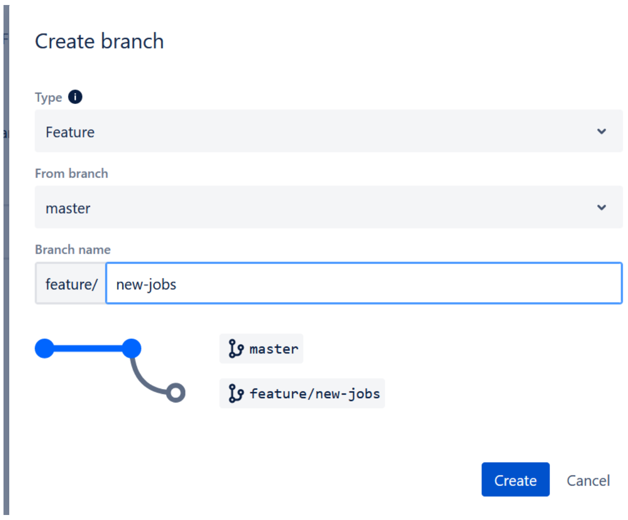
Once again, you are transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

Click on Branches in the left pane to transition to the branches view, where you can see a listing of all branches, including the current active one, master, which is designated as the main development branch.

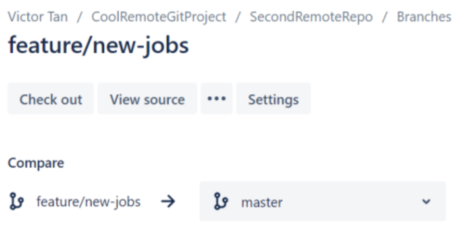


Click on Create Branch in the upper right hand corner. The Type dropdown in the dialog box offers 4 categories to classify branches, as well as Other for branches that don't fall into any category

Select a Feature type, and for the new branch name, enter: `new-jobs` and click Create.

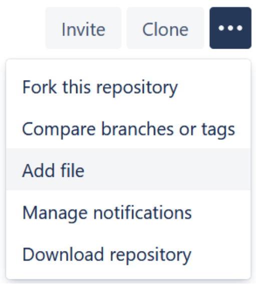


In the view for the newly created branch, click on the View Source button to transition back to the Source view



In the Source view, we will now be viewing the contents of the project folder corresponding to the newly created branch: `feature/new-jobs`, which at this point of time will be exactly identical to `master` as we have not added any new commits to it yet.

Let's proceed to add a new file to this branch, add some content to it and commit these changes. Select Add file from the vertical three dots menu option.



At the top, give the file the name: `jobs.txt`

The file view should clearly indicate you are in the `feature/new-jobs` branch. In the file edit tab, enter this new content, making sure to include an empty line at the bottom:

```
developer
project manager
```

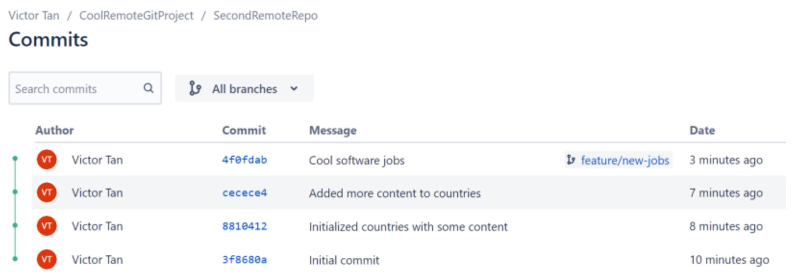
Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

```
Cool software jobs
```

You are again transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

Click on Source in the left pane to return to the Source view. By default, the Source view shows the contents of the latest commit in the master branch, so you won't be able to see `jobs.txt` (as it was not created in this branch). Select `feature/new-jobs` from the branch drop down menu to switch to this branch, and you should now see `jobs.txt`

Click on Commits in the left pane to switch to the commits view, where you will be shown by default the commit history of all branches in the repo. Select between the branches available from the drop-down menu at the top to view the commit history of a specific branch (at the moment, either `master` or `feature/new-jobs`)


Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Commits

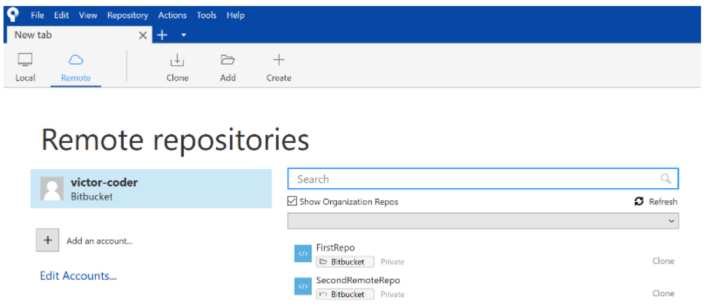
Author	Commit	Message		Date
Victor Tan	4f0fdab	Cool software jobs	feature/new-jobs	3 minutes ago
Victor Tan	cecece4	Added more content to countries		7 minutes ago
Victor Tan	8810412	Initialized countries with some content		8 minutes ago
Victor Tan	3f8680a	Initial commit		10 minutes ago

Switch back to the Source view, and ensure that master is selected as the current / active branch. We are now ready to clone this remote repo to a local repo on our local machines.

6 Cloning a remote repo

Start up SourceTree from its quickstart icon or by locating it in the Search box in Windows. You will be prompted to go through the same sequence of steps in the initial installation process.

Click on the Remote button (cloud icon) to get the main Remote repos view and refresh it if necessary to view the repositories in the active BitBucket cloud account. You should be able to see the remote repo `SecondRemoteRepo` that you created earlier.



Remote repositories

victor-coder
Bitbucket

Add an account...

Edit Accounts...

Search

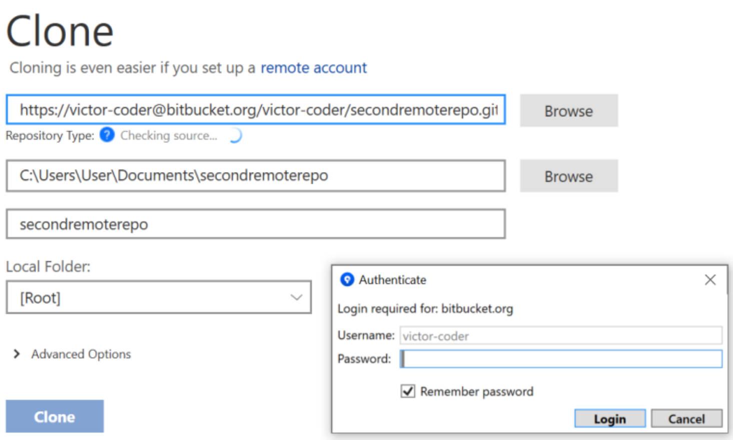
Show Organization Repos

Refresh

FirstRepo Bitbucket Private Clone

SecondRemoteRepo Bitbucket Private Clone

Select this entry and click clone.
You will be transitioned to a screen with fields that provide details on the clone operation (source, destination, name, etc) as well as an authenticate dialog box.



Clone

Cloning is even easier if you set up a remote account

https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

Repository Type: Checking source...

C:\Users\User\Documents\secondremoterepo

secondremoterepo

Local Folder: [Root]

Authenticate

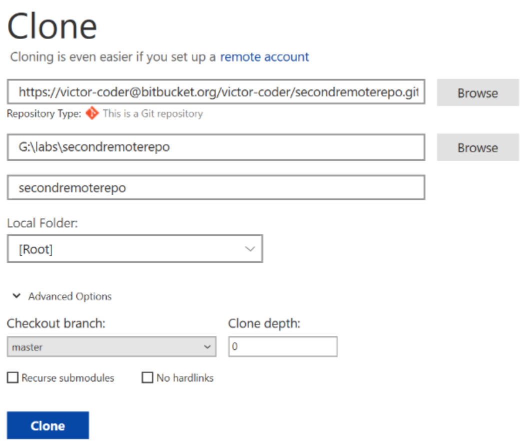
Login required for: bitbucket.org

Username: victor-coder

Password:

Remember password

Enter the app password that you obtained from the previous lab for this account. If you have forgotten it, you will need to go back to that session and repeat the operation to obtain an app password. Then fill in the fields with a suitable location on your local drive where you want to clone the remote repo, along with the checkout branch (`master` by default)



Clone

Cloning is even easier if you set up a remote account

https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

Repository Type: This is a Git repository

G:\labs\secondremoterepo

secondremoterepo

Local Folder: [Root]

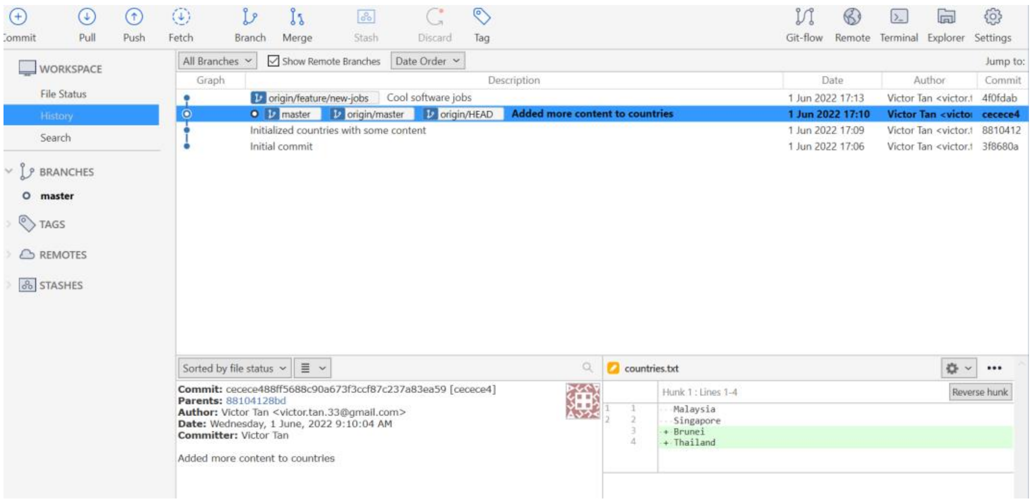
Advanced Options

Checkout branch: master

Clone depth: 0

☐ Recurse submodules ☐ No hardlinks

Click Clone and the clone operation will proceed and after a while you will be transitioned back to the main view for this remote repo. There are 3 possible views for the open workspace: currently you are viewing the History view (equivalent to a `git log --all`). You can switch to the other two views (File Status, Search) from the sidebar.



Graph	Description	Date	Author	Commit
	origin/feature/new-jobs Cool software jobs	1 Jun 2022 17:13	Victor Tan <victor.t	4f0fdab
	master origin/master origin/HEAD **Added more content to countries**	**1 Jun 2022 17:10**	**Victor Tan <victo**	**cecece4**
	Initialized countries with some content	1 Jun 2022 17:09	Victor Tan <victor.t	8810412
	Initial commit	1 Jun 2022 17:06	Victor Tan <victor.t	3f8680a

Commit: cecece488ff5688c90a673f3ccf87c237a83ea59 [cecece4]
Parents: 88104128bd
Author: Victor Tan <victor.tan.33@gmail.com>
Date: Wednesday, 1 June, 2022 9:10:04 AM
Committer: Victor Tan

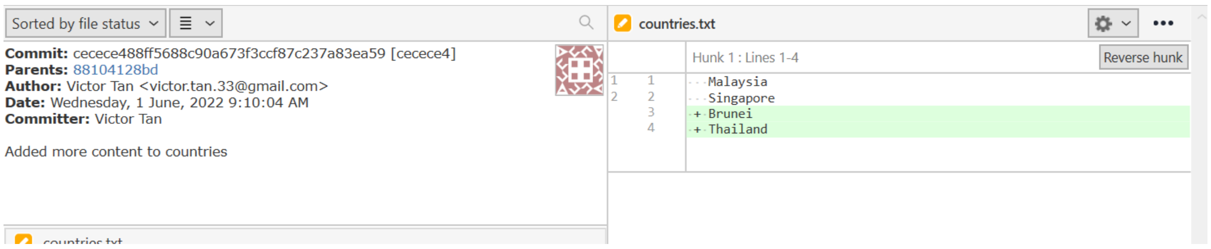
Added more content to countries

countries.txt

```
Hunk 1 : Lines 1-4
  Malaysia
  Singapore
+ Brunei
+ Thailand
```

7 Inspecting working directory content of commits in the commit history listing

This view provides the commit history for the current branch or all branches (which you can select from the branch drop down box) as well as the remote tracking branches (`origin/xxx`) which are shown by default. You can click on any of the commits in the commit history and the info for the commit as well as the contents of the files in that commit and its diff with its parent (previous) commit is shown.



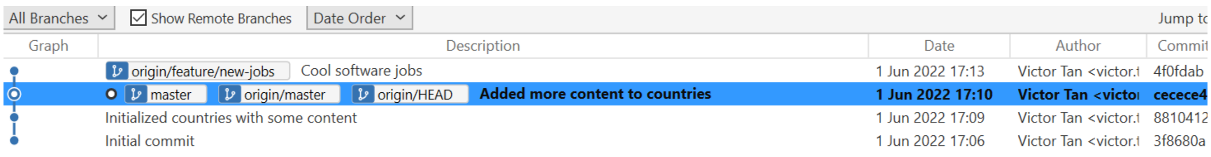
Commit: cecece488ff5688c90a673f3ccf87c237a83ea59 [cecece4]
Parents: 88104128bd
Author: Victor Tan <victor.tan.33@gmail.com>
Date: Wednesday, 1 June, 2022 9:10:04 AM
Committer: Victor Tan

Added more content to countries

countries.txt

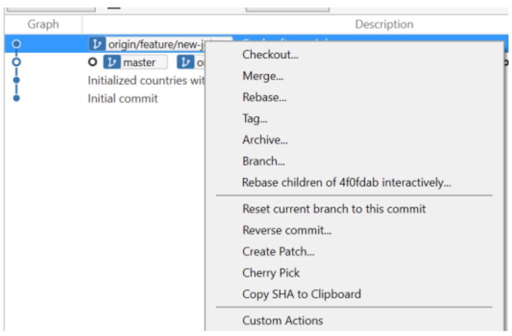
```
Hunk 1 : Lines 1-4
  Malaysia
  Singapore
+ Brunei
+ Thailand
```

Right now, with the All Branches option, selected we can see that we have a local master tracking the remote master, but we only have a remote `feature/new-jobs` pointing to the single commit `Cool software jobs` in it. There is no local branch counterpart for it yet.

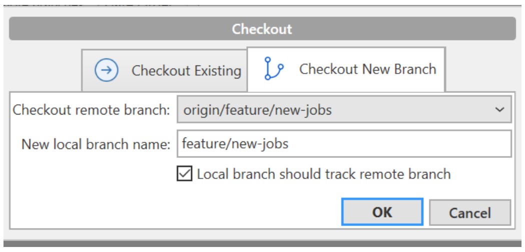


Graph	Description	Date	Author	Commit
	origin/feature/new-jobs Cool software jobs	1 Jun 2022 17:13	Victor Tan <victor.t	4f0fdab
	master origin/master origin/HEAD **Added more content to countries**	**1 Jun 2022 17:10**	**Victor Tan <victor**	**cecece4**
	Initialized countries with some content	1 Jun 2022 17:09	Victor Tan <victor.t	8810412
	Initial commit	1 Jun 2022 17:06	Victor Tan <victor.t	3f8680a

Right click on the remote tracking branch `origin/feature/new-jobs` and select Checkout from the context menu.



The dialog box appears from which you can then select Ok.

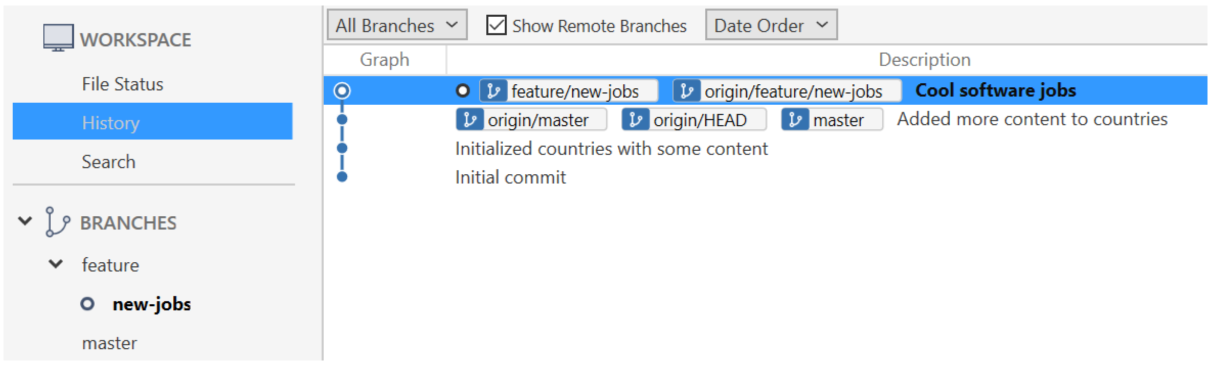


Checkout remote branch: origin/feature/new-jobs

New local branch name: feature/new-jobs

☑ Local branch should track remote branch

When this is done, you should now see a new local branch with this name in the side bar, and also in the commit history listing.



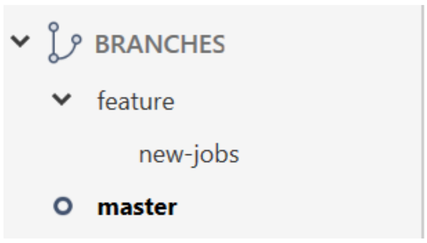
feature/new-jobs origin/feature/new-jobs **Cool software jobs**

origin/master origin/HEAD master Added more content to countries

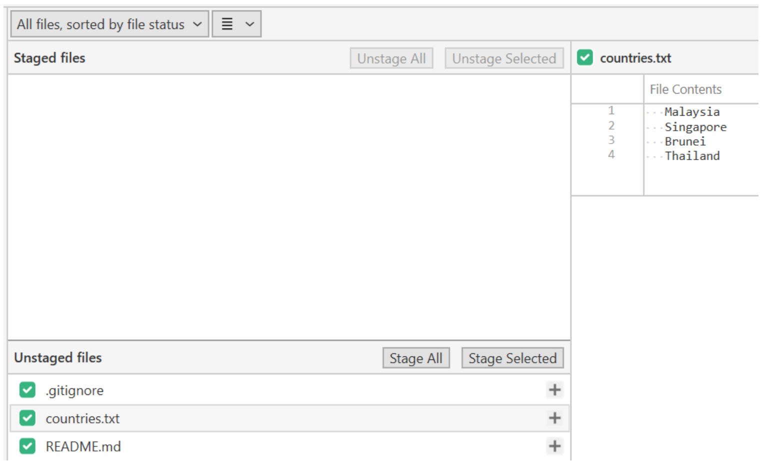
Initialized countries with some content

Initial commit

You can double click on any commit in the history listing above in order to view its contents. It is the equivalent of performing `git checkout` *`commit-hash`* from the command line. Double click on the commit with the description: `Added more content to countries`. Since the local master is pointing to this commit, you will see that highlighted in the side bar:

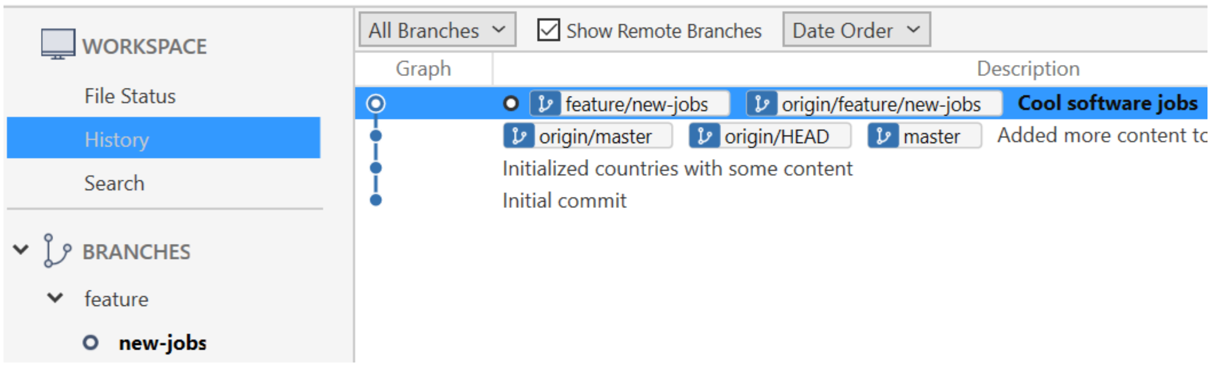


Click on the File Status option in the Workspace entry in the side bar to switch the main view to display the files in the working directory (the project folder) corresponding to the contents of the commit being referenced. Remember to select the option `All files, stored by file status` from the drop down list at the top. You can then click on any of the files in the list to view their contents in the second window.



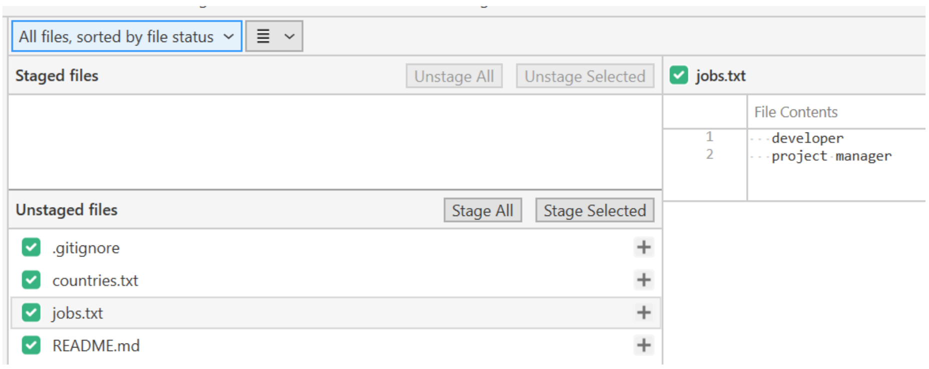
```
Malaysia
Singapore
Brunei
Thailand
```

Return back to the Workspace History view, and then double click on the commit that `feature/new-jobs` is referencing. Notice now that the Branches view in the side bar now indicates that `new-jobs` is the current / active branch



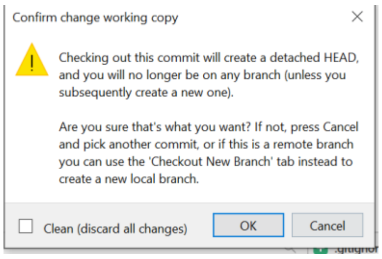
Cool software jobs
Added more content tc
Initialized countries with some content
Initial commit

Switching back to the Workspace File Status view, you will be able to see the new `jobs.txt` that was added to this branch while we were working with the remote repo via the browser UI.



```
developer
project manager
```

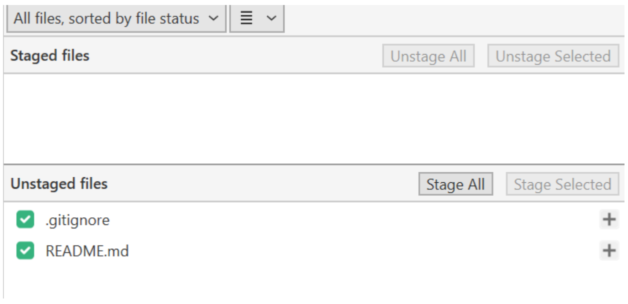
Return back to the Workspace History view, and then double click on the `Initial commit`. Since this is equivalent of performing a `git checkout` on a specific commit rather than a branch, this will move HEAD into a detached state (as we have already seen in a previous lab at the command line) and SourceTree will issue the same warning here as well. Click ok to continue.



Checking out this commit will create a detached HEAD, and you will no longer be on any branch (unless you subsequently create a new one).

Are you sure that's what you want? If not, press Cancel and pick another commit, or if this is a remote branch you can use the 'Checkout New Branch' tab instead to create a new local branch.

You can then switch again to the Workspace File Status view, where you will see the two files that were automatically created by BitBucket and placed in the root commit:



As the commit history of your repository grows over time, you can use this technique (double click on any commit in the history listing) to examine the contents of the working directory stored in that particular commit.

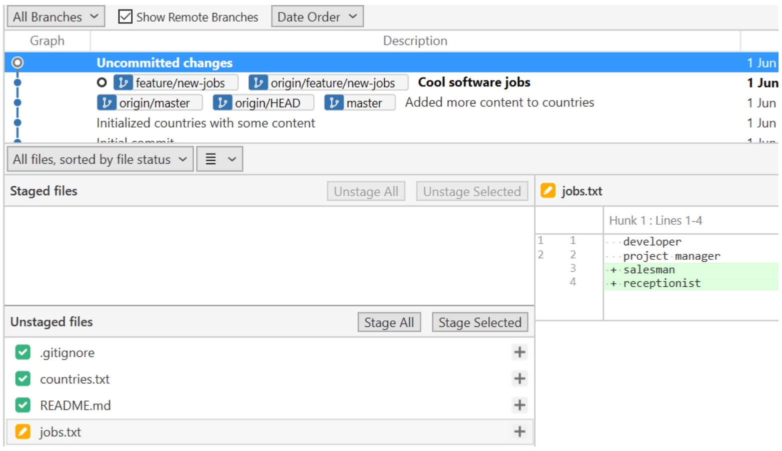
8 Making local commits and pushing to the remote repo

Go back to the Workspace History view, and switch to the `feature/new-jobs` branch. If you double click on the `jobs.txt` file name in the bottom tab, this should open up that file for editing in the default configured editor for that file type in your system (typically this will be Notepad or Notepad++ for files with extension *.txt). Alternatively, you can just use whatever text / code editor or IDE of your choice to open any file you want in the Git project folder, and SourceTree will automatically pick up the changes.

Add the following content to the end of `jobs.txt`, ensuring you leave a new line at the bottom:

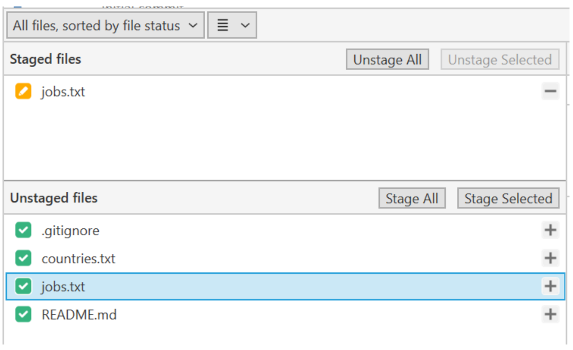
```
salesman
receptionist
```

Once you have saved in the editor of your choice, SourceTree will pick up this changes and indicate this in the Workspace History view and File Status view as well:

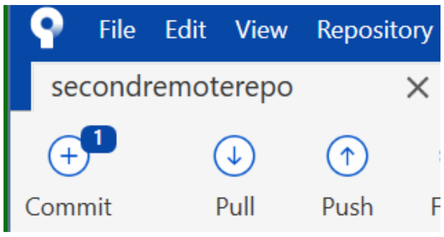


```
Hunk 1 : Lines 1-4
1  1    developer
2  2    project manager
   3  + salesman
   4  + receptionist
```

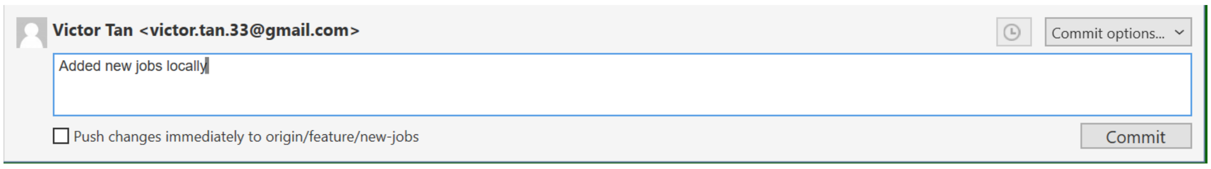
As you can see, it marks the files in the working directory that have unstaged modifications in them with a yellow tick and provides a diff of these changes against the previous committed state (the equivalent of doing `git diff`).
If you click the + symbol next to the marked file, this stages that specific file. If you have changes in all the files in the working directory, you can click Stage All to stage all of them. The staged files should be shown appropriately in the correct section:



To unstage a staged change, simply click on the - symbol for that related file. You can then stage it again by clicking on the + symbol.
To commit all the staged changes into a new commit, click on the Commit button at the upper left hand corner of the app.



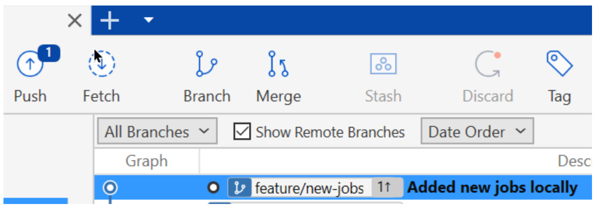
Then type in the commit message: `Added new jobs locally` in the text box at the bottom and then click Commit. Notice that you have the option to immediately push this new commit to the corresponding upstream branch at the same time, but we will skip that for now.


Victor Tan <victor.tan.33@gmail.com>

Added new jobs locally

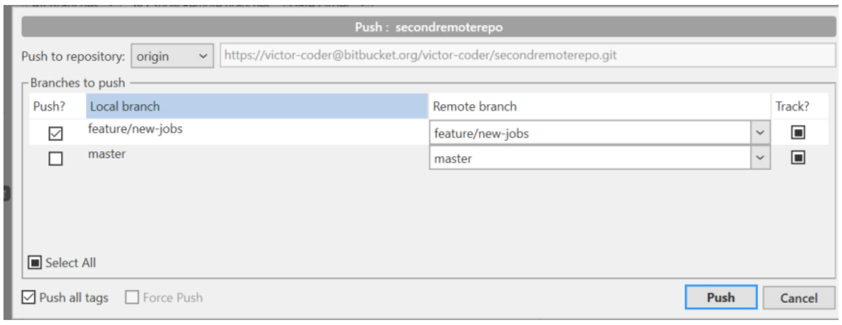
☐ Push changes immediately to origin/feature/new-jobs

After this has been done, notice that the new commit appears in the commit history with a number 1 next to an arrow. This indicates that the local new-jobs branch is now ahead of its matching upstream branch by 1 commit. The Push button will also feature the same number next to it as a reminder / prompt to you to push these new commits to the upstream branch ASAP.



feature/new-jobs 1↑ Added new jobs locally

Go ahead and click on the Push button. A confirmation dialog box showing the destination remote repo (the short name `origin` as well as the complete URL) as well as the branches affected by the push operation will be shown. Click Push to proceed.


Push : secondremoterepo

Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

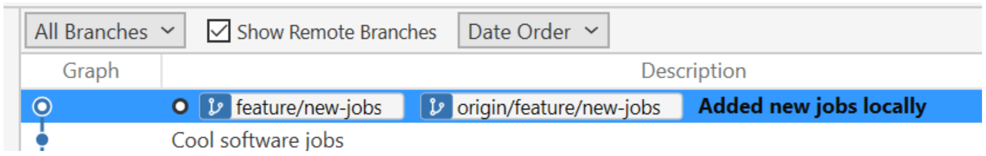
Branches to push

Push?	Local branch	Remote branch	Track?
☑	feature/new-jobs	feature/new-jobs	▣
☐	master	master	▣

▣ Select All

☑ Push all tags ☐ Force Push

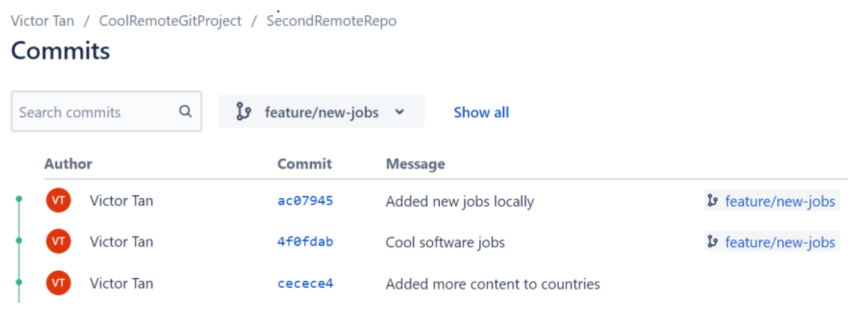
When you are done, the History listing should now show the remote tracking branch and the local branch are in sync on the latest commit.



feature/new-jobs origin/feature/new-jobs Added new jobs locally

Cool software jobs

You can double check this as well in the Commit view in the browser.


Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Commits

Search commits | feature/new-jobs | Show all

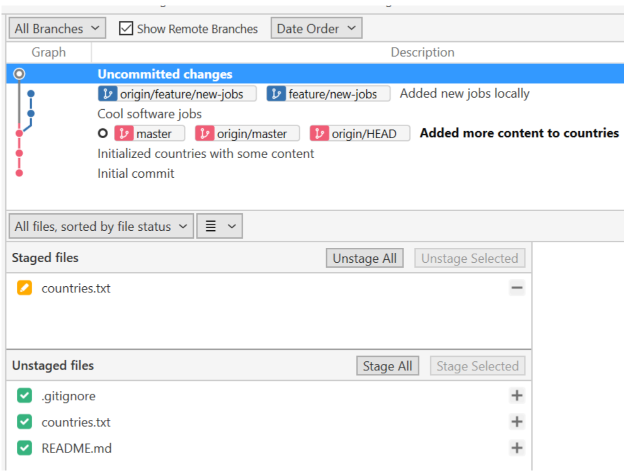
Author	Commit	Message	
Victor Tan	ac07945	Added new jobs locally	feature/new-jobs
Victor Tan	4f0fdab	Cool software jobs	feature/new-jobs
Victor Tan	cecece4	Added more content to countries	

We will now create a new local branch. Back in SourceTree, double click on the master branch to make that the current / active branch.

Add the following content to the end of `countries.txt`, ensuring you leave a new line at the bottom:

```
Vietnam
Philippines
```

Stage this file by clicking on the + symbol.



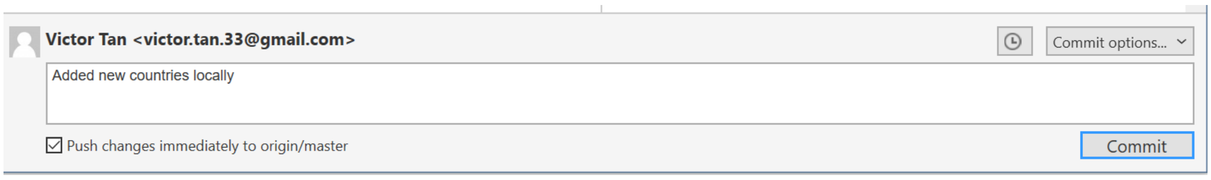
Uncommitted changes
origin/feature/new-jobs feature/new-jobs Added new jobs locally
Cool software jobs
master origin/master origin/HEAD Added more content to countries
Initialized countries with some content
Initial commit

countries.txt

.gitignore
countries.txt
README.md

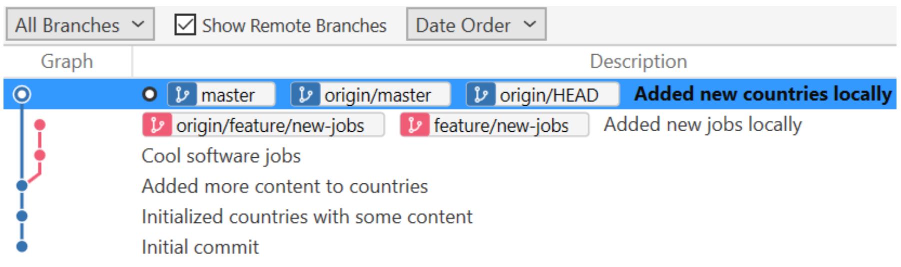
Finally, click on the Commit button to create the new commit.

Then type in the commit message: `Added new countries locally` in the text box at the bottom. This time, check the option to immediately push this new commit to the corresponding upstream branch at the same time. Finally, click Commit.


Victor Tan <victor.tan.33@gmail.com>

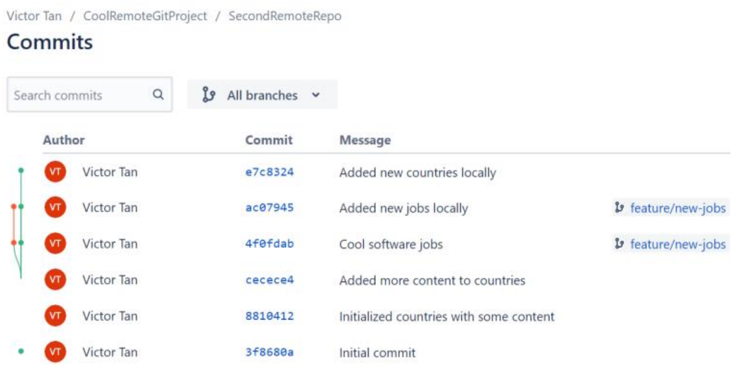
Added new countries locally
☑ Push changes immediately to origin/master

When you are done, the History listing should now show the remote tracking master and the local master are in sync on the latest commit. You should also be able to clearly see the divergence between the two branches in the repository (`master` and `feature/new-jobs`).



master | origin/master | origin/HEAD Added new countries locally
origin/feature/new-jobs | feature/new-jobs Added new jobs locally
Cool software jobs
Added more content to countries
Initialized countries with some content
Initial commit

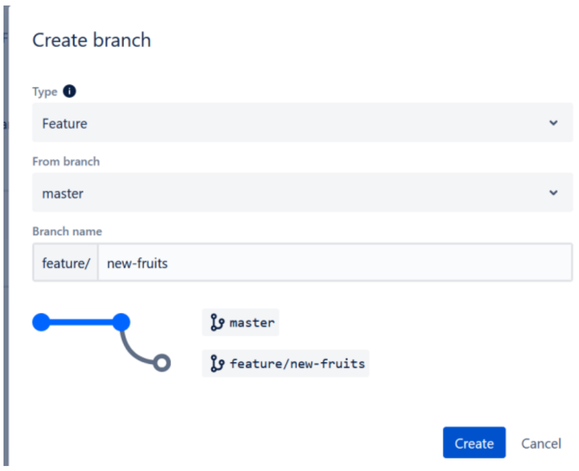
You can double check this as well in the Commits view in the browser.



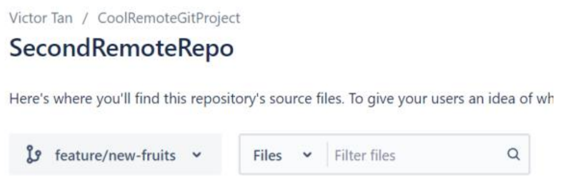
Author	Commit	Message	
Victor Tan	e7c8324	Added new countries locally	
Victor Tan	ac07945	Added new jobs locally	feature/new-jobs
Victor Tan	4f0fdab	Cool software jobs	feature/new-jobs
Victor Tan	cecece4	Added more content to countries	
Victor Tan	8810412	Initialized countries with some content	
Victor Tan	3f8680a	Initial commit	

9 Fetching new branches from remote to local repo

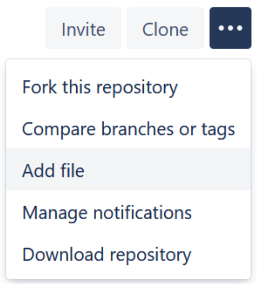
Back in the remote repo, switch to the Branches view and click on the Create Branch button. In the Dialog box, create a new feature branch named: `new-fruits` which starts from `master`



In the view for this newly created branch, select View Source.


Victor Tan / CoolRemoteGitProject
SecondRemoteRepo
Here's where you'll find this repository's source files. To give your users an idea of wh
feature/new-fruits
Files
Filter files

Let's proceed to add a new file to this branch, add some content to it and commit these changes. Select Add file from the vertical three dots menu option.



At the top, give the file the name: `fruits.txt`

The file view should clearly indicate you are in the `feature/new-fruits` branch. In the file edit tab, enter this new content, making sure to include an empty line at the bottom:

```
apple
orange
```

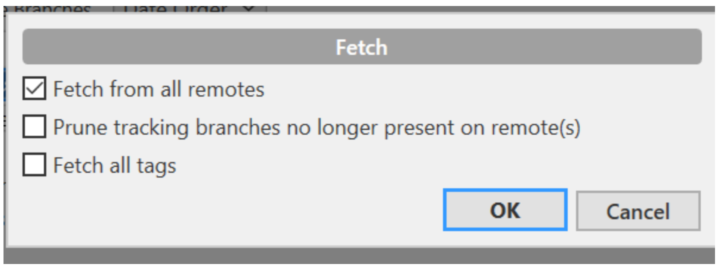
Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

```
Some new fruits initialized
```

You are again transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

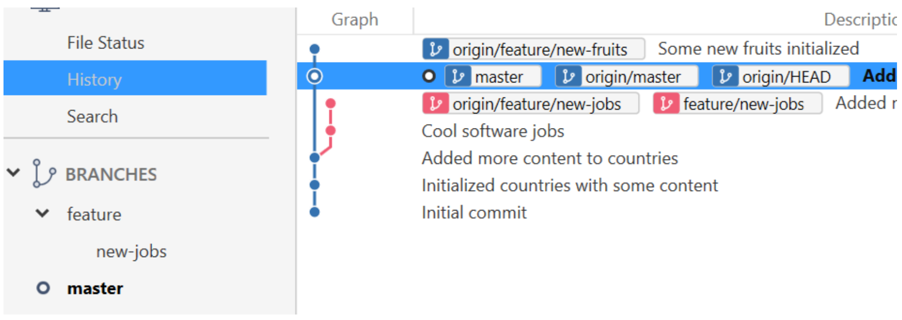
Back in the SourceTree client, check in the Workspace History view and notice that there is still no indication yet of this new upstream branch.

Click on the Fetch button from the main menu. Select Ok from the dialog box present.



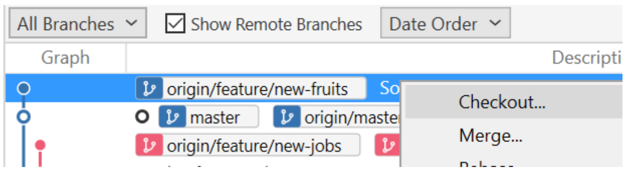
☑ Fetch from all remotes
☐ Prune tracking branches no longer present on remote(s)
☐ Fetch all tags

You should now be able to see the new upstream branch, although there is no local branch counterpart created for it yet.



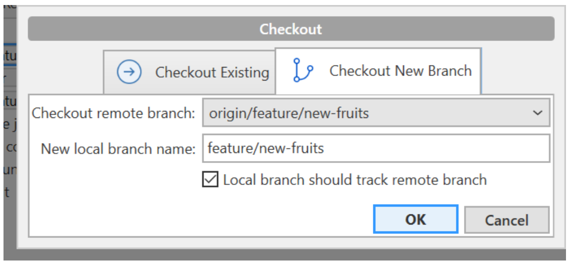
origin/feature/new-fruits Some new fruits initialized
master origin/master origin/HEAD Add
origin/feature/new-jobs feature/new-jobs Added r
Cool software jobs
Added more content to countries
Initialized countries with some content
Initial commit

As we have seen before previously, to create the local counterpart for the new upstream, we right click on the remote tracking branch entry in the listing and select Checkout from the context menu.



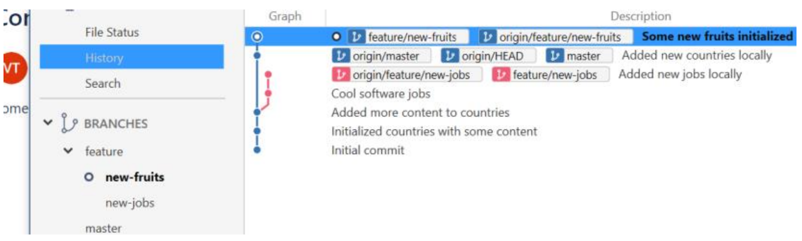
origin/feature/new-fruits So
master origin/maste
origin/feature/new-jobs

Click ok in the Dialog box that appears.



Checkout remote branch: origin/feature/new-fruits
New local branch name: feature/new-fruits
☑ Local branch should track remote branch

We should now see the new local branch that is tracking its upstream counterpart in both the Branches view in the side bar and in the main history listing.



feature/new-fruits origin/feature/new-fruits Some new fruits initialized
origin/master origin/HEAD master Added new countries locally
origin/feature/new-jobs feature/new-jobs Added new jobs locally
Cool software jobs
Added more content to countries
Initialized countries with some content
Initial commit

10 Merging changes from upstream to local branches

In the browser, return back to the Source view for the `master` branch, and select `countries.txt`. Click on Edit and add this additional content, making sure to include an empty line at the bottom:

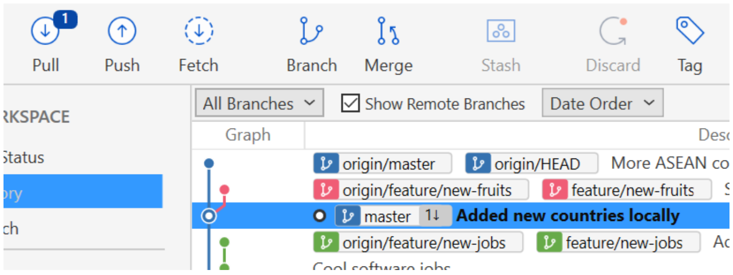
```
Cambodia
Myanmar
```

Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

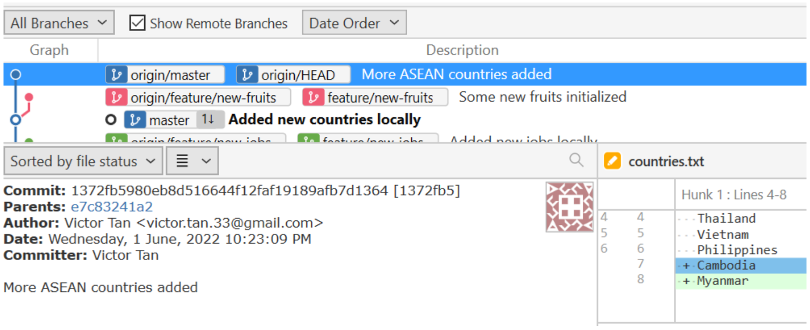
```
More ASEAN countries added
```

You are now transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

Back in SourceTree, click on Fetch to obtain the latest status of the remote repo. Click on OK in the dialog box that appears.
If you double click on the master branch in the history listing, you will see a 1 with a down arrow key next to it indicating that master is outdated by 1 commit and that we can make it equivalent to the upstream master with a `git pull` operation



If you want to see the diff between the current master branch and the latest commit in the upstream counterpart, just click on the relevant entry in the history listing and inspect the diff in the lower right hand window.

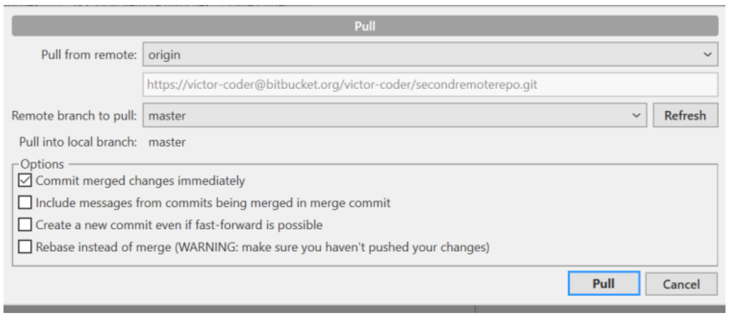


Commit: 1372fb5980eb8d516644f12faf19189afb7d1364 [1372fb5]
Parents: e7c83241a2
Author: Victor Tan <victor.tan.33@gmail.com>
Date: Wednesday, 1 June, 2022 10:23:09 PM
Committer: Victor Tan

More ASEAN countries added

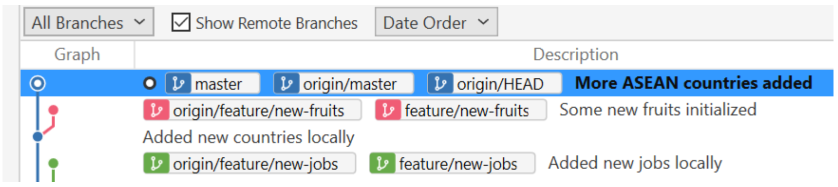
```
4   4     Thailand
5   5     Vietnam
6   6     Philippines
    7   + Cambodia
    8   + Myanmar
```

Finally, when you are ready to update the local master by merging in the latest commit from the upstream master, click on Pull. Click ok in the dialog box that appears.



Pull from remote: origin
https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git
Remote branch to pull: master
Pull into local branch: master
Options
☑ Commit merged changes immediately
☐ Include messages from commits being merged in merge commit
☐ Create a new commit even if fast-forward is possible
☐ Rebase instead of merge (WARNING: make sure you haven't pushed your changes)

You should now see both the local and upstream master in sync on the latest new commit.



master origin/master origin/HEAD **More ASEAN countries added**
origin/feature/new-fruits feature/new-fruits Some new fruits initialized
Added new countries locally
origin/feature/new-jobs feature/new-jobs Added new jobs locally

In the browser, return back to the Source view for the `feature/new-fruits` branch, and select `fruits.txt`. Click on Edit and add this additional content, making sure to include an empty line at the bottom:

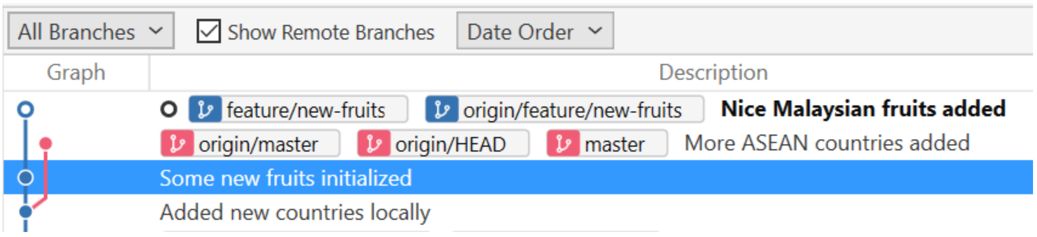
```
mango
durian
```

Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

```
Nice Malaysian fruits added
```

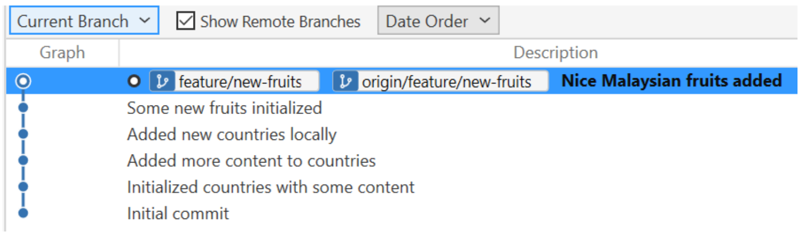
You are now transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

Back in the SourceTree client, double click to select on the `feature/new-fruits` branch. Without performing a prior Fetch, we will immediately click on Pull. Click Pull in the dialog box that appears.
When the pull operation is complete, you will see the new commit in the local `feature/new-fruits` branch

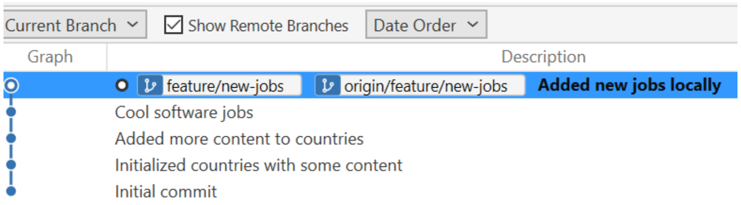


feature/new-fruits origin/feature/new-fruits **Nice Malaysian fruits added**
origin/master origin/HEAD master More ASEAN countries added
Some new fruits initialized
Added new countries locally

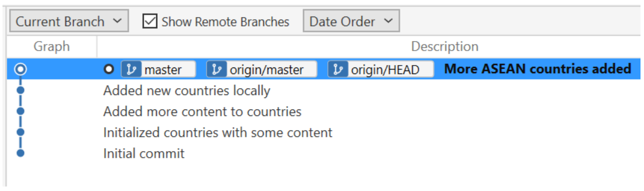
At this point of time, we have histories for 3 branches being shown simultaneously in the history view. If the number of branches and their commits increase significantly, it might become quite difficult to view. In that case, you can use switch to a selected branch by double clicking on it and then selecting the Show current branch entry in the drop down options. For e.g.



feature/new-fruits origin/feature/new-fruits **Nice Malaysian fruits added**
Some new fruits initialized
Added new countries locally
Added more content to countries
Initialized countries with some content
Initial commit



feature/new-jobs | origin/feature/new-jobs | **Added new jobs locally**
Cool software jobs
Added more content to countries
Initialized countries with some content
Initial commit



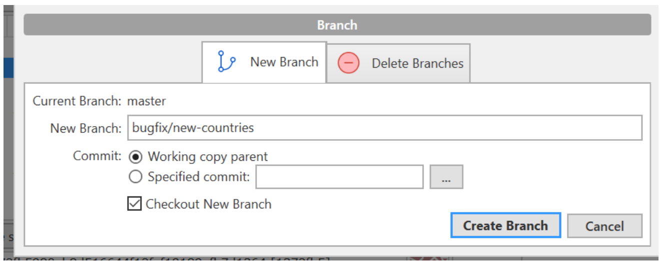
master | origin/master | origin/HEAD | **More ASEAN countries added**
Added new countries locally
Added more content to countries
Initialized countries with some content
Initial commit

Notice that in all listings, we can see that the local branch is in sync with their upstream counterpart in referencing the latest commit for that particular branch.

11 Pushing a new local branch to the remote repo

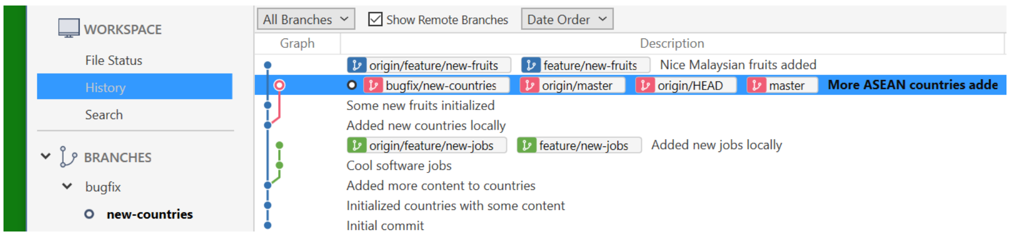
We will create a new local branch, which we will classify as a bugfix (to distinguish it from the previous 2 branches that we classified as feature):

Switch back to master in the History view, then click the Branch button. Enter the name for this new branch in the dialog box: `bugfix/new-countries` and select Create Branch.



Current Branch: master
New Branch: bugfix/new-countries
Commit: [✓ Working copy parent] Specified commit:
[✓ Checkout New Branch]

You should see this new branch highlighted in both the Branches side panel view and main History view.



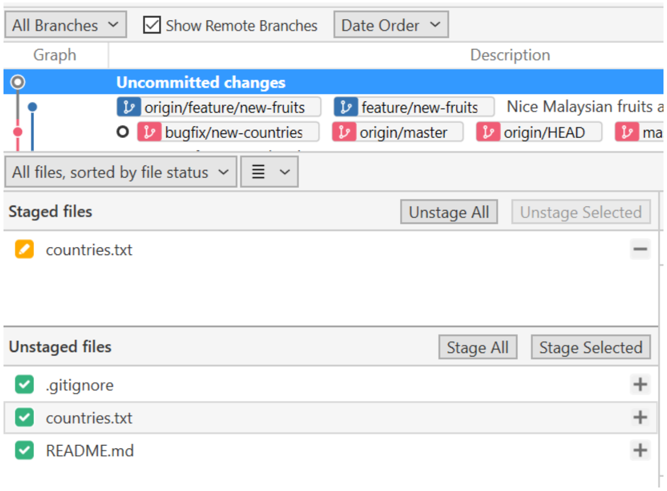
origin/feature/new-fruits | feature/new-fruits | Nice Malaysian fruits added
bugfix/new-countries | origin/master | origin/HEAD | master | **More ASEAN countries adde**
Some new fruits initialized
Added new countries locally
origin/feature/new-jobs | feature/new-jobs | Added new jobs locally
Cool software jobs
Added more content to countries
Initialized countries with some content
Initial commit

Just like before, double click on the `countries.txt` file in either the Workspace File Status or History view to open this file for editing in the default system editor, or simply open the file directly using the editor or IDE of choice.

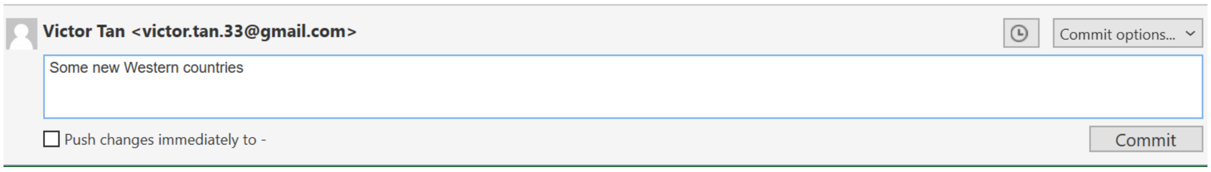
Add the following content to the end of `countries.txt`, ensuring you leave a new line at the bottom:

```
Australia
New Zealand
```

As we have seen before, click on the + symbol for `countries.txt` to move it into the staged section:

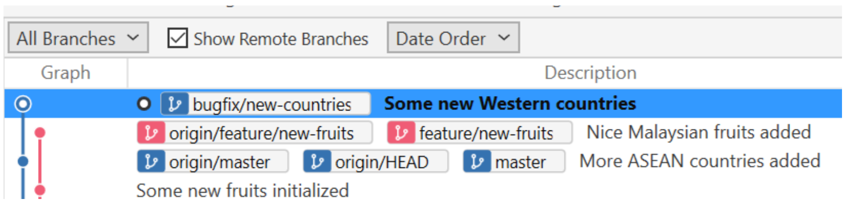


Then click on Commit to add in the first commit for this new branch. In the dialog box at the bottom, provide this message for the new commit: `Some new Western countries` and click Commit.


Victor Tan <victor.tan.33@gmail.com>

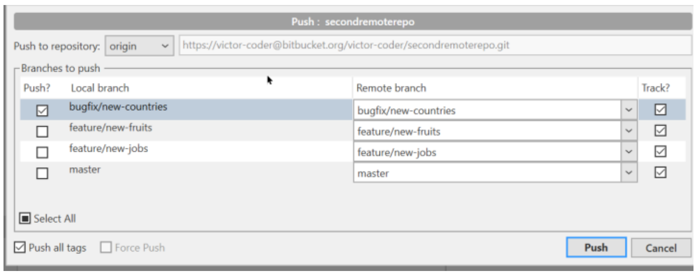
Some new Western countries

The new commit and branch will appear in the History view listing, but without a remote tracking branch as we have not pushed it yet to the remote repo.


All Branches
Show Remote Branches
Date Order
Graph
Description
bugfix/new-countries
Some new Western countries
origin/feature/new-fruits
feature/new-fruits
Nice Malaysian fruits added
origin/master
origin/HEAD
master
More ASEAN countries added
Some new fruits initialized

Ensure this new branch is selected and click the Push button. A dialog box will appear giving you a chance to provide a name of the new upstream branch that you are going to push the new local branch to. As usual, we will keep to the convention of having both branches have identical names, particularly when we want our local branch to track its upstream counterpart.

Provide the remote branch field with the same name: `bugfix/new-countries` and click on the checkbox for that entry. Then click on Push.


Push : secondremoterepo

Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

Branches to push

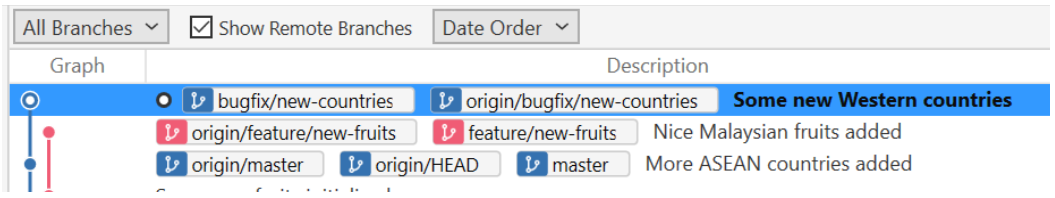
Push?	Local branch	Remote branch	Track?
✓	bugfix/new-countries	bugfix/new-countries	✓
☐	feature/new-fruits	feature/new-fruits	✓
☐	feature/new-jobs	feature/new-jobs	✓
☐	master	master	✓

■ Select All

✓ Push all tags ☐ Force Push

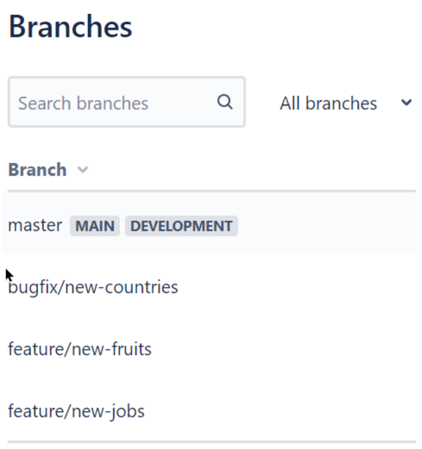
Push Cancel

You should now see a new remote tracking branch for this new local branch in the History view listing.


All Branches ✓ Show Remote Branches Date Order

Graph	Description
	bugfix/new-countries origin/bugfix/new-countries **Some new Western countries**
	origin/feature/new-fruits feature/new-fruits Nice Malaysian fruits added
	origin/master origin/HEAD master More ASEAN countries added

Head back to the browser and check the Branches view and list all branches to verify that a new upstream branch has been created.


Branches

Search branches All branches

Branch

master MAIN DEVELOPMENT

bugfix/new-countries

feature/new-fruits

feature/new-jobs

Switch to the Source view, switch to the new `bugfix/new-countries` branch and click on `countries.txt` to verify that it contains the latest content that we added locally.

12 Resolving merge conflicts in pushing and pulling changes

So far, all the merging of content that resulted from a `git push` and `git pull` have been simple fast forward merges which do not require user intervention. However, as we have seen before in a previous lab where we studied merging in detail, occasionally merge conflicts will occur which will then require manual resolution. We will see how to repeat this process here.

In the browser, return back to the Source view for the `bugfix/new-countries` branch, and select `countries.txt`. Click on Edit and add this additional content, making sure to include an empty line at the bottom:

```
Egypt
```

```
Kuwait
```

Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

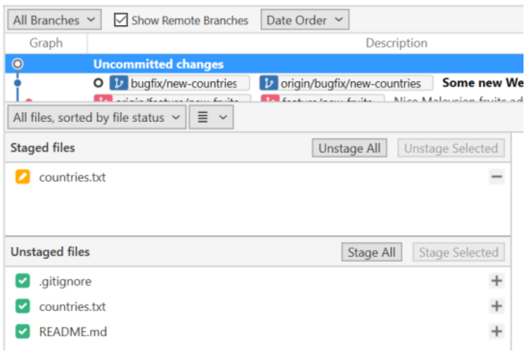
```
Some really hot countries !
```

You are now transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

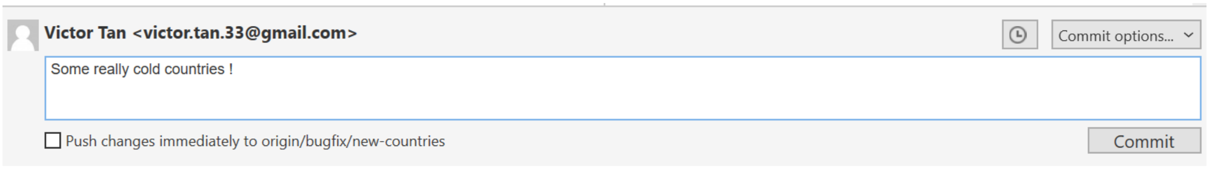
Back in the SourceTree client, select the `bugfix/new-countries` branch and add this new content to the end of `countries.txt`, ensuring you leave a new line at the bottom:

```
Finland
Norway
```

After the SourceTree client refreshes, stage the `countries.txt` file in the usual way as we did earlier.

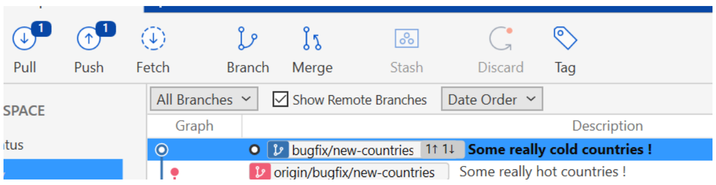


Click the Commit button and add in this new commit message: `Some really cold countries !` in the box at the bottom and select Commit.


Victor Tan <victor.tan.33@gmail.com>
Commit options...
Some really cold countries !
Push changes immediately to origin/bugfix/new-countries
Commit

Next click on the Fetch Button, and select Ok in the dialog box that appears.

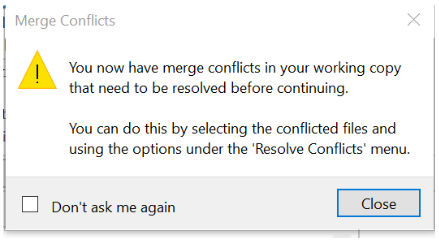
Now you will see in the History view listing, the `bugfix/new-countries` branch is shown with the number 1 next to an up and down arrow key, simply indicating that it is ahead and behind its upstream counterpart by 1 commit. This indicates a divergent branch history. We also have the option to either Pull or Push (as shown in the buttons) to synchronize both branches.



bugfix/new-countries 1↑ 1↓ Some really cold countries !
origin/bugfix/new-countries Some really hot countries !

Let's go ahead and attempt to merge the upstream branch into the local branch by clicking on Pull. Select Pull in the dialog box that appears.

When the Pull operation completes, an merge conflict dialog box appears, and you can click close on it.



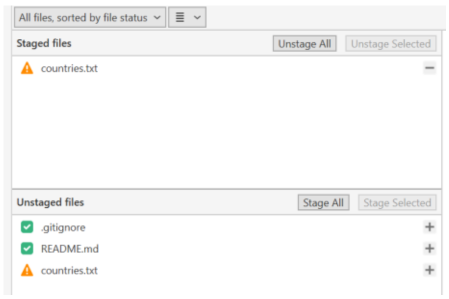
Merge Conflicts

You now have merge conflicts in your working copy that need to be resolved before continuing.

You can do this by selecting the conflicted files and using the options under the 'Resolve Conflicts' menu.

Don't ask me again Close

Notice the symbol for a file with a merge conflict in the Files section below.



countries.txt
.gitignore
README.md
countries.txt

We will resolve this manually as we did in the previous lab when we worked from the command line. Open up `countries.txt` locally. You should see the source of the conflict at the bottom:

```
…
…
Australia
New Zealand
<<<<<<< HEAD
Finland
Norway
=======
Egypt
Kuwait
>>>>>>> 512d425d2315166a45132e8053aa03e7d6490621
```

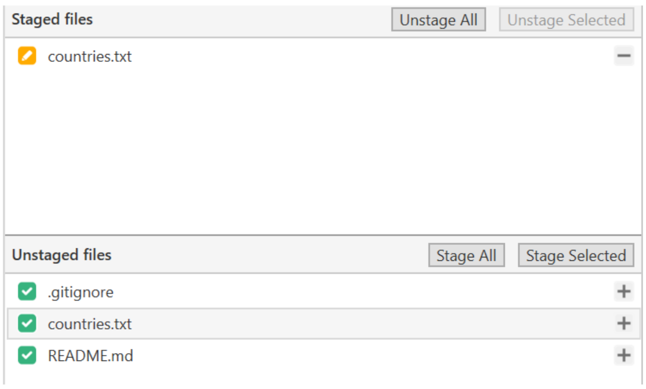
The <<<<<<< indicates that the content below is from the current branch that is being merged into. In this case, it is the branch that HEAD is pointing to: the local `bugfix/new-countries`
The >>>>>>> indicates that the content below is from the branch that is being merged from. In this case, we have the hash of the latest commit from the remote `bugfix/new-countries`

The ======= is a divider between the conflicting content

Change the conflicting part to the content below and save it:

```
Italy
Spain
```

Remember to stage these changes in the usual way by clicking on the + symbol for `countries.txt` in the unstaged section.



After the SourceTree client refreshes with your latest changes, you can proceed to click the Commit button. The commit dialog box will show an autogenerated merge commit message:

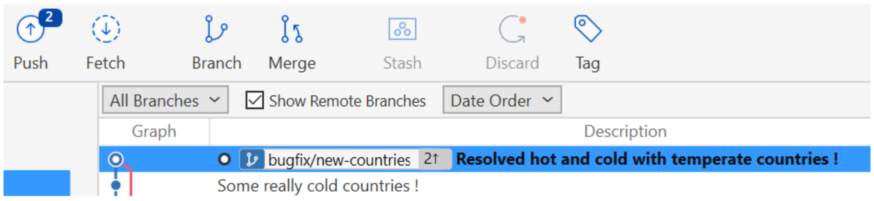
```
Merge branch 'bugfix/new-countries' of https://bitbucket.org/victor-coder/secondremoterepo into bugfix/new-countries

# Conflicts:
#     countries.txt
```

You can use this if you wish, but we replace that with a simpler message instead:

```
Resolved hot and cold with temperate countries !
```

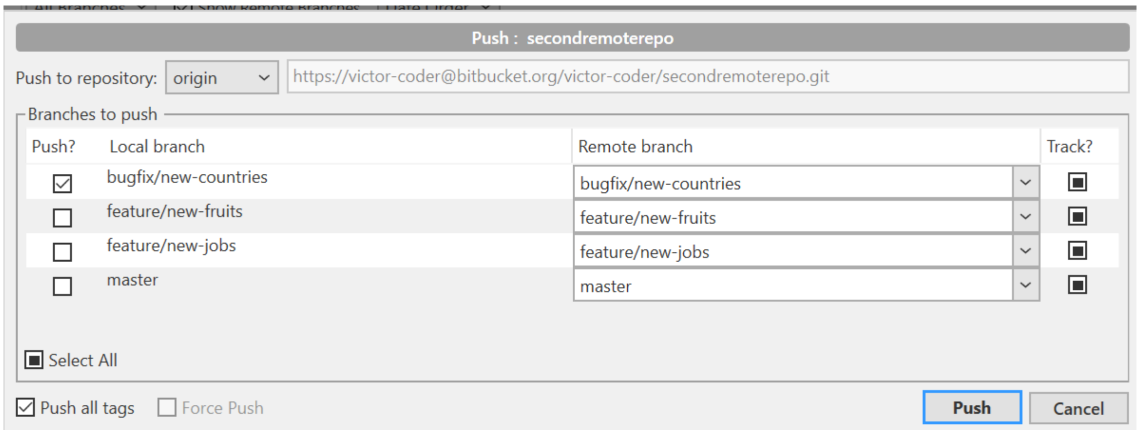
and then click Commit.

Now, as expected, the History listing shows that this local branch is now ahead of its upstream counter part by 2 commits and we are now prompted to push these 2 latest commits back to the remote repo to update the upstream counterpart. Notice that the merge commit has two parent commits, as explained in detail in a previous lab.



Resolved hot and cold with temperate countries !
Some really cold countries !

Click on the Push button and the dialog box that appears should already indicate correctly the branches involved in the push operation. Click Push to continue.


Push : secondremoterepo

Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

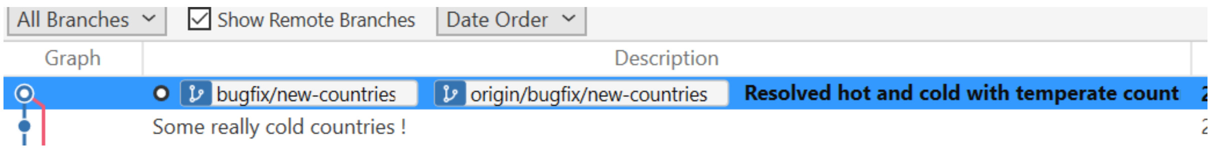
Branches to push

Push?	Local branch	Remote branch	Track?
☑	bugfix/new-countries	bugfix/new-countries	■
☐	feature/new-fruits	feature/new-fruits	■
☐	feature/new-jobs	feature/new-jobs	■
☐	master	master	■

■ Select All

☑ Push all tags ☐ Force Push

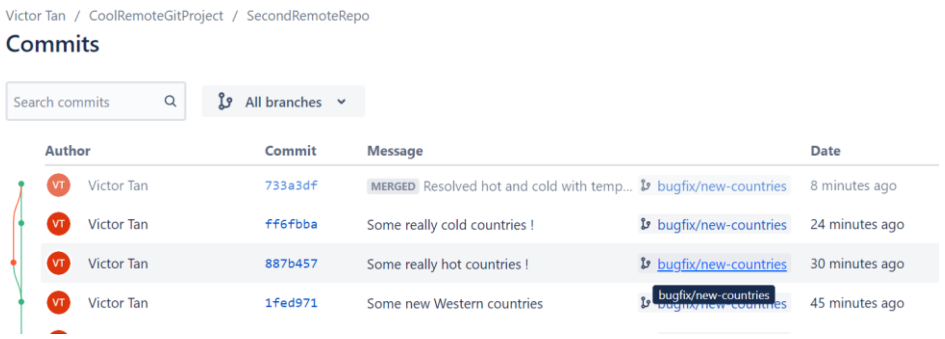
The History view listing will now show both the local and upstream `bugfix/new-countries` synchronized at the same commit.



bugfix/new-countries origin/bugfix/new-countries **Resolved hot and cold with temperate count**

Some really cold countries !

In the browser, navigate to the Commits view and check the commit history for `bugfix/new-countries`. Again, notice the two divergent histories for the latest merge commit.


Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Commits

Author	Commit	Message		Date
Victor Tan	733a3df	MERGED Resolved hot and cold with temp...	bugfix/new-countries	8 minutes ago
Victor Tan	ff6fbba	Some really cold countries !	bugfix/new-countries	24 minutes ago
Victor Tan	887b457	Some really hot countries !	bugfix/new-countries	30 minutes ago
Victor Tan	1fed971	Some new Western countries	bugfix/new-countries	45 minutes ago

Let's repeat the process above with a different branch: `feature/new-fruits`

In the browser, return back to the Source view for the `feature/new-fruits` branch, and select `fruits.txt`. Click on Edit and add this additional content, making sure to include an empty line at the bottom:

```
avocado
guava
```

Click Commit on the lower left hand corner to save the changes into a commit on the current branch (master). Enter this for your commit message in the dialog box that appears, then click Commit:

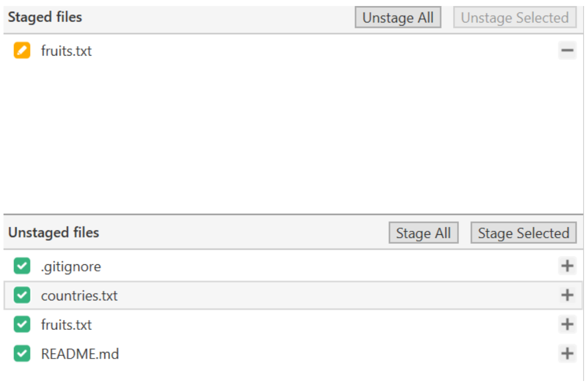
```
Nice fruits from Spain
```

You are now transitioned to a commit page where you can view a variety of information regarding this newly created commit (the commit hash, the branch the commit is on, the commit message and a diff between the current commit content and the previous commit in the commit history timeline).

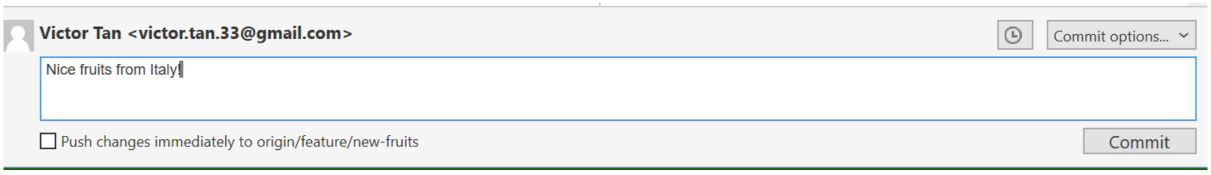
Back in the SourceTree client, select the `feature/new-fruits` branch and add this new content to the end of `fruits.txt`, ensuring you leave a new line at the bottom:

```
pear
peach
```

After the SourceTree client refreshes, stage the `fruits.txt` file in the usual way as we did earlier.



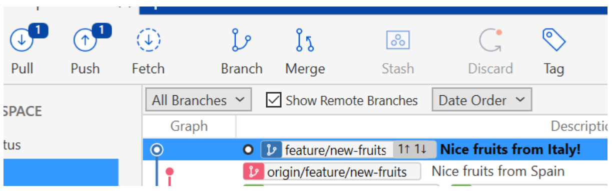
Click the Commit button and add in this new commit message: `Nice fruits from Italy!` in the box at the bottom and select Commit.


Victor Tan <victor.tan.33@gmail.com>

Nice fruits from Italy!

Next click on the Fetch Button, and select Ok in the dialog box that appears.

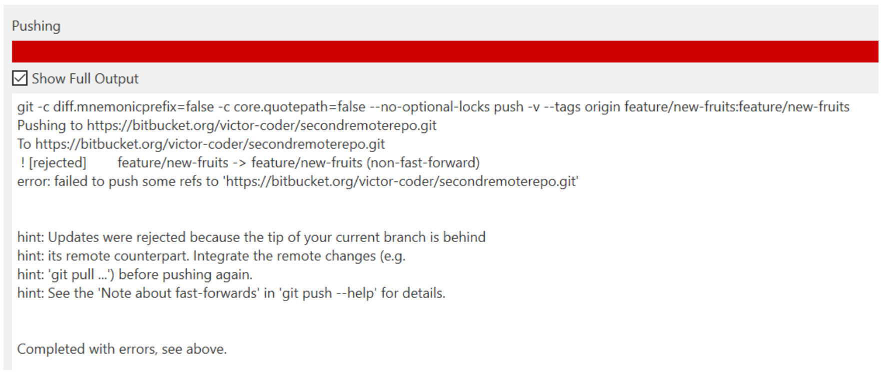
Again, as before, we will now see signs indicating that the current `feature/new-fruits` branch is 1 commit ahead and behind its upstream branch, indicating divergent branch histories.



feature/new-fruits 1↑ 1↓ Nice fruits from Italy!
origin/feature/new-fruits Nice fruits from Spain

Let's see what happens if we now try to push changes from the local branch to its upstream counterpart by click on the Push button. Click Push in the dialog box that appears.

You will now get an error dialog box with the same error messages that you will get if you would performing this from the command line:



Pushing

☑ Show Full Output

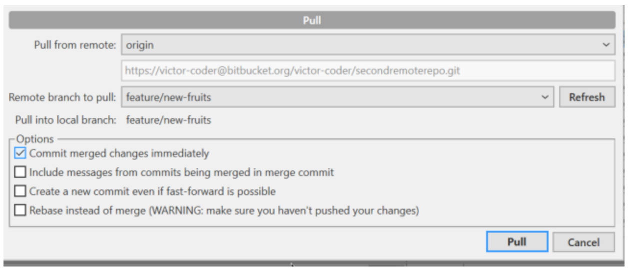
```
git -c diff.mnemonicprefix=false -c core.quotepath=false --no-optional-locks push -v --tags origin feature/new-fruits:feature/new-fruits
Pushing to https://bitbucket.org/victor-coder/secondremoterepo.git
To https://bitbucket.org/victor-coder/secondremoterepo.git
 ! [rejected]        feature/new-fruits -> feature/new-fruits (non-fast-forward)
error: failed to push some refs to 'https://bitbucket.org/victor-coder/secondremoterepo.git'


hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.


Completed with errors, see above.
```

Click Close to close the error dialog box, and cancel to close the Push Dialog box.

Just like the case of the lab where we worked from the command line, if a push operation results in a merge conflict, Git will prevent the operation from going ahead because there is no guarantee that the current user can access the remote repo to resolve the conflict properly. Instead, Git insists on a pull operation (as we have done previously) to resolve the conflict on the local repo. In other words, if there is a merge conflict whenever we try to do a `push`, we **MUST** resolve the conflict locally with a `pull`.

The rest of the operations that we perform next are identical to what we did previously.

Click on the Pull button. Click on Pull in the dialog box that comes up.



Pull

Pull from remote: origin

https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

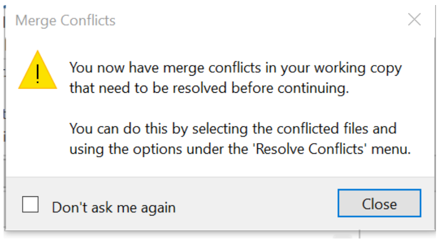
Remote branch to pull: feature/new-fruits

Pull into local branch: feature/new-fruits

Options

- ☑ Commit merged changes immediately
- ☐ Include messages from commits being merged in merge commit
- ☐ Create a new commit even if fast-forward is possible
- ☐ Rebase instead of merge (WARNING: make sure you haven't pushed your changes)

When the Pull operation completes, a merge conflict dialog box appears, and you can click close on it.



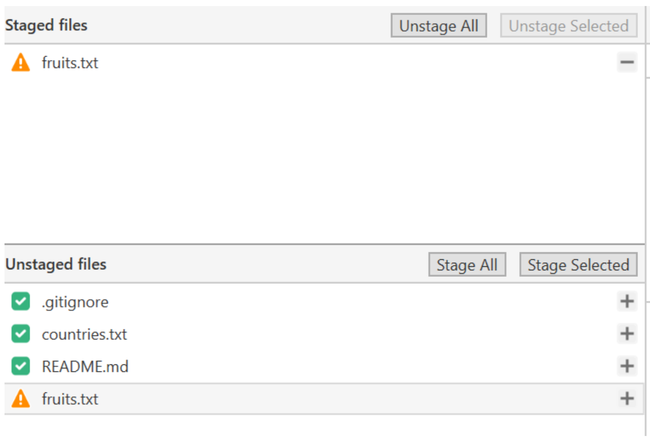
Merge Conflicts

You now have merge conflicts in your working copy that need to be resolved before continuing.

You can do this by selecting the conflicted files and using the options under the 'Resolve Conflicts' menu.

☐ Don't ask me again

Notice the symbol for a file with a merge conflict in the Files section below.



We will resolve this manually as we did in the previous lab when we worked from the command line. Open up `fruits.txt` locally. You should see the source of the conflict at the bottom:

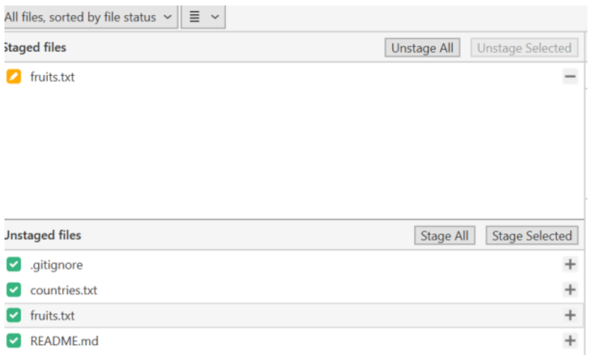
```
…
…
durian
<<<<<<< HEAD
pear
peach
=======
avocado
guava
>>>>>>> 6cce700edc5543339a34be07f0c0c73e3b6f869a
```

The <<<<<<< indicates that the content below is from the current branch that is being merged into. In this case, it is the branch that HEAD is pointing to: the local `feature/new-fruits`
The >>>>>>> indicates that the content below is from the branch that is being merged from. In this case, we have the hash of the latest commit from the remote `feature/new-fruits`
The ======= is a divider between the conflicting content

Change the conflicting part to the content below and save it:

```
avocado
guava
```

Remember to stage these changes in the usual way by clicking on the + symbol for `fruits.txt` in the unstaged section.



After the SourceTree client refreshes with your latest changes, you can proceed to click the Commit button. The commit dialog box will show an autogenerated merge commit message:

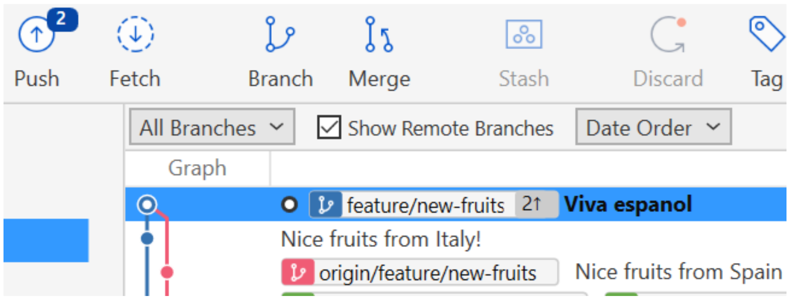
```
Merge branch 'feature/new-fruits' of https://bitbucket.org/victor-
coder/secondremoterepo into feature/new-fruits

# Conflicts:
#	fruits.txt
```

You can use this if you wish, but we replace that with a simpler message instead:

```
Viva espanol
```

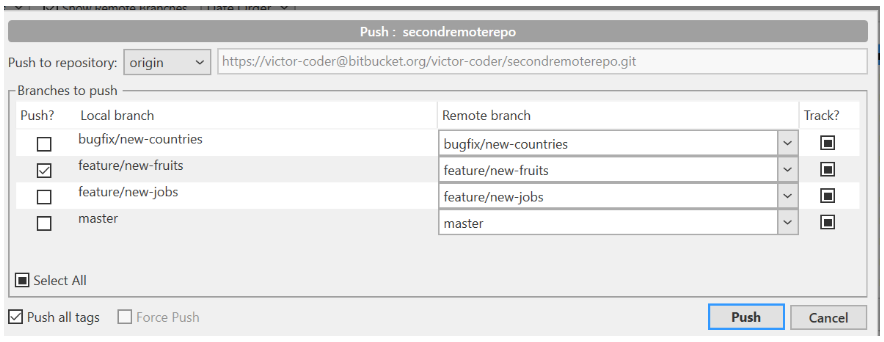
and then click Commit.

Now, as expected, the History listing shows that this local branch is now ahead of its upstream counter part by 2 commits and we are now prompted to push these 2 latest commits back to the remote repo to update the upstream counterpart. Notice that the merge commit has two parent commits, as explained in detail in a previous lab.



feature/new-fruits 2↑ **Viva espanol**
Nice fruits from Italy!
origin/feature/new-fruits Nice fruits from Spain

Click on the Push button and the dialog box that appears should already indicate correctly the branches involved in the push operation. Click Push to continue.


Push : secondremoterepo

Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

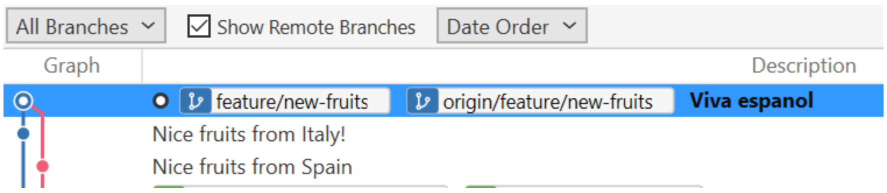
Branches to push

Push?	Local branch	Remote branch	Track?
☐	bugfix/new-countries	bugfix/new-countries	■
☑	feature/new-fruits	feature/new-fruits	■
☐	feature/new-jobs	feature/new-jobs	■
☐	master	master	■

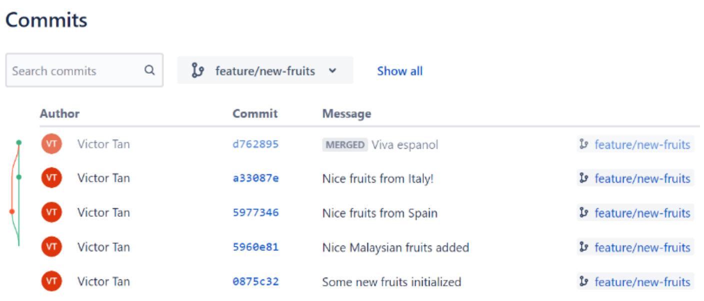
■ Select All

☑ Push all tags ☐ Force Push

The History view listing will now show both the local and upstream `feature/new-fruits` synchronized at the same commit.



In the browser, navigate to the Commits view and check the commit history for `feature/new-fruits`. Again, notice the two divergent histories for the latest merge commit.

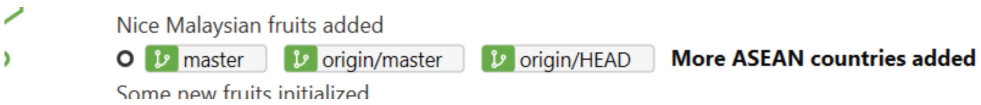


Author	Commit	Message	
Victor Tan	d762895	MERGED Viva espanol	feature/new-fruits
Victor Tan	a33087e	Nice fruits from Italy!	feature/new-fruits
Victor Tan	5977346	Nice fruits from Spain	feature/new-fruits
Victor Tan	5960e81	Nice Malaysian fruits added	feature/new-fruits
Victor Tan	0875c32	Some new fruits initialized	feature/new-fruits

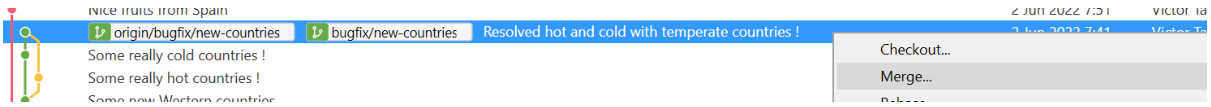
13 Merging branches into master in remote and local repo

In many development workflows, once we have completed implementation for a particular branch (feature, bugfix, hotfix, release, etc), we will integrate it into the main / master branch and then delete that branch. We will see how to do this on the remote and local repo, as well as ensuring that both repos remain in sync after these changes have been made.

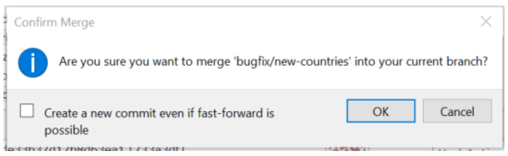
We will start with the local repo. Double click on the local master branch to select it.


Nice Malaysian fruits added
master
origin/master
origin/HEAD
More ASEAN countries added
Some new fruits initialized

Then right click on the `bugfix/new-countries` branch, and select Merge.

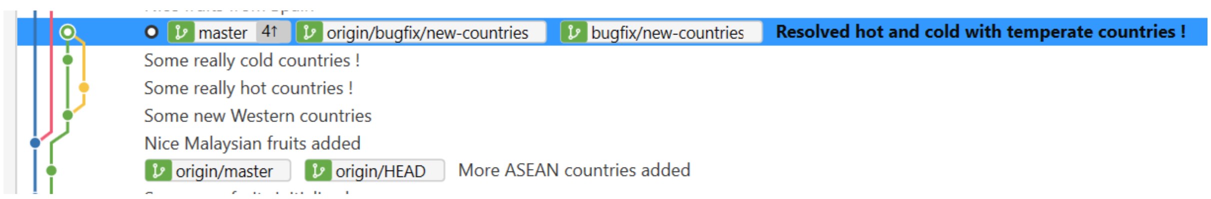

Nice fruits from Spain
origin/bugfix/new-countries
bugfix/new-countries
Resolved hot and cold with temperate countries !
Some really cold countries !
Some really hot countries !
Checkout...
Merge...

In the Dialog box, select Ok.

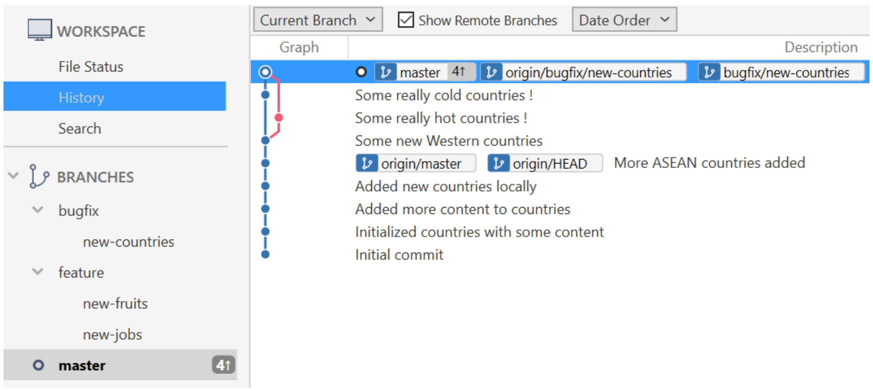

Confirm Merge
Are you sure you want to merge 'bugfix/new-countries' into your current branch?
Create a new commit even if fast-forward is possible
OK
Cancel

As expected, merging in the changes from the `bugfix/new-countries` branch is a straight forward process with a fast forward merge, so no manual merge conflict is required from the user.

The commit history listing should now show the local master being 4 commits ahead of the upstream master.


master 4↑ | origin/bugfix/new-countries | bugfix/new-countries | Resolved hot and cold with temperate countries !
Some really cold countries !
Some really hot countries !
Some new Western countries
Nice Malaysian fruits added
origin/master | origin/HEAD | More ASEAN countries added

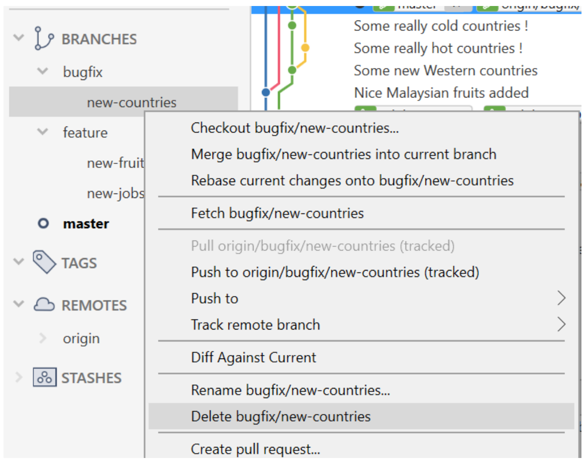
Remember that you can double click on any commit in the listing to make the branch pointing to it the current branch and then use the drop down list to get the commit history for that bracnh. If there is more than one local branch pointing to a given commit, you can use the Branches panel in the side bar to select a specific branch to be the current branch.



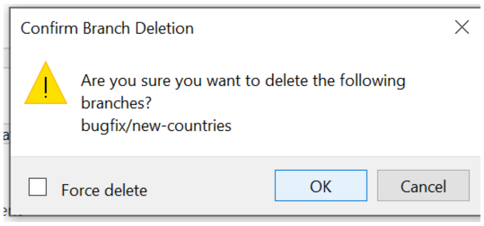
master 4↑ | origin/bugfix/new-countries | bugfix/new-countries
Some really cold countries !
Some really hot countries !
Some new Western countries
origin/master | origin/HEAD | More ASEAN countries added
Added new countries locally
Added more content to countries
Initialized countries with some content
Initial commit

After merging in the latest changes in a feature / bugfix branch (such as `bugfix/new-countries`) into master, we then have the option of deleting this branch.

To delete a local branch, we can right click on that branch in the Branches side bar and select Delete from the context menu.


BRANCHES
bugfix
new-countries
feature
new-fruit
new-jobs
master
TAGS
REMOTES
origin
STASHES
Some really cold countries !
Some really hot countries !
Some new Western countries
Nice Malaysian fruits added
Checkout bugfix/new-countries...
Merge bugfix/new-countries into current branch
Rebase current changes onto bugfix/new-countries
Fetch bugfix/new-countries
Pull origin/bugfix/new-countries (tracked)
Push to origin/bugfix/new-countries (tracked)
Push to
Track remote branch
Diff Against Current
Rename bugfix/new-countries...
Delete bugfix/new-countries
Create pull request...

Click Ok in the dialog box that appears.



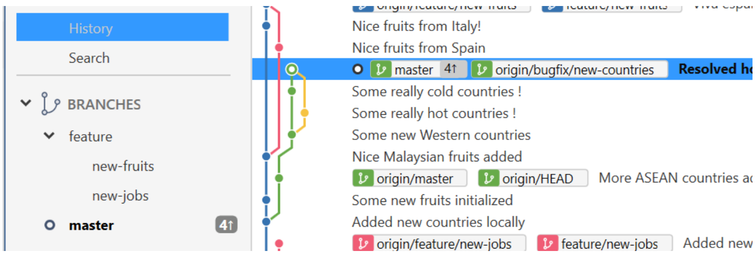
Confirm Branch Deletion

Are you sure you want to delete the following branches?
bugfix/new-countries

☐ Force delete

OK Cancel

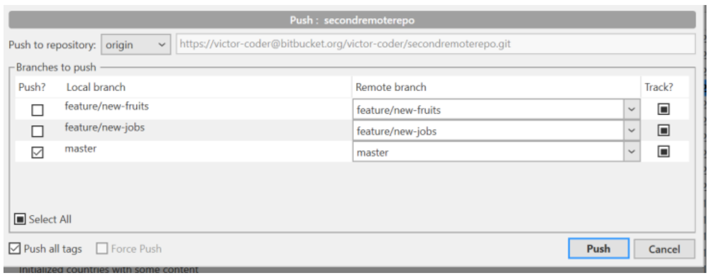
The local branch should now be gone from the commit history listing as well as the Branches side panel. However, the upstream branch equivalent is still there.



History
Search
BRANCHES
feature
new-fruits
new-jobs
master 4↑

Nice fruits from Italy!
Nice fruits from Spain
master 4↑ origin/bugfix/new-countries Resolved h
Some really cold countries !
Some really hot countries !
Some new Western countries
Nice Malaysian fruits added
origin/master origin/HEAD More ASEAN countries a
Some new fruits initialized
Added new countries locally
origin/feature/new-jobs feature/new-jobs Added new

Now, click on the Push button to upload the latest commits from the local master to update the upstream master. Click Push in the dialog box that appears.



Push : secondremoterepo

Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

Branches to push

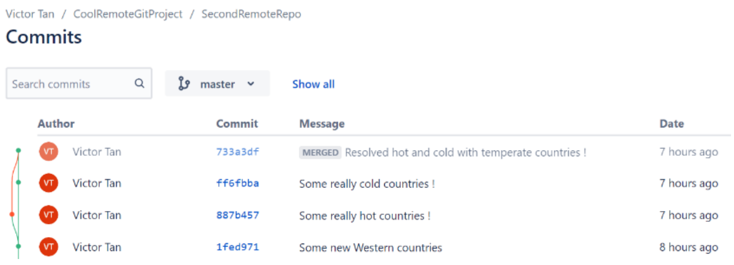
Push?	Local branch	Remote branch	Track?
☐	feature/new-fruits	feature/new-fruits	▣
☐	feature/new-jobs	feature/new-jobs	▣
☑	master	master	▣

▣ Select All

☑ Push all tags ☐ Force Push

Push Cancel

In the browser, navigate to the Commits main view and check the commit history for `master`. Again, notice the two divergent histories for the latest merge commit.



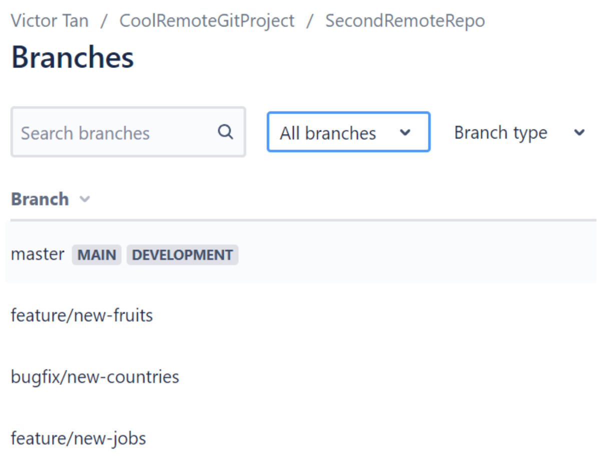
Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Commits

Search commits master Show all

Author	Commit	Message	Date
VT Victor Tan	733a3df	MERGED Resolved hot and cold with temperate countries !	7 hours ago
VT Victor Tan	ff6fbba	Some really cold countries !	7 hours ago
VT Victor Tan	887b457	Some really hot countries !	7 hours ago
VT Victor Tan	1fed971	Some new Western countries	8 hours ago

If you check the Branches main view, you will notice that the push operation has NOT resulted in the deletion of the upstream `bugfix/new-countries`

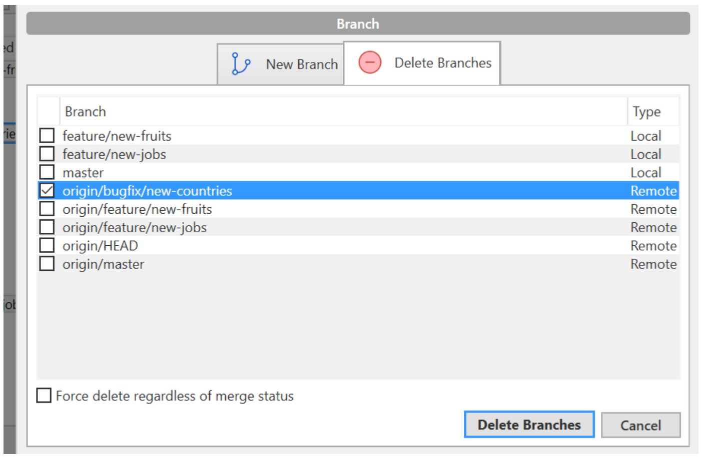

Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Branches

Search branches | All branches | Branch type

Branch
master MAIN DEVELOPMENT
feature/new-fruits
bugfix/new-countries
feature/new-jobs

We have to explicitly specify the deletion of an upstream branch.
Return back to SourceTree, click on Branch Button, select the Delete Branches tab in the dialog box and then check the box for the remote `origin/bugfix/new-countries`. Then select Delete Branches.

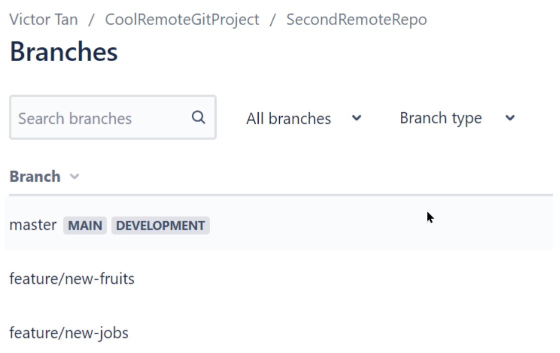


	Branch	Type
☐	feature/new-fruits	Local
☐	feature/new-jobs	Local
☐	master	Local
☑	origin/bugfix/new-countries	Remote
☐	origin/feature/new-fruits	Remote
☐	origin/feature/new-jobs	Remote
☐	origin/HEAD	Remote
☐	origin/master	Remote

☐ Force delete regardless of merge status

Click Yes to the confirmation dialog box that appears next.

If we now check the Branches main view in the browser, the `bugfix/new-countries` branch should now be gone.

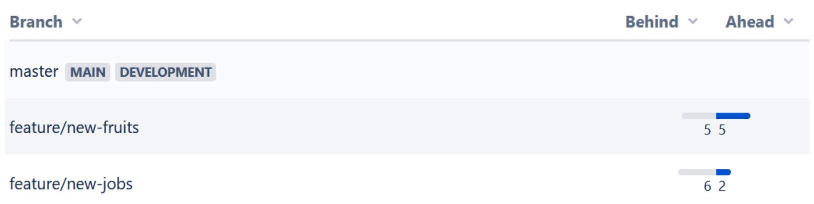

Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Branches

Search branches | All branches | Branch type

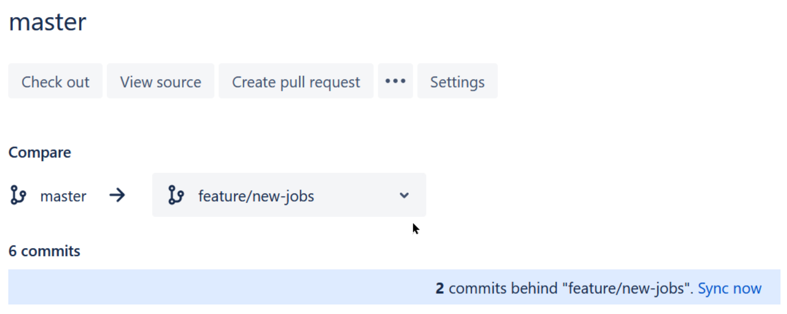
Branch
master MAIN DEVELOPMENT
feature/new-fruits
feature/new-jobs

We will work with the `feature/new-jobs` branch next. Notice at the moment that this branch is 6 commits behind and 2 commits ahead of `master` (all branches are automatically compared with the `master` branch because the expectation is that we will merge into `master` or merge from `master` at some point in the development workflow).



Branch	Behind	Ahead
master MAIN DEVELOPMENT		
feature/new-fruits	5	5
feature/new-jobs	6	2

Then, in the same view, click on `master`. In the drop down list, select the `feature/new-jobs` branch. We now have a comparison (diff) this branch as well as an option to merge this branch into `master` (the `sync now` link).



master

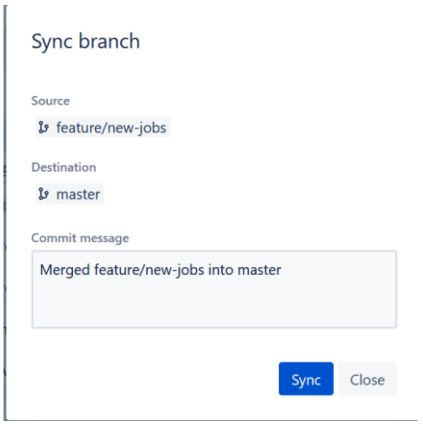
Compare

master → feature/new-jobs

6 commits

2 commits behind "feature/new-jobs". Sync now

Click on the `sync now` link. A dialog box appears. Click Sync to accept the default commit message.



Sync branch

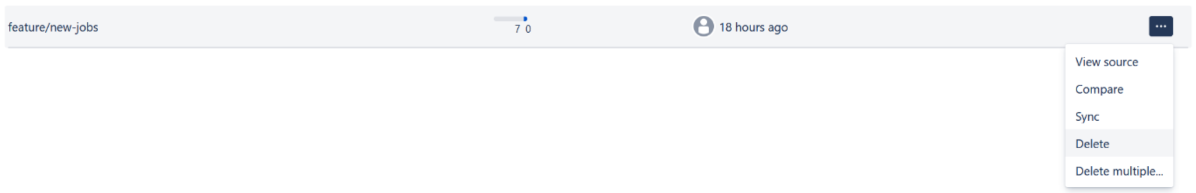
Source
feature/new-jobs

Destination
master

Commit message
Merged feature/new-jobs into master

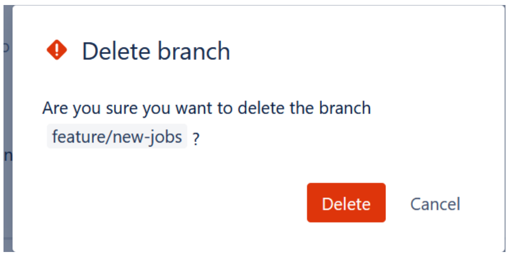
When this is complete, you can return to the Source view in the master branch and check that the project folder now contains `jobs.txt` with the latest content from `feature/new-jobs`

Next, return to the Branches view and select the Delete action for the `feature/new-jobs` branch



feature/new-jobs 7 0 18 hours ago

Select Delete from the pop up box that appears.

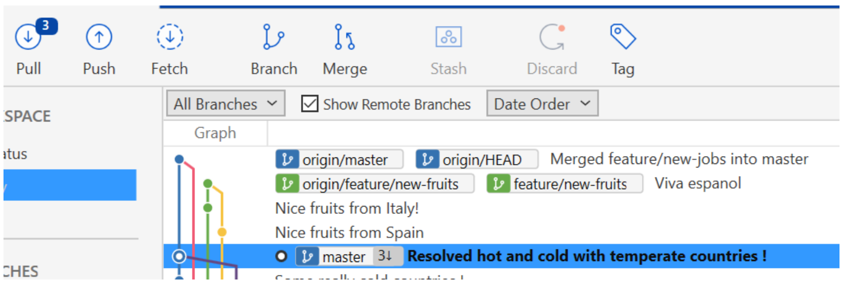


Delete branch

Are you sure you want to delete the branch feature/new-jobs ?

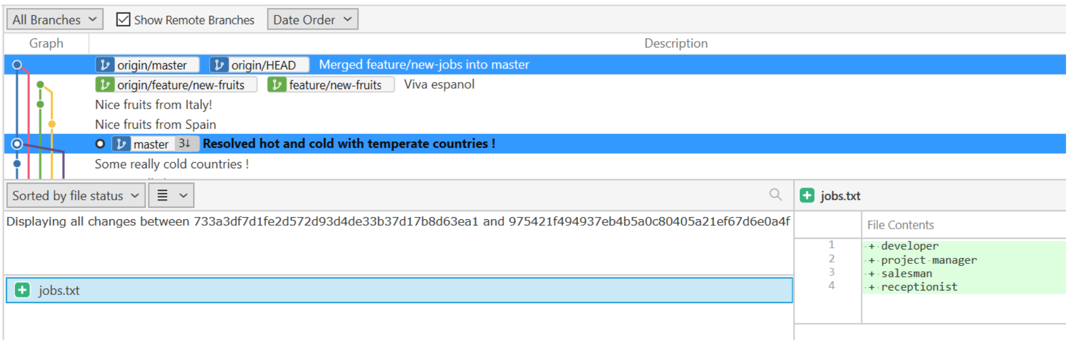
A pop up message indicating successful deletion appears in the lower left hand corner. You should no longer see this branch in the list in the main Branches view.

Back in the SourceTree client, click on the Fetch Button and click on Ok. We can now see that the upstream master is 3 commits ahead of the local master.



origin/master origin/HEAD Merged feature/new-jobs into master
origin/feature/new-fruits feature/new-fruits Viva espanol
Nice fruits from Italy!
Nice fruits from Spain
master 3↓ **Resolved hot and cold with temperate countries !**

To do a diff between the local and remote master branches (or in fact, to do a diff between any 2 commits in the commit history listing), highlight the 2 commits concerned using Ctrl and then click on the relevant files in the section below (in this case, simply `jobs.txt`).



origin/master origin/HEAD Merged feature/new-jobs into master
origin/feature/new-fruits feature/new-fruits Viva espanol
Nice fruits from Italy!
Nice fruits from Spain
master 3↓ **Resolved hot and cold with temperate countries !**
Some really cold countries !

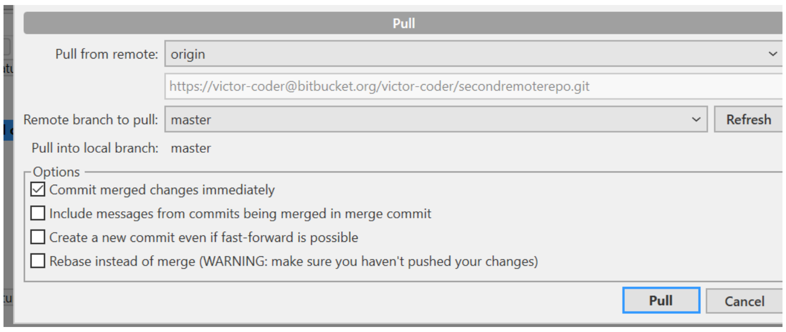
Displaying all changes between 733a3df7d1fe2d572d93d4de33b37d17b8d63ea1 and 975421f494937eb4b5a0c80405a21ef67d6e0a4f

jobs.txt

jobs.txt

```
1  + developer
2  + project manager
3  + salesman
4  + receptionist
```

Select the local master branch again in the commit history listing, and click on the Pull button. Select Pull in the Dialog box that appears.


Pull

Pull from remote: origin

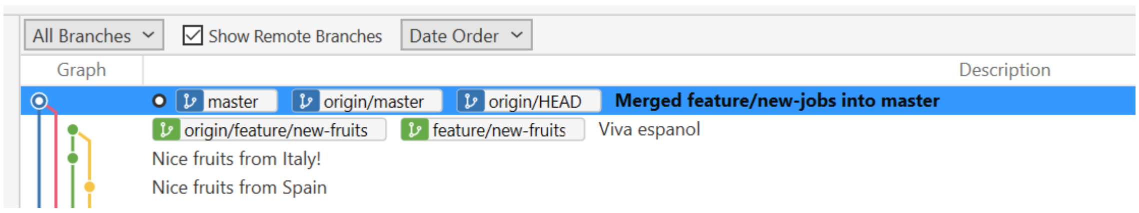
https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

Remote branch to pull: master

Pull into local branch: master

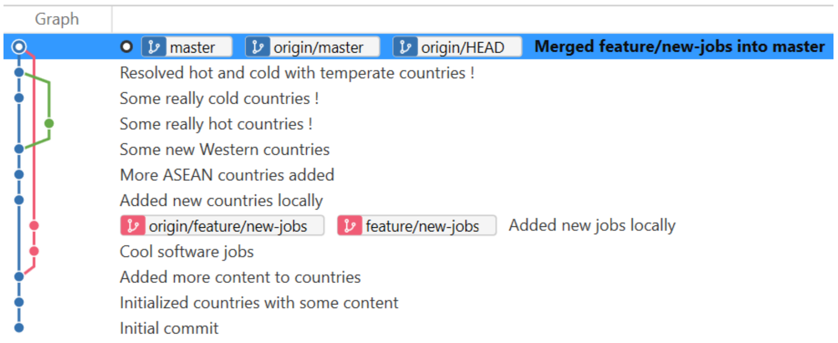
Options
- [x] Commit merged changes immediately
- [ ] Include messages from commits being merged in merge commit
- [ ] Create a new commit even if fast-forward is possible
- [ ] Rebase instead of merge (WARNING: make sure you haven't pushed your changes)

This will be a straightforward fast forward merge, and now our local and upstream master branches are fully synchronized.



master | origin/master | origin/HEAD Merged feature/new-jobs into master
origin/feature/new-fruits | feature/new-fruits Viva espanol
Nice fruits from Italy!
Nice fruits from Spain

As usual, if the commit history of all divergent branches is too complex to understand, you can always zoom down on showing the commit history of the current branch (lets say master).

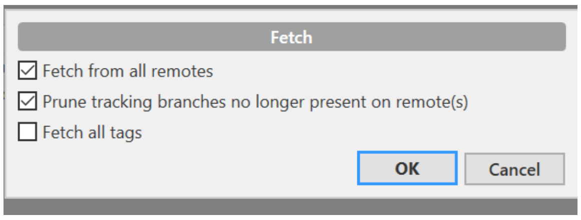


master | origin/master | origin/HEAD Merged feature/new-jobs into master
Resolved hot and cold with temperate countries !
Some really cold countries !
Some really hot countries !
Some new Western countries
More ASEAN countries added
Added new countries locally
origin/feature/new-jobs | feature/new-jobs Added new jobs locally
Cool software jobs
Added more content to countries
Initialized countries with some content
Initial commit

Here, the blue line at the back is the commit timeline for master. There are two other colored branches that have been merged into master. The pink one is for `feature/new-jobs` which we just only merged in - this branch still exists in the local repo. We can also see a green divergent branch although there is no existing branch corresponding to this. This is actually the history for the `bugfix/new-countries` branch that we merged into master earlier on, and then subsequently deleted. Remember that a branch is simply a pointer to a commit, so deleting a branch DOES NOT delete the commit. We still have the `Some really hot countries !` commit that belongs to that branch which is still in the repo. So even though we may have deleted a branch, the color coding scheme that SourceTree utilizes will actually tell us how many different branches have been merged into the master branch over the life time of the project.

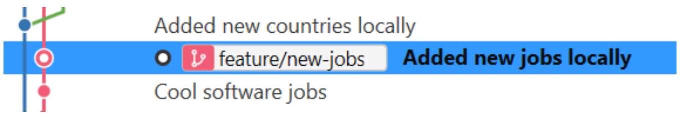
Next, we follow up on the local `feature/new-jobs` branch. Double click on it to select it in the commit history listing then click Fetch button again and then click ok.
Notice that it still shows the upstream `origin/feature/new-jobs`, although we already deleted this earlier through the browser UI in the main Branches view.
As we saw in the previous lab that we did via the command line, this issue is due to outdated (or stale) remote tracking branches. To resolve this, we need to prune the stale references in our remote tracking branches.

Click the Fetch button again, and this time select the Prune tracking branches option and then then click ok.



- [✓] Fetch from all remotes
- [✓] Prune tracking branches no longer present on remote(s)
- [ ] Fetch all tags

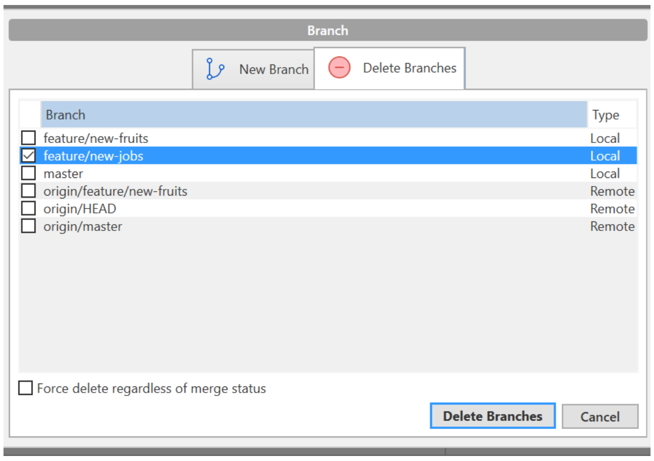
After the fetch is complete, we can now clearly see that we only have a local `feature/new-jobs`, the remote counterpart has been deleted. Note that pruning our remote tracking branches does not automatically delete the local counterpart branch, that action has to be explicitly initiated by us.



Added new countries locally
feature/new-jobs **Added new jobs locally**
Cool software jobs

To delete the local `feature/new-jobs`, we can select it in the Branches side panel and select Delete from the right-click context menu as we did before for another local branch. Alternatively, we can just click the Branch button and select this branch to delete.

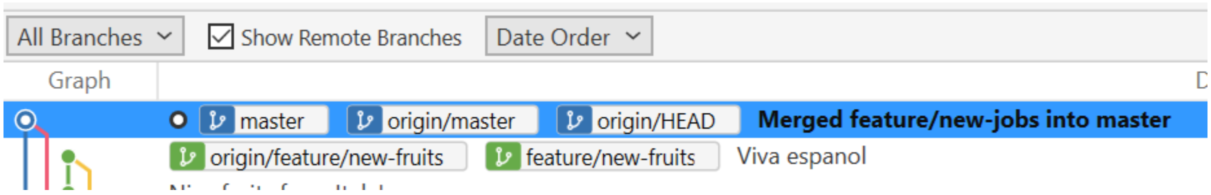
Before doing that, we need to move to another branch first because we cannot delete `feature/new-jobs` if it is the current / active branch. So double click on master first to make it the active branch, then click the Branch button and select the `feature/new-jobs` branch to delete. Click Delete Branches, and respond Yes to the confirmation query.



	Branch	Type
☐	feature/new-fruits	Local
☑	feature/new-jobs	Local
☐	master	Local
☐	origin/feature/new-fruits	Remote
☐	origin/HEAD	Remote
☐	origin/master	Remote

- [ ] Force delete regardless of merge status

The deletion should be successful, so now there will only one additional branch left in the repo other than master: `feature/new-fruits`.



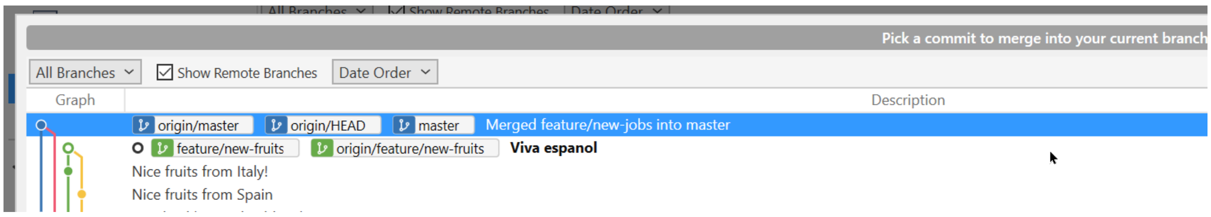
master origin/master origin/HEAD **Merged feature/new-jobs into master**
origin/feature/new-fruits feature/new-fruits Viva espanol

So far, we have demonstrated merging the contents of a branch into `master`. Sometimes, we may also wish to merge the contents of `master` into an existing branch if we still want to continue development on that branch but we want it to be updated with the latest content from `master`.

Let's do that now for `feature/new-fruits`: we will merge the content of master into it.

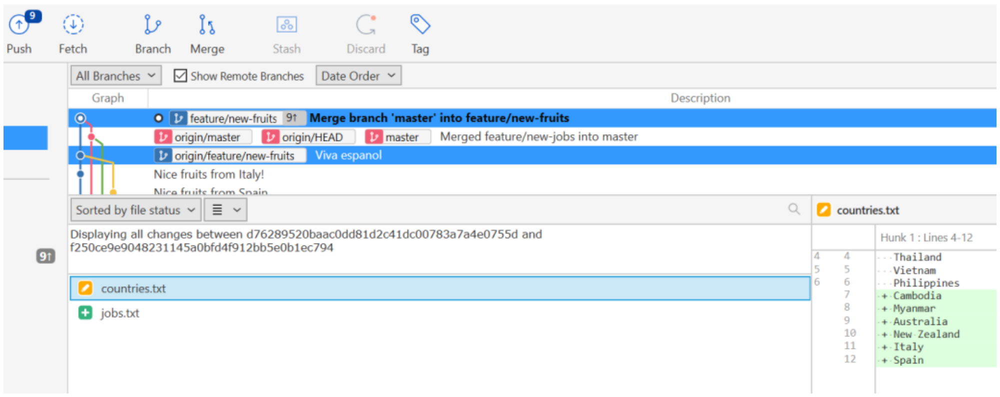
There are two main ways perform a merge, we have already seen one of them earlier. For e.g. we could double click on `feature/new-fruits` to make it the current branch, then right click on master and select merge from the context menu.

The other approach is to double click on `feature/new-fruits` to make it the current branch, then click on the Merge button. In the dialog box that appears, select the master branch as the commit to merge into the current branch, and then select Ok.



origin/master origin/HEAD master Merged feature/new-jobs into master
feature/new-fruits origin/feature/new-fruits **Viva espanol**
Nice fruits from Italy!
Nice fruits from Spain

With the merge complete, we can now see that `feature/new-fruits` is 9 commits ahead of its upstream counterpart. As usual, if we want to see a diff between the local `feature/new-fruits` and its upstream counterpart, simply highlight both branches and check the files section below to view the diff.



feature/new-fruits 9↑ **Merge branch 'master' into feature/new-fruits**
origin/master origin/HEAD master Merged feature/new-jobs into master
origin/feature/new-fruits Viva espanol
Nice fruits from Italy!
Nice fruits from Spain

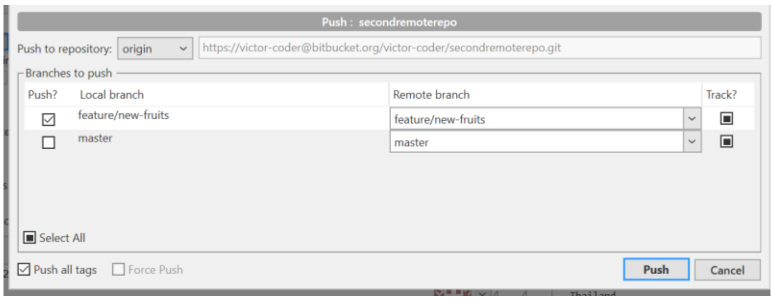
Displaying all changes between d76289520baac0dd81d2c41dc00783a7a4e0755d and f250ce9e9048231145a0bfd4f912bb5e0b1ec794

countries.txt
jobs.txt

countries.txt
Hunk 1 : Lines 4-12

```
4  4    Thailand
5  5    Vietnam
6  6    Philippines
   7  + Cambodia
   8  + Myanmar
   9  + Australia
   10 + New Zealand
   11 + Italy
   12 + Spain
```

Finally, select `feature/new-fruits` again and click the Push button. Click the Push button in the Dialog box that appears.



Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondremoterepo.git

Branches to push

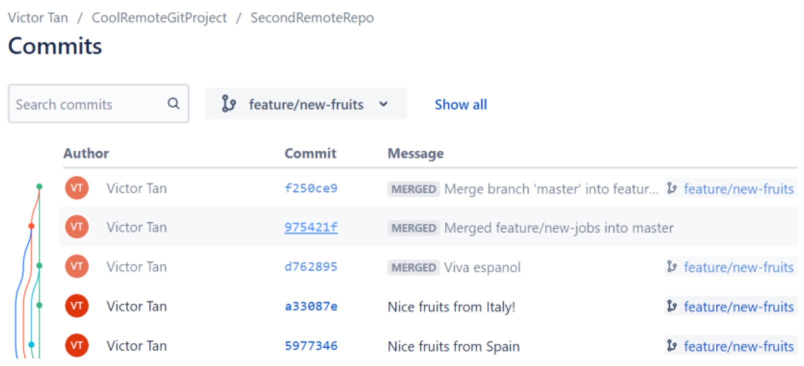
Push?	Local branch	Remote branch	Track?
☑	feature/new-fruits	feature/new-fruits	■
☐	master	master	■

■ Select All

☑ Push all tags ☐ Force Push

After this, we should see both the local and upstream `feature/new-fruits` fully in sync

If we head back to main Commits view in the browser and check out this branch, we should see the latest merge commit from our local repo that we have pushed up into it:


Victor Tan / CoolRemoteGitProject / SecondRemoteRepo

Commits

Author	Commit	Message	
Victor Tan	f250ce9	MERGED Merge branch 'master' into featur...	feature/new-fruits
Victor Tan	975421f	MERGED Merged feature/new-jobs into master	
Victor Tan	d762895	MERGED Viva espanol	feature/new-fruits
Victor Tan	a33087e	Nice fruits from Italy!	feature/new-fruits
Victor Tan	5977346	Nice fruits from Spain	feature/new-fruits

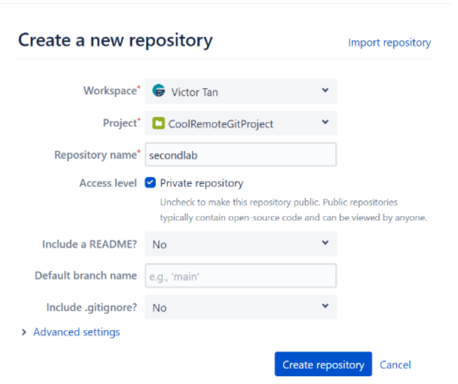
14 Publishing a local repository to a remote repository

In this lab, we started off with an empty (or nearly empty) remote repository, which team members can then clone to a local repository on their respective machines and perform their development work there. The other typical scenario is that we may have an existing local repository that was initially private that we now wish to upload to a remote repository so that can be shared with others in a collaborative team effort.

We will use the `secondlab` project folder from a previous lab to demonstrate this scenario. First, we will need to create an empty remote repository to which we can upload our existing local repository to. To keep things simple, we will create it with the same name as the local repository, although it can be different as well.

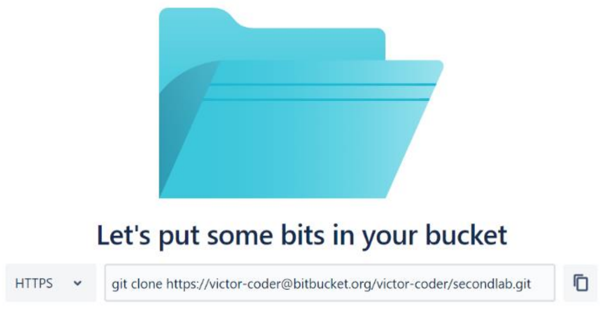
Go ahead and repeat the process of creating a new repository that we did earlier, but this time we will create a bare repository with no content to facilitate the process of pushing the contents of our local repository to it:

Enter the values for the following fields.
Project Name: `CoolRemoteGitProject`
Repository Name: `secondlab`


Create a new repository

Workspace*	Victor Tan
Project*	CoolRemoteGitProject
Repository name*	secondlab
Access level	☑ Private repository Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.
Include a README?	No
Default branch name	e.g., 'main'
Include .gitignore?	No

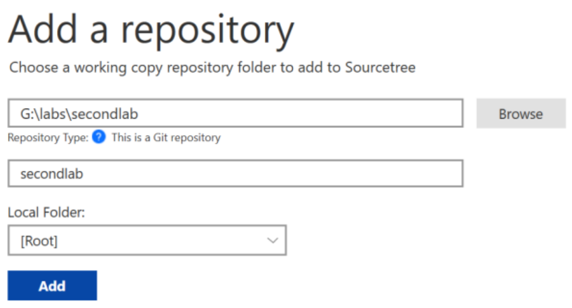
When you are done specifying the values for the fields as shown above, click Create Repository. You will be transitioned to the Source view for the newly created repo, where some instructions will be provided on how to get started.



git clone https://victor-coder@bitbucket.org/victor-coder/secondlab.git

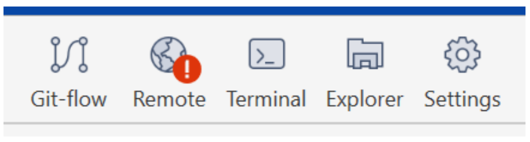
Click on the Clone button to copy the URL (e.g. https://xxx@bitbucket.org/yyy/secondlab.git) for this new remote repo to an empty document. We will refer to this URL as *`remote-url`* in the configuration to follow.

In SourceTree, click on the + key to open a new tab, and click on Add. Select the `secondlab` project folder and click Add.

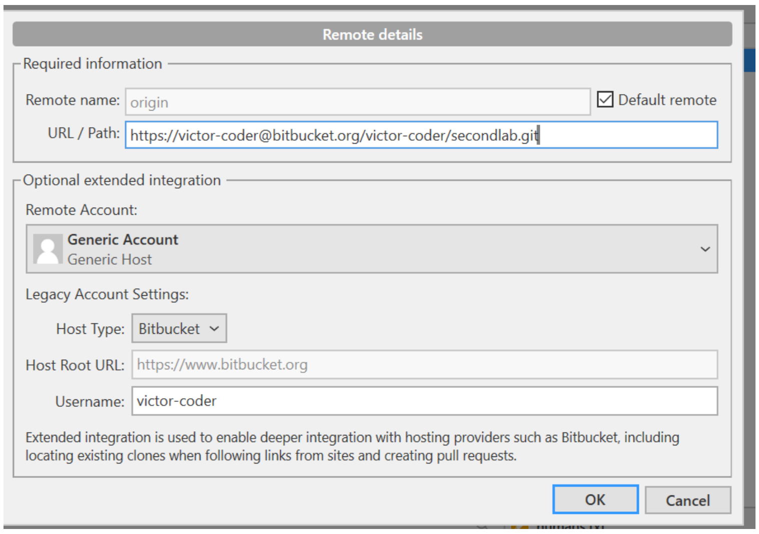

Add a repository
Choose a working copy repository folder to add to Sourcetree
Browse
Add

G:\labs\secondlab
Repository Type: This is a Git repository
secondlab
Local Folder:
[Root]

The Commit history view listing will be shown in the main view, but the Remote button will be marked with a red apostrophe indicating that the repository is completely local and not connected to any remote repo.

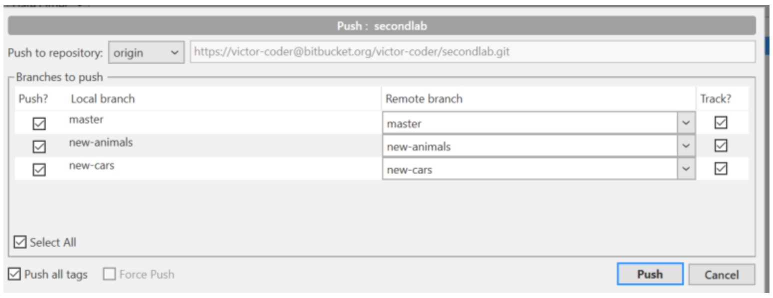


Click on the Settings button and in the Repository Settings dialog box, select Remotes and click Add. Enter the following entry in the Remote Details box, using the *`remote-url`* you copied earlier.


Remote details
Required information
Remote name: origin
☑ Default remote
URL / Path: https://victor-coder@bitbucket.org/victor-coder/secondlab.git
Optional extended integration
Remote Account:
Generic Account
Generic Host
Legacy Account Settings:
Host Type: Bitbucket
Host Root URL: https://www.bitbucket.org
Username: victor-coder
Extended integration is used to enable deeper integration with hosting providers such as Bitbucket, including locating existing clones when following links from sites and creating pull requests.
OK
Cancel

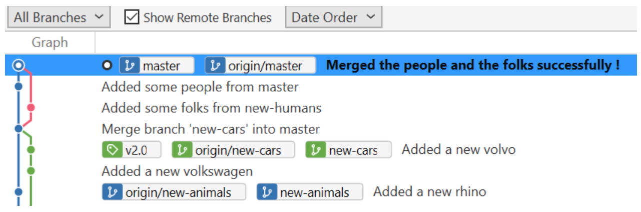
Finally, click Ok and then click Ok in the main Repository Settings.

Click on the Push button and select all the existing local branches and tags, and click Push.


Push : secondlab
Push to repository: origin https://victor-coder@bitbucket.org/victor-coder/secondlab.git
Branches to push
☑ Select All
☑ Push all tags
☐ Force Push
Push
Cancel

Push?	Local branch	Remote branch	Track?
☑	master	master	☑
☑	new-animals	new-animals	☑
☑	new-cars	new-cars	☑

The remote tracking branches for all local branches should now appear in the main commit history view listing and Remotes side bar panel.



Return back to the browser and check through the Source, Commits and Branches main view to verify that these mirrors the status of the local repo that you just pushed up into it. Note that you may need to navigate out to the main Repositories tab in the main menu, and then click on the `secondlab` repo entry in order for it to refresh properly and show the uploaded content. This may take some time to complete correctly.

Back in the Git Bash shell in `secondlab`, type the following commands to verify that the remote tracking branches have been set up and that the local and upstream master are both in sync with each other.

```
$ git remote show origin
```

```
$ git status
```

Other users can now clone this remote repo in the same way that we have just demonstrated previously.

15 Creating and working with a local repository

So far, we have seen examples of using SourceTree to link a remote and local repo. However, we can also use SourceTree as a Git GUI client for a local only repository.
Let's repeat the Git Lab 4 exercise in a new folder `labs\thirdlab`

Create 3 files named as below and populate them with a single line each as shown using a text editor. Make sure you include a new line after the end of the single line

```
1: developer
```

humans.txt

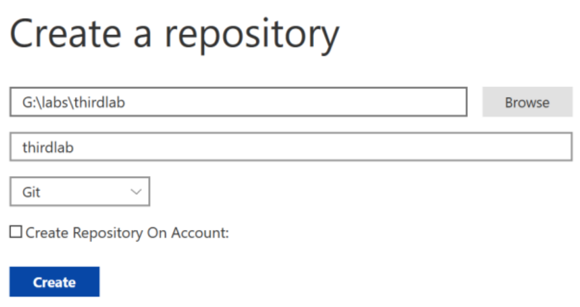
```
1: cat
```

animals.txt

```
1: honda
```

cars.txt

In SourceTree, click on the + key to open a new tab, and click on Create. Enter the appropriate details. Notice that you have the option to create a corresponding remote repo to link to this newly initialized repo if you wish to.


Create a repository

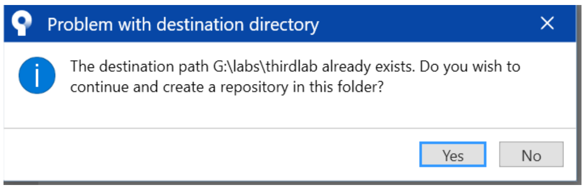
G:\labs\thirdlab

thirdlab

Git

☐ Create Repository On Account:

A message pops up to verify an issue with the destination directory (this is because SourceTree typically expects to initialize a completely blank directory, not one with some existing content). Click Yes to continue.


Problem with destination directory

The destination path G:\labs\thirdlab already exists. Do you wish to continue and create a repository in this folder?

Click Create.

Check that you now have a .git folder for the Git repository in `thirdlab`.

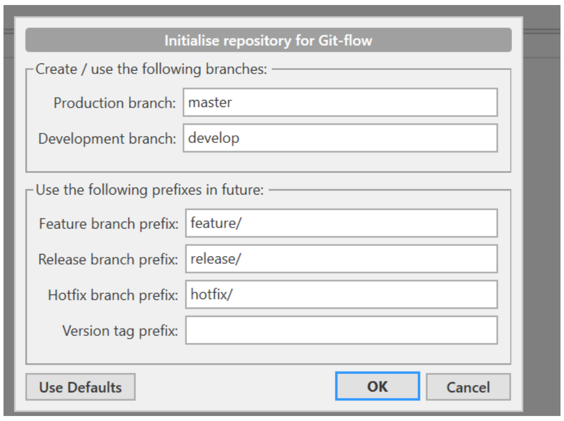
In the files view, make sure you select All files, and you should be able to see the unstaged changes in the 3 new files. Go and ahead and stage them and commit them in accordance to the instructions in Git Lab 4.

16 Conclusion

We have seen here some of the more common operations that will be performed in interaction with a remote repo. There are a variety of other Git operations that have not been demonstrated in this lab such as:

- Stashes
- Tags
- Rebase
- Cherry Pick
- Reverse Commit
- Etc, etc

SourceTree provides a template for creating a GitFlow workflow by clicking the GitFlow button. The more popular workflow now for DevOps team is trunk-based development as it works well with CI / CD tools.


Initialise repository for Git-flow

Create / use the following branches:

Production branch: master

Development branch: develop

Use the following prefixes in future:

Feature branch prefix: feature/

Release branch prefix: release/

Hotfix branch prefix: hotfix/

Version tag prefix:

Clicking on the Remote button transitions you to the URL for the remote repo that the current local repo is linked to.

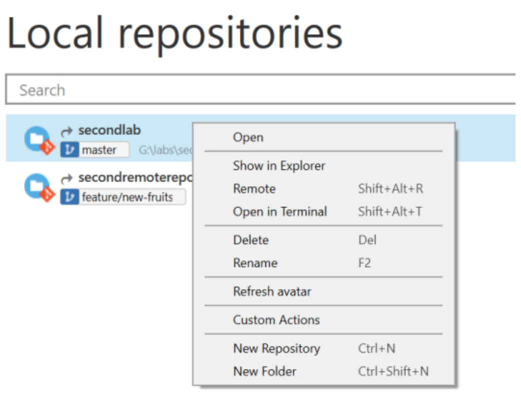
Clicking on Terminal opens a Git Bash shell in the folder of the current branch. Remember that all the actions that we have performed so far via SourceTree are accomplished under the hood using the standard Git command line commands - so we can always resort to working from the command line if there is a specific specialized Git operation that we cannot effectively perform from inside SourceTree (for e.g. `git reflog`).

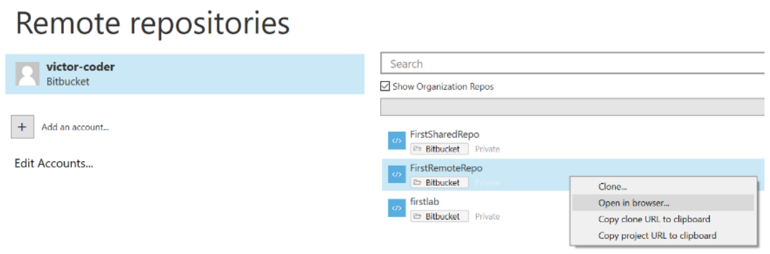
Clicking on Explorer opens the File Explorer in the root of the current Git project.

Clicking on Settings allows you to customize the Setting for the remote repo as well as edit the relevant configuration file, and some other Advanced Settings.

The Tools -> Options dialog box allows you to customize the appearance and functionality of SourceTree in various aspects. One of the useful supplementary tools that you can use to enhance your work with Git is to use an external diff tool that significantly helps when comparing major differences between large files. This can be configured here, otherwise the standard Git diff functionality is provided.

In a new Tab, you can right click on either the Local or Remote list of repositories to get a context menu with a set of options that are relevant for that repository type.





Another useful functionality available in SourceTree is to specify custom actions that you wish to repeat many times which is not immediately available from existing SourceTree functionality.