Unity Version Control

with BitBucket.org and SourceTree

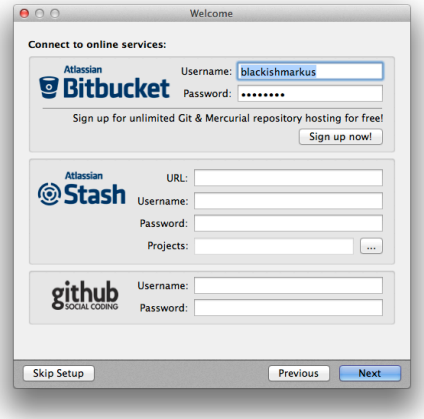
BLACKISH - last updated: April 2013
http://blog.blackish.at

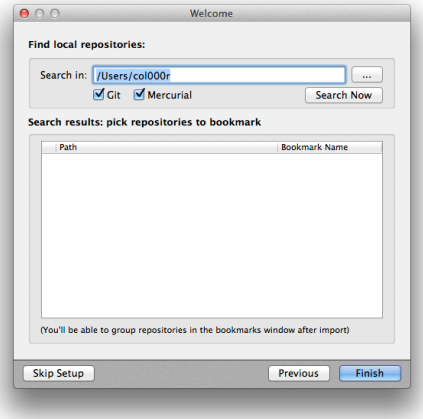
Table of Contents

Setup

Sign up with bitbucket.org - remember your username!
Download SourceTree from sourcetreeapp.com (After a few weeks of use you'll be required to register with Atlassean to get a free license)

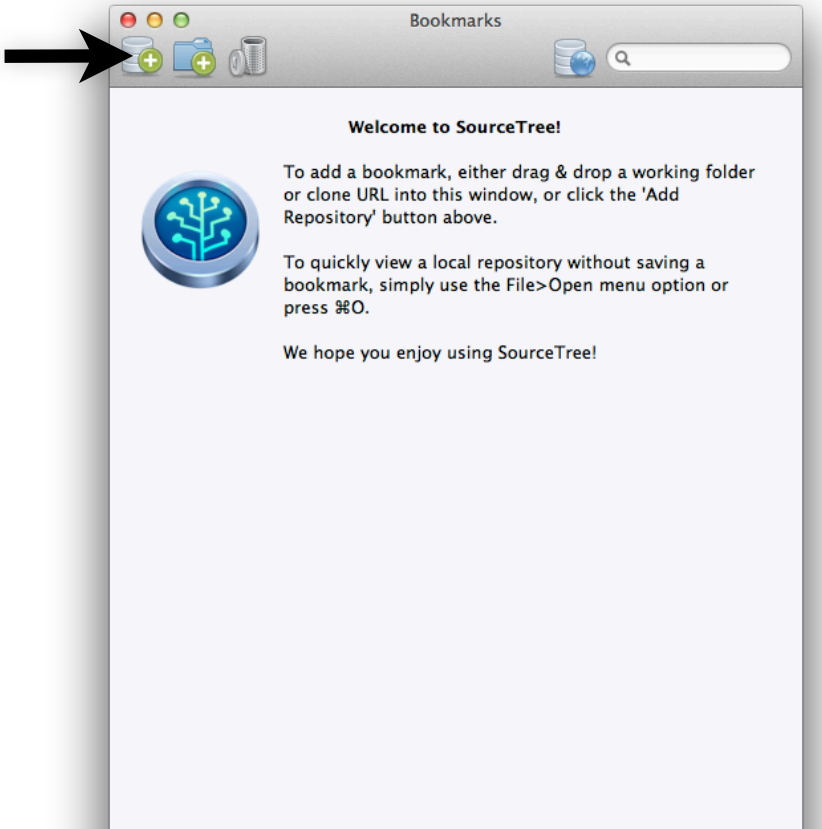
Launch SourceTree, run through the Setup wizard (enter your bitbucket username+pw on page 2)





Find the Bookmarks window (⌘B)

Click the Add Repository button - And you're ready to join an existing repository or set up a new one!



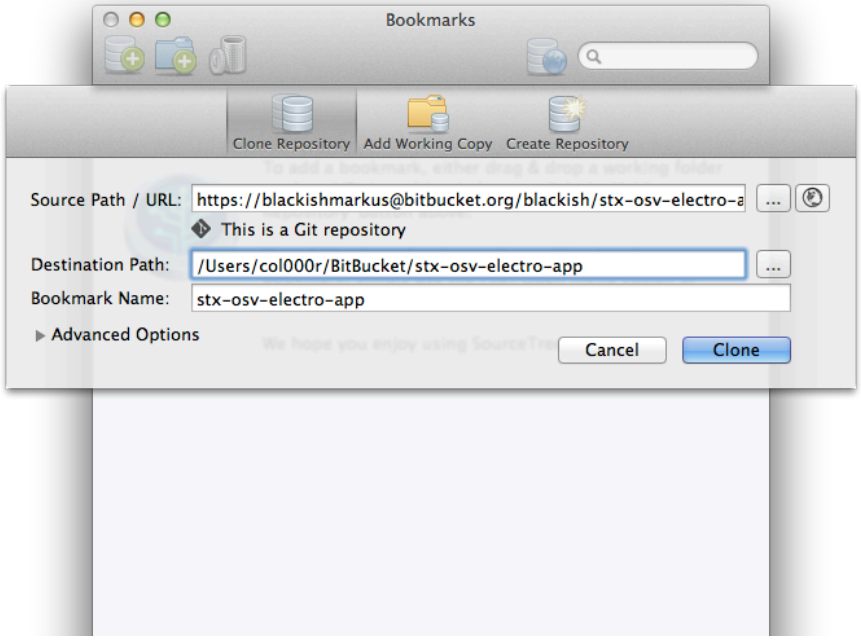
Join an existing repository

In the Bookmarks window, click Add Repository.

If you were invited to join a repository, enter the URL you received in the URL field, replace YOURSUSERNAME with your BitBucket username:

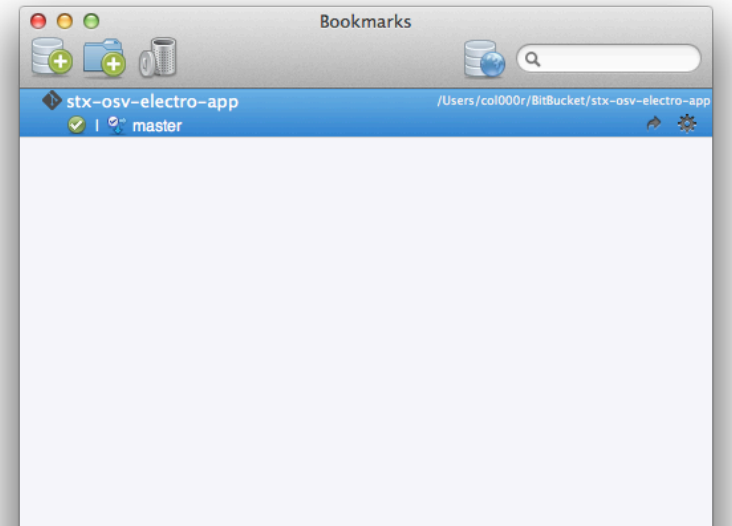
For Example: https://**YOURUSERNAME**@bitbucket.org/OTHERUSER/projectname.git

Select where you want the repository to reside on your HardDrive.

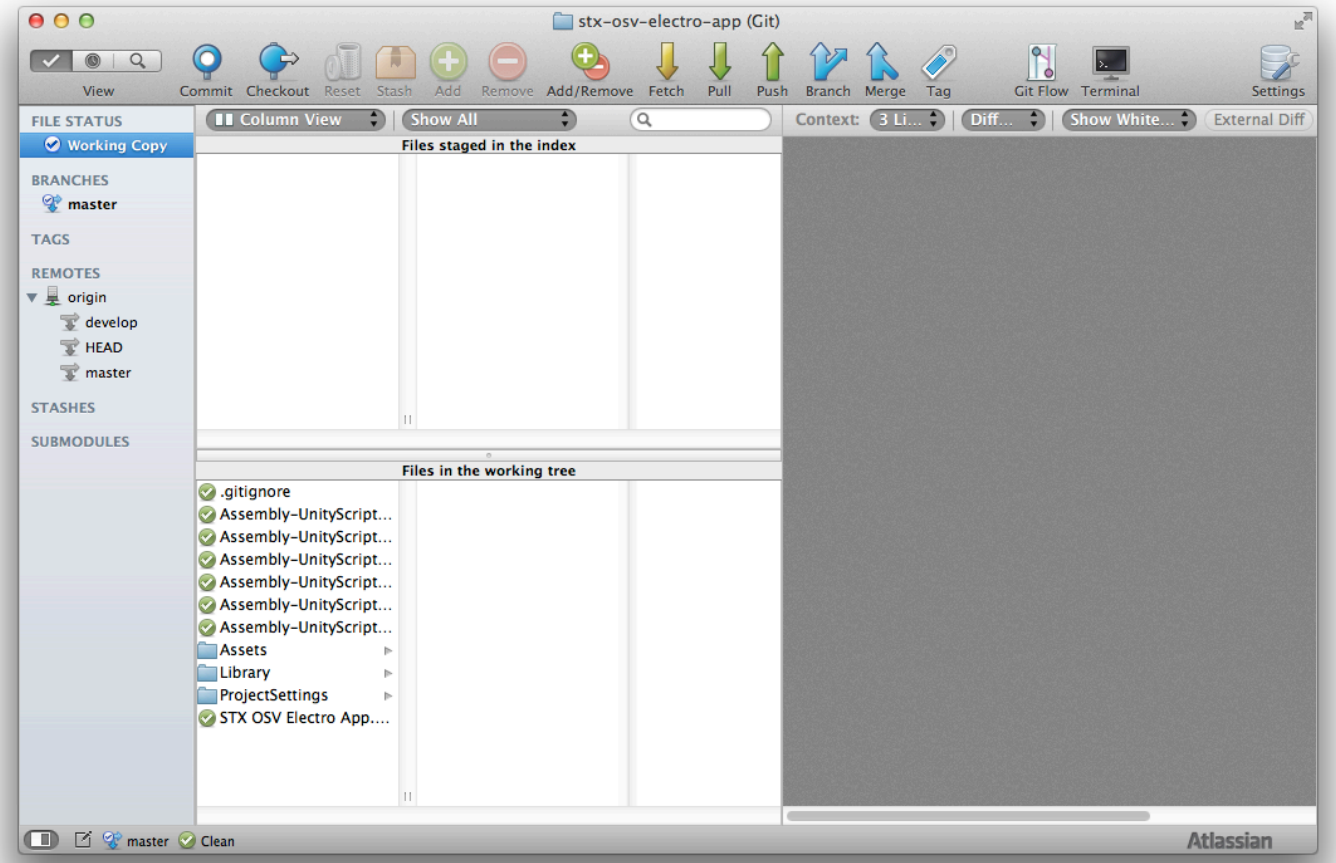


Click Clone and SourceTree will now download the repository.
After a few seconds/mins the new bookmark will show up in the Bookmarks window and you will now have a local copy of the remote repository.

Double-Click the bookmark to open the repository:

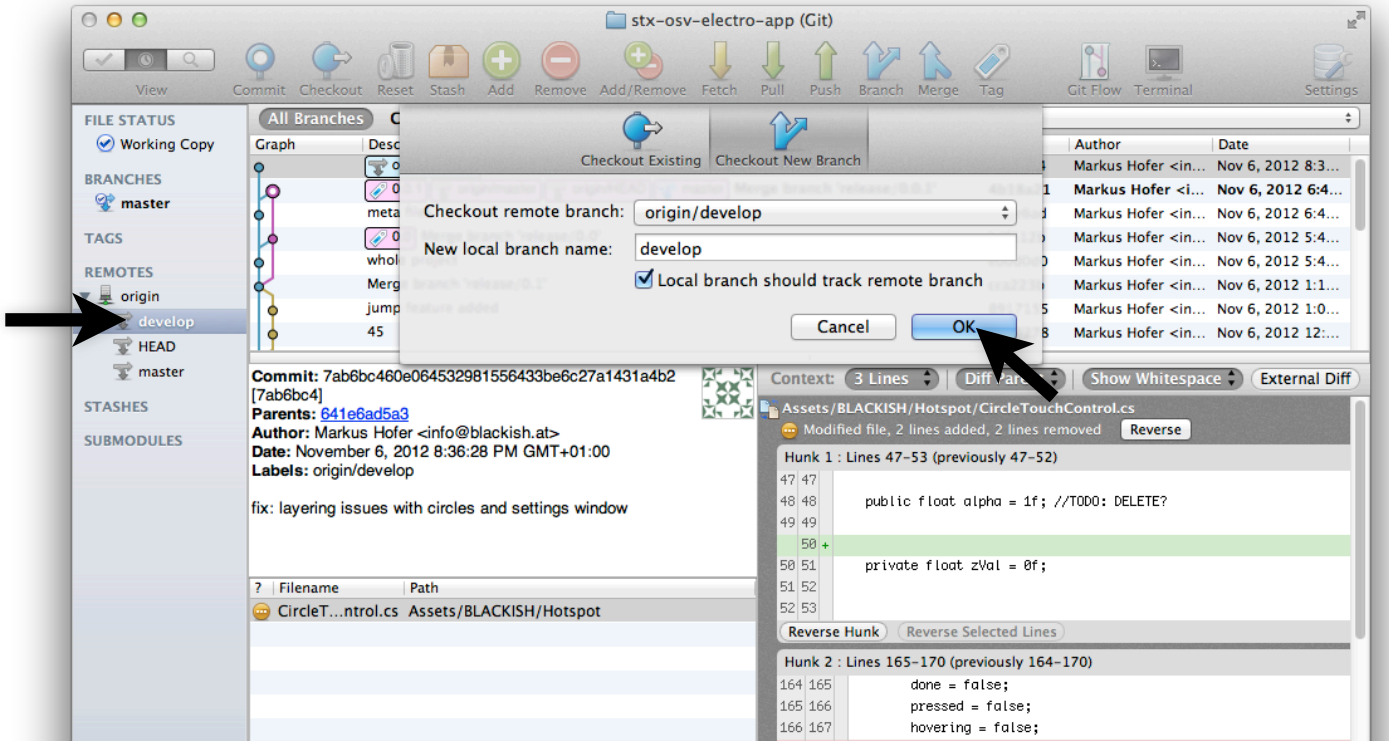


This is what you will see:

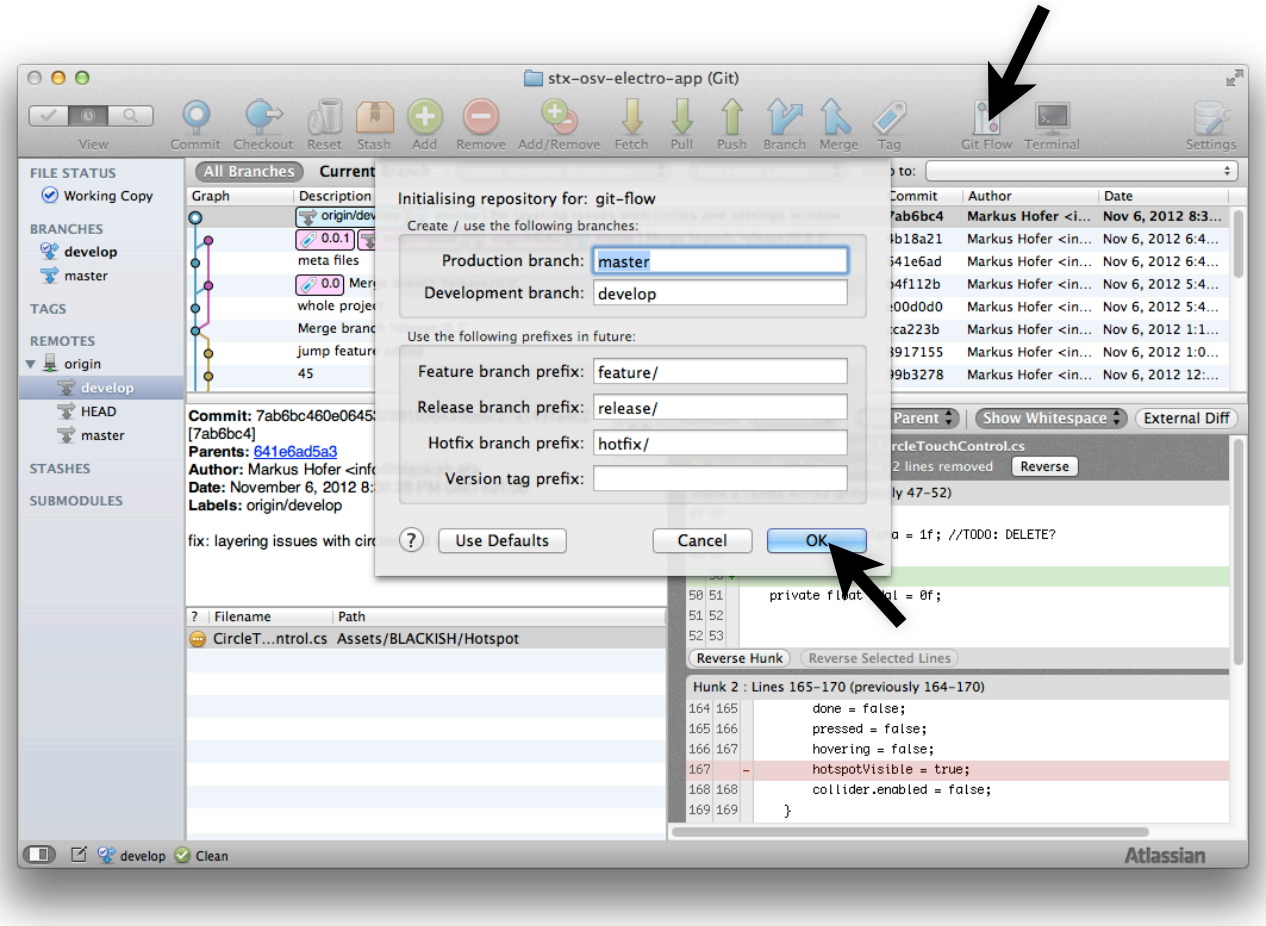


A few things are left to do, but almost done!

Double-click on **origin/develop** to checkout this branch to a new local branch called **develop**.



One last thing:
Click the Git Flow Button and say **OK** to the default settings!

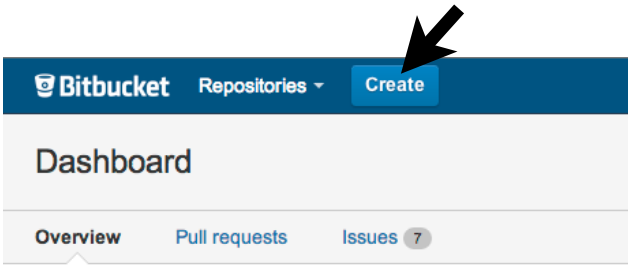


GitFlow is a branching model that I suggest we use. More in the next section!

Or you can read the full story here: http://nvie.com/posts/a-successful-git-branching-model/

Create a new Repository

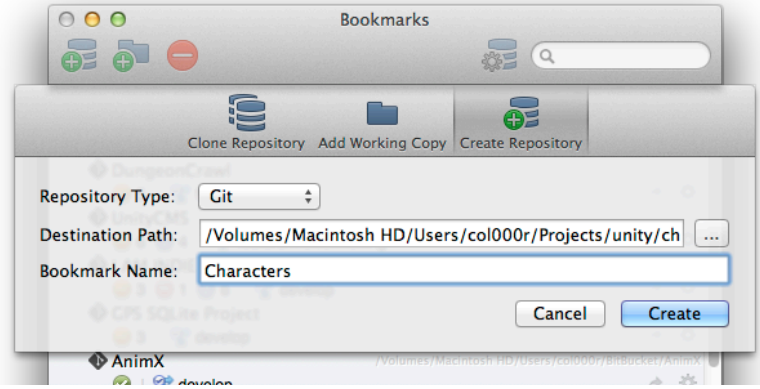
Go to bitbucket.org and create a new repository (use the big Create button on the top)



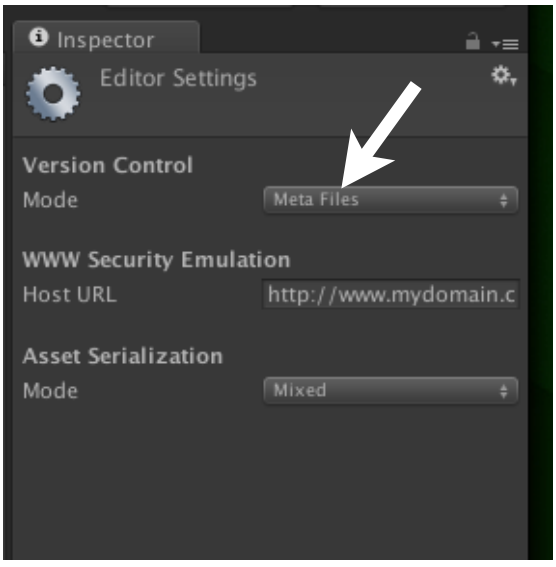
Fill in the name, make it a Git repository and click Create Repository

Back to SourceTree!

Open the Bookmarks window in **Sourcetree** and click Add Repository, select the Create Repository tab and use the **local path** (on your HD!) **of your Unity project** as the Destination Path!



Wait! You did switch on **Meta Files** in **Unity**, right? Open your Unity Project and go to Edit > Project Settings > Editor and switch Version Control Mode to Meta Files!



Create a .gitignore file

The .gitignore file **contains instructions on what git should ignore.**

In other words: All the stuff you don't want to sync, like Unity's cache, builds, temporary files, etc.

Just copy the text on the right into a textfile and save it as ".gitignore" - no file-ending.
Put it into the root directory of your local repository (that's the same directory we used as Destination Path above, remember?)

```
# Unity3D .gitignore file.
#     Store this file at the root of your local repository.
.DS_Store
Library/AssetImportState
Library/AssetServerCacheV3
Library/FailedAssetImports.txt
Library/ScriptAssemblies
Library/ScriptMapper
Library/assetDatabase3
Library/cache
Library/expandedItems
Library/metadata
Library/previews
Library/guidmapper
/Temp
/builds
*.csproj
*.pidb
*.sln
*.userprefs
*.unityproj
InspectorExpandedItems.asset
EditorUserBuildSettings.asset
```

On OS X files that start with a dot are hidden by default. If that makes it difficult for you, open the Terminal and enter the following two lines:

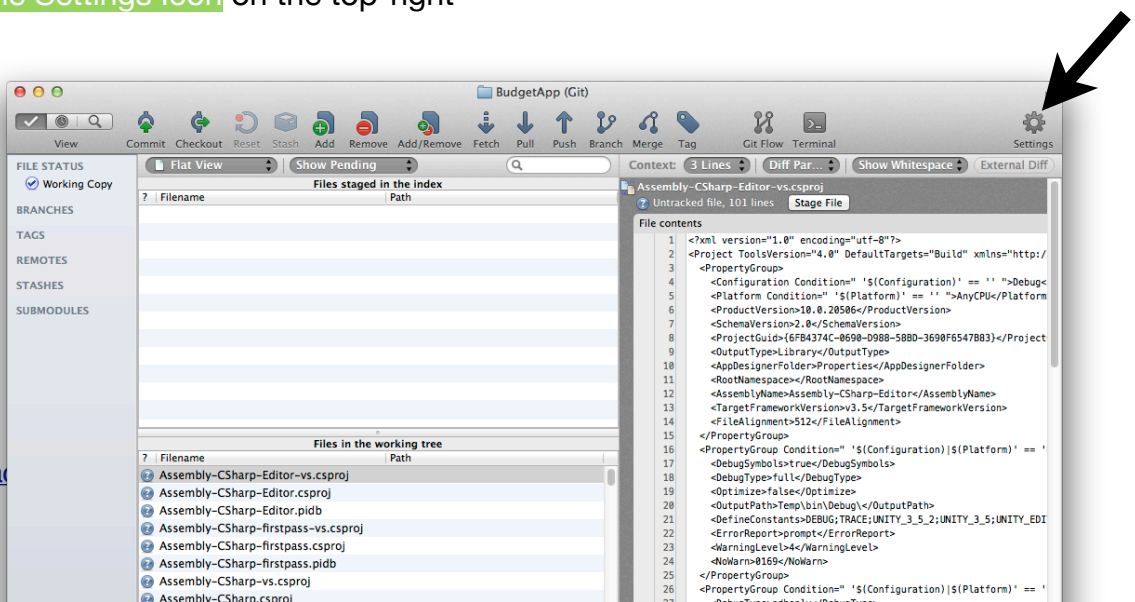
```
defaults write com.apple.finder AppleShowAllFiles TRUE
killall Finder
```

This will restart the Finder and all invisible files will be visible! Do the same with FALSE instead of TRUE to hide them again...

Now we're going to connect our local repository to our new and empty repository on bitbucket.org.

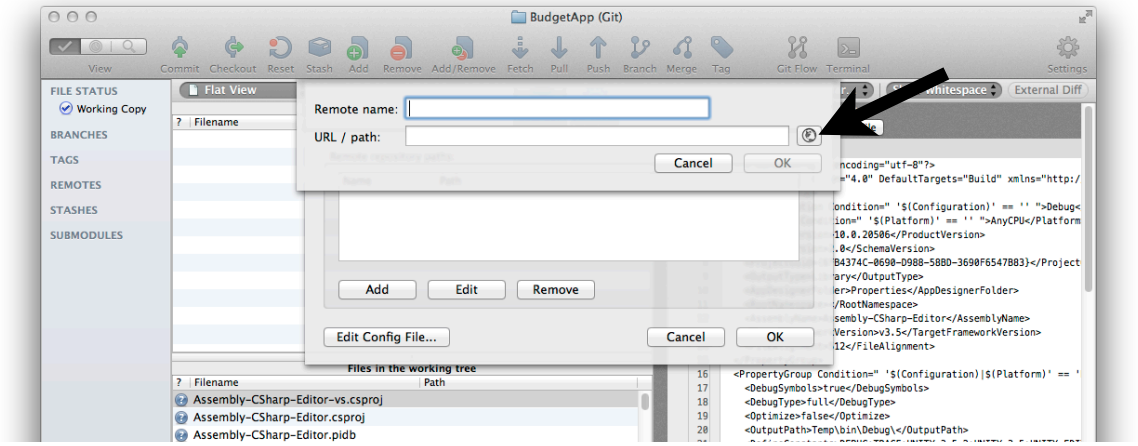
Double-Click the newly created bookmark in the Bookmark Window to open the repository.

Click the Settings Icon on the top-right

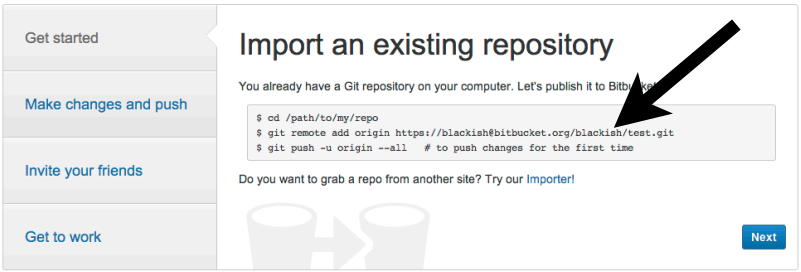


Enter a name for the Remote (for example: “BitBucket”)

Then click the world-icon and select the remote repository we created above.



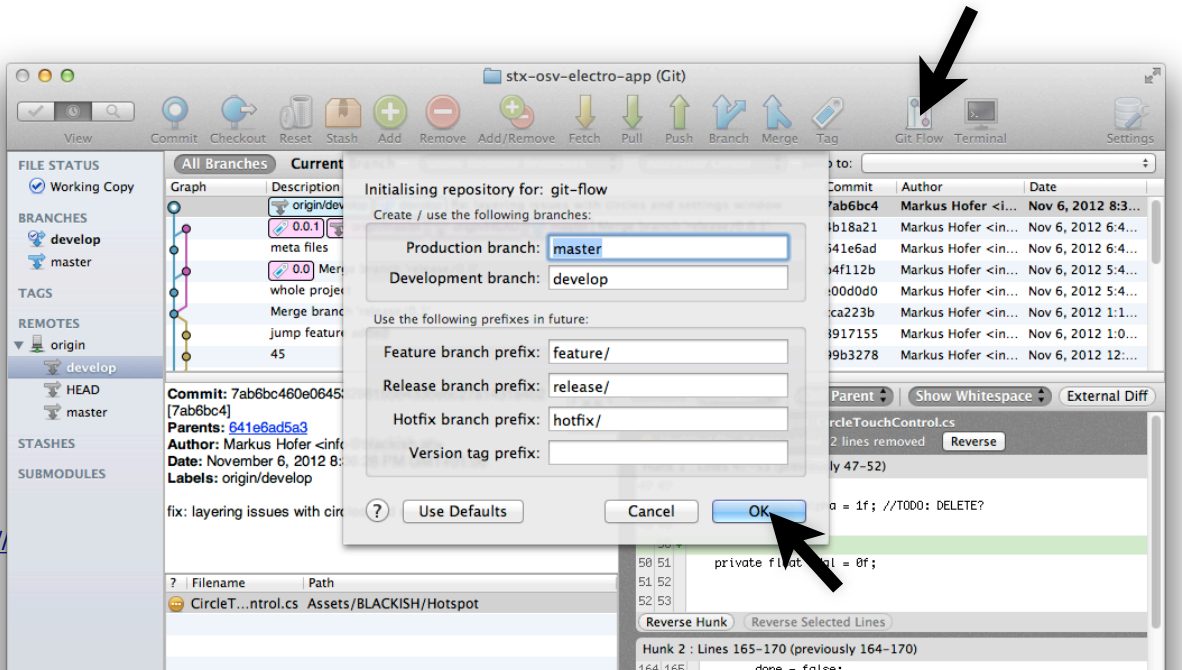
Alternatively you can paste the URL of your bitbucket repository into the URL field (For a newly created repository you can find the URL on bitbucket.org here: Overview > Get Started > I have code I want to import > Look for the https:// url in the grey box)



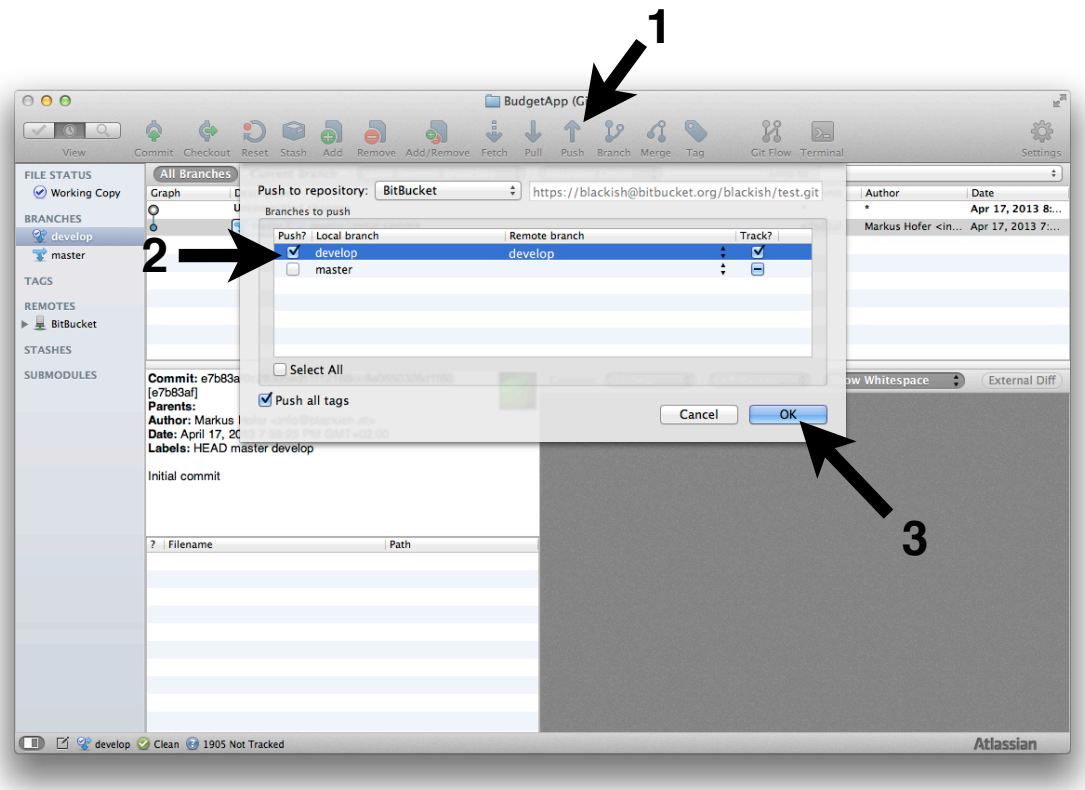
Almost done!

Click the Git Flow Button and say **OK** to the default settings!

This will create all the necessary branches for you and do an initial commit to the develop branch containing all the files in your project. Yes, correct - except the ones we excluded through the **.gitignore** file! You were paying attention, well done!



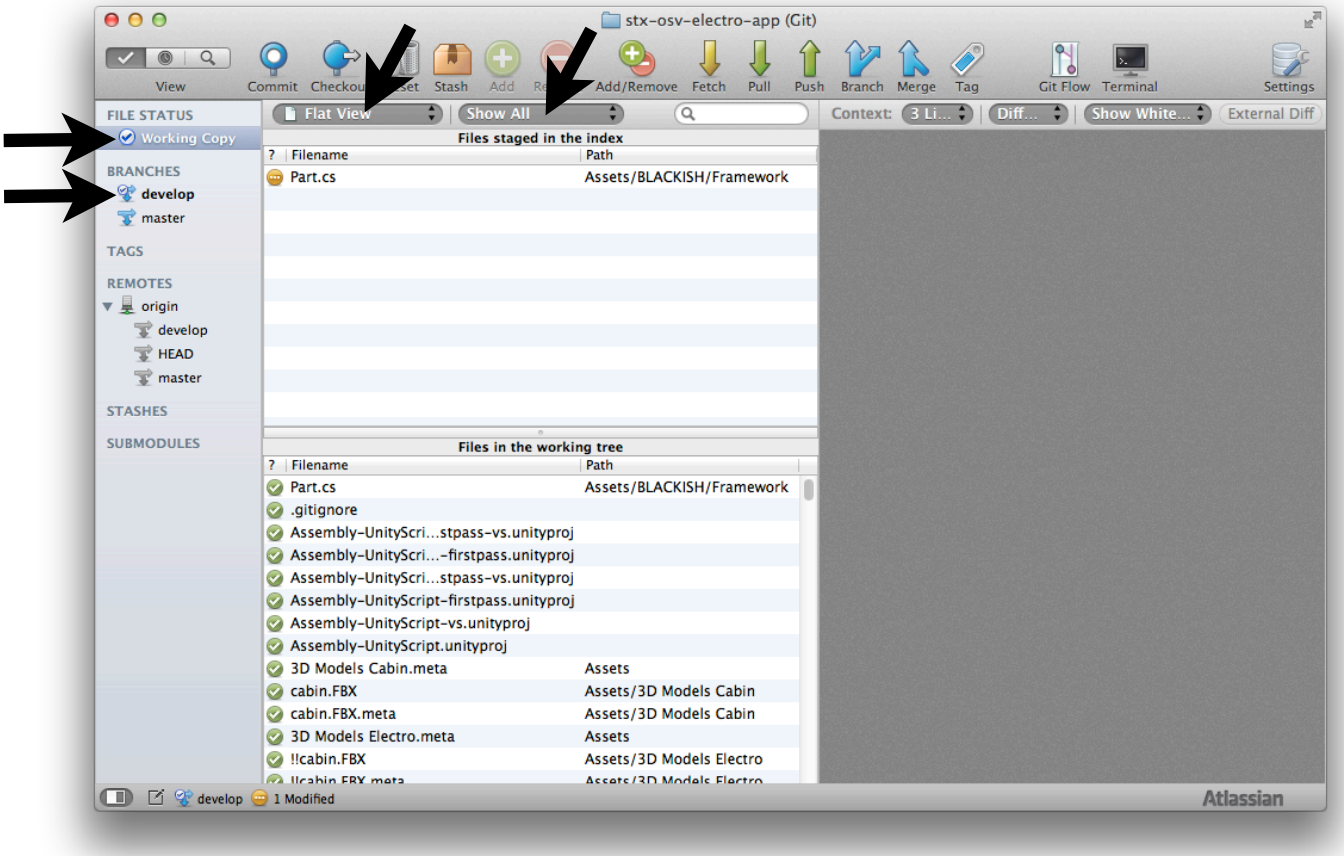
Now click Push, select the branches you want to push (for now we only need to push develop), then click OK and your files will be copied to your remote BitBucket repository!



Daily Use - Most important concepts

WORKING COPY: These are the files that are in your project folder on your HD right now. This is what you're working with at the moment.

The **bold branch** is the **currently active local branch**. This is where the files from your working copy came from. If you change the active branch, the files in your working copy will change! You should be working in the develop branch (or your own new feature branch), NEVER in **master**! Double click on a branch to switch to it (but be aware that this will make you lose uncommitted changes)



FILE LIST:

Notice how you can **change views** and **filter which files to show**!

At the bottom you see a list of all the files (If you set the filter from Show All to Pending you will only see the files that changed since the last commit!)

Above it is the Stage. The stage holds the files that you want to do something with (most likely: commit to the active local branch)

An example of what the stage is for: If you changed a few files they will show up as modified in the bottom list (because they are different from their counterparts in the local branch), you can then drag them up onto the **stage** and **Commit** them to the active local

branch. Only the files on the stage will be updated/added to the active local branch! This means you can commit only certain parts of what you're working on.

Important: This is all still local! These branches are on your computer. Once you want to share your changes with the other users, you need to **Push** your local branch to its remote counterpart.

BRANCHES - What are they for?

master: The latest stable/released/deployed codebase. Never commit directly to this!

develop: The main development branch. Small changes happen directly in here, bigger stuff will be merged in from feature branches. In most cases this should be your active branch!

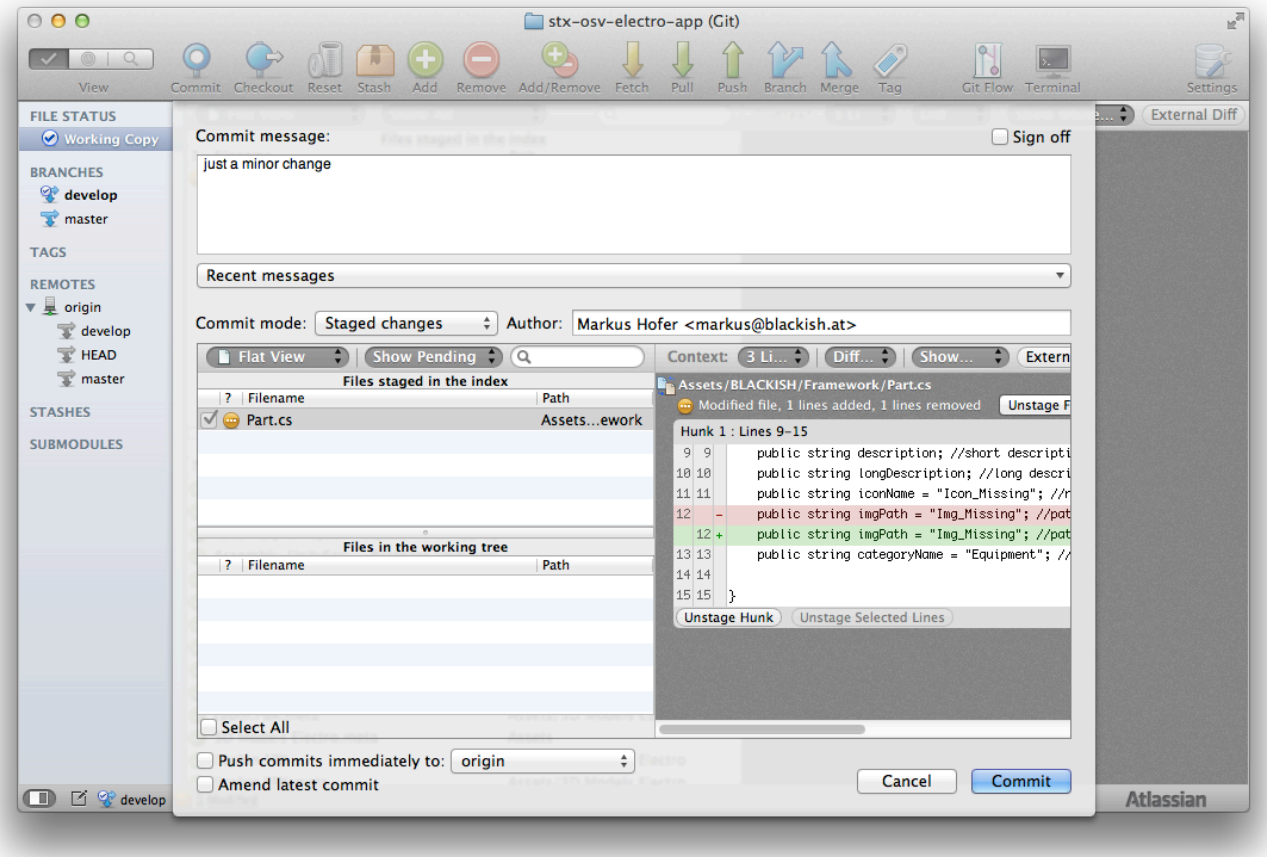
feature/xyz: If you're working on something more complex, use the GitFlow Button to create a new feature and merge it back into develop when it's done.

LOCAL vs REMOTE

Work on your local branch until something works, then push it to the remote branch.

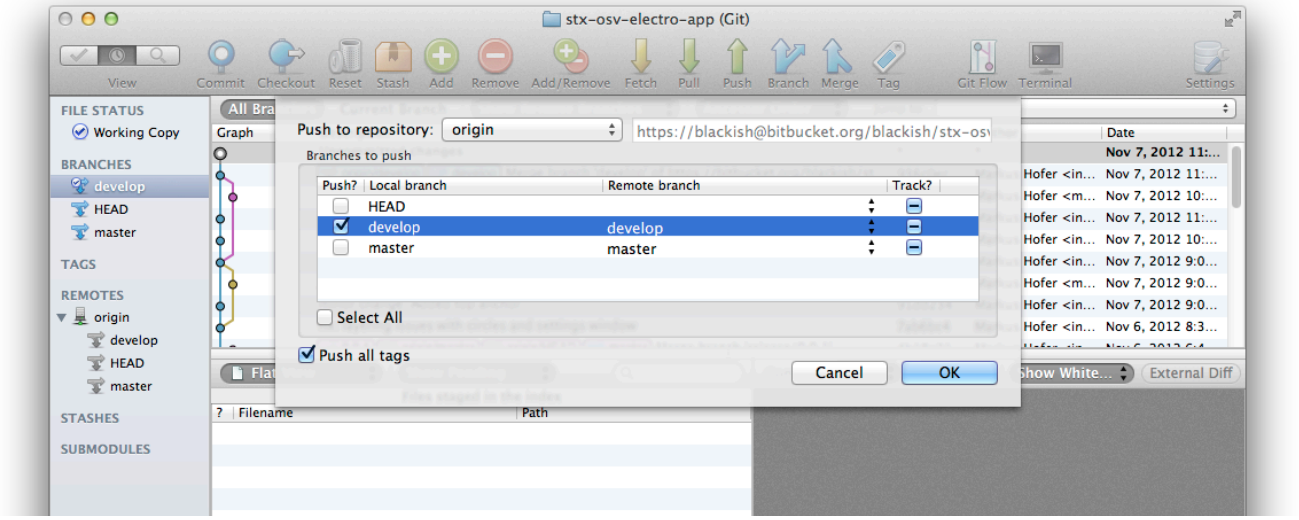
COMMIT

If you change/add local files they will show up as modified. You can stage them and commit the staged changes to your local branch. (You can even stage only certain parts of a file - nice!)



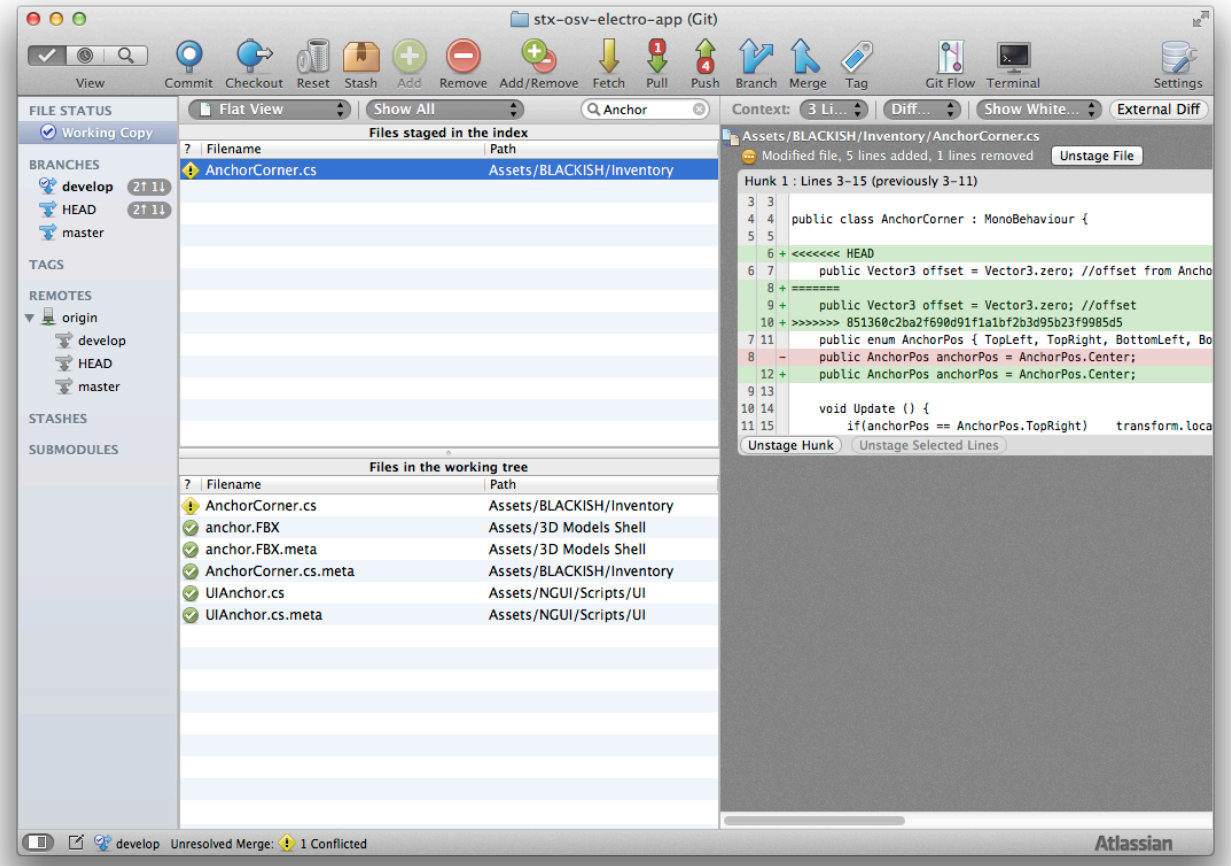
PUSH

Once you're ready to share your accomplishments with the others, **push** your branch to its remote counterpart.



Ouch! Someone else pushed while you were working!

If someone else has updated the remote branch since you've pulled it from the server and added your changes, there might be an error (It might say: `the tip of your current branch is behind its remote counterpart`)!



To solve this do the following:

- Pull the current branch from the server (this will merge the remote branch into your local branch.)
- Resolve conflicts (If someone else changed a file that you have changed also, that's a **conflict.** You will now have to decide what to keep and what to throw out - You have 3 choices, choose wisely!
  A) Try to merge the two files and create the ultimate super-file,
  B) Meh, my changes weren't so great anyway: *right-click on the conflicted file* and choose *Resolve using Theirs*.
  C) My file rocks: *right-click on the conflicted file* and choose *Resolve using Mine*.
  (Because: Fuck 'em, right?)
- Commit the changes (yes, you need to commit the resolved conflicts)
- And finally PUSH the entire thing to the server!

FETCH and PULL

Fetch updates your local copies of the remote branches, but doesn't bring anything into your working copy yet. (Basically just checks if someone else submitted something)

Pull downloads new commits from the remote branch and then tries to merge them into your local branch. This is what you usually want to do to get to the latest version of a project!

TL;DR - I just want the newest version!

Someone else set this up for me, I don't do much, I just want to look at the newest version...

Make sure you're in the local **develop** branch and **Pull!**