**Class tutorial overview -** [**Class wiki site**](https://devtopia/devtopia/git-class/wiki)

**This is how you work with git and GitHub**

The blue highlighted area denotes what exists on your local system, including the commands that are performed locally, and could be performed while disconnected from the server.

**[Initial setup](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia" \l "initial-class-setup)**

**1.** [**Clone a remote repo**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#1-clone-remote-repo)

**2.** [**You need a working branch when you work on a task**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#2-you-need-a-working-branch-when-you-work-on-a-task-1)

**3.** [**Iterative work to do throughout the day:**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#3-iterative-work-throughout-the-day)

* **3.1** [**Change your disk files**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#31-make-changes-to-disk-files)
* **3.2** [**Add changes to your stage index**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#32-add-disk-file-changes-to-the-stage-index)
* **3.3** [**Commit your changes to your branch in the local repo**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#33-commit-your-file-change-into-the-local-git-repo)
* [**Go to 3.1 to iterate on updates made to the local working branch**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#iterate-on-updates-made-to-the-local-working-branch)
* **3.4** [**Push your work into your remote working branch**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#34-push-your-work-into-the-remote-working-branch)
* [**Go to 3.1 to iterate on your updates made to the remote working branch**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#iterate-on-updates-made-to-the-remote-working-branch)

**4** [**Periodically merge your changes from the remote master branch to your local working branch**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#4-periodically-merge-changes-from-the-remote-master-branch-to-your-local-working-branch)

**5** [**When done working on your task use a Pull request to merge your remote working branch into the remote master**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#5-when-done-use-a-pull-request-to-merge-the-remote-working-branch-into-the-remote-master)

* **5.1** [**Create a Pull Request for your work**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#51-create-a-pull-request)
* **5.2** [**Process the Pull Request**](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#52-process-a-pull-request)

**[\*\*\*\* Additional Git Documentation and Tutorials \*\*\*\*](https://devtopia/devtopia/git-management/wiki/Git-Documentation-and-Tutorials)**

**Initial class setup**

**You will have push access to the production repos that you work in, so normally you will simply clone your repo. You do not have push access to the** [**devtopia/git-class**](https://devtopia/devtopia/git-class) **repo, so we will first *fork* the git-class template repo to your devtopia account. Once there, you will have push access to it. *A fork is a copy of a GitHub repo to another location on GitHub.*** [**More information on Cloning and Forking is available here**](https://devtopia/devtopia/git-management/wiki/devtopia-101-Only-fork-if-you-don%27t-have-push-access-to-a-repo)

1. Start by going to <https://devtopia/devtopia/git-class> and pressing the button in the upper right corner of the screen. In the "Where should we fork" dialog, select your @ *accountname*. Go to your [user account](https://devtopia/) and select the git-class repo that you just forked from the **Your repositories** section.
2. Before we make the working branch, we need an issue in the repo on devtopia to work against. In production repos the issues will already be created and assigned to milestones and to you by using the instructions described at the [issue-management](https://devtopia/devtopia/issues-management/wiki) wiki. Your git-class repo does not contain any issues, so you will need to create one. Start by pressing the “github issues” button. If you don't see the “issues button ” button on the right side of the page, press the “settings button” button, and check on the  checkbox. Once on the issue page for your git-class repo press the “github new issue button” button. In the *Title* field enter *"change text to Hello esri"*, assign the issue to yourself, and press the “github submit new issue button” button to complete the issue create process.
3. Copy the git-class repo URL by selecting the *copy* button highlighted here “Github copy repo ssh” on the lower right side. Make sure the URL is set to **SSH**, if not select the blue SSH text just below the button to switch to the git@ URL prefix.
4. Next open SourceTree or your git GUI or IDE that supports git.

**1. Clone remote repo**

**A clone is a copy of an existing repo to a local disk workspace. The local disk workspace will get a copy of the repo into a .git folder, including all branches. It will also automatically checkout the source files and folders from the default master branch. This is shown in the blue box in the image at the top of the this page. *Because these resources are local, you could perform steps 2.2, 3.1, 3.2, 3.3 of the class disconnected from the network***

In SourceTree press the “SourceTree Clone Button” button and paste the git SSH URL for your fork of the git-class repo on devtopia into the *Source Path / URL:* field in SourceTree. Click on the *Destination Path:* and it will automatically be set, and acknowledge that *This is a Git repository*. Then press the **Clone** button at the bottom.

[clone git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#1-clone-remote-repo).

open the index.html file in your browser to verify. You should see:

Class index\_html file

If you select the “log history tab ”tab at the bottom of the page you should now see near the top of the page the following information listed.

Existing branches

This is normal and simply means that there are other branches that exist in the repo where work is being done but the work has not yet been merged from the working branch in the master branch.

**2. You need a working branch when you work on a task**

You will always make file changes in a working branch, **never the master branch**. The master branch is only updated via a Pull Request, which is processed on the GitHub server. Pull Requests are covered in [section 5](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#5-when-done-use-a-pull-request-to-merge-the-remote-working-branch-into-the-remote-master). So if you haven't created a working branch for your task already, you will need to create one now.

**2.1. Now create the branch**

Branch names should include your name/issue number-a short description. This allows others to determine what the branch is for and who is responsible for it without having to open the branch and inspect it. In SourceTree press the “SourceTree Branch Button” button and for the issue we will be addressing, enter <YourLogin>/Issue1-esriText in the *New Branch:* field. Then press the **Create Branch** button. Setting the branch name to your login, the issue number and what you are working on makes it clear what the branch is for to everyone. [create working branch git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#21-create-working-branch-when-you-start-a-task).

**2.2. Now push the branch to the repo on devtopia as a public backup**

In SourceTree press the “SourceTree Push Button” button and check the box to the left of the <YourLogin>/Issue1-esriText branch. Then press the **OK** button. [push working branch git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#22-now-push-the-branch-to-the-repo-on-devtopia-as-a-public-backup).

If you select the “log history tab” tab at the bottom of the page you should now see at the top of the page the following information listed on the last git commit

sourcetree branch state information:

* origin = the remote repo on devtopia
* master = the master branch
* <YourLogin>/Issue1-esriText = the working branch
* origin/HEAD = a reference to the last commit in the master branch in the remote repo on devtopia origin/<branch\_name> = branches in the remote for other work

These labels allow you to understand the state of the different branches you are tracking. The current state is that *your branch*, the *master branch*, both locally and on devtopia, as well as the *last commit on the remote repo* all reference the same thing. The other branches in the remote contain committed changes that have not yet been merged into origin/master, and therefore are ahead of origin/master. As you commit changes locally, and/or as changes get made in the remote repo you will be able to track the state of each branch through these labels. At this point we have an GitHub issue to work on, and a local working branch to work in, as well as a public location to save the work in your working branch on devtopia. It's time to do your iterative work.

**3 Iterative work throughout the day**

**Checkout branch to context switch in your workspace**

At this point if you double left click any of the four branches it will checkout the content of the branch into your workspace, you can refresh your internet browser that displays the index.html file changes to the file content of that branch.

[checkout local branch git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#checkout-branch-to-context-switch-in-your-workspace).

* master or <YourLogin>/Issue1-esriText - shows a white background with *Hello World* text “Class index\_html file”
* red - shows a red background with *Hello World* text “Class index\_html red” file
* blue - shows a blue background with *Hello World* text “Class index\_html blue” file

What git provides you locally is a single workspace for working with a repo, and the ability to very quickly context switch to the content of a specific branch so that you can view and work on the files for a specific task.

**3.1 Make changes to disk files**

Make sure to context switch to the <YourLogin>/Issue1-esriText branch. Then edit the index.html disk file. You will want to change the *Hello World* text field on line 8 to *Hello esri* to make the changes to address your issue. Save your changes. The index.html file should now look like:

Class index\_html flie

**3.2 Add disk file changes to the stage index**

In SourceTree select the “file status tab” at the bottom of the page and you should notice “index.html change” in the Working Copy Changes section. Now left click this changed index.html disk file and drag it into the Staged Changes section. If you switch back to the “log history tab” tab you will note that you now have **Uncommitted changes**. This means that the modification is now ready to be committed into working branch in your local repo.

[add files to stage git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#32-add-you-disk-file-change-to-the-stage-index).

**3.3 Commit your file change into the local git repo**

In SourceTree select the “commit button”. This will open a commit dialog where you can review your changes. When ready, enter the commit message *change text message to Hello esri*, and press the **Commit** Button at the bottom of the dialog.

[commit files to local branch git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#33-commit-your-file-change-into-the-local-branch).

In **Log/History** tab you should now see that your local <YourLogin>/Issue1-esriText label is *1 ahead* of the other labels, and there is a “push 1” available to push to the remote.

**Iterate on updates made to the local working branch**

Go back up to [3.1](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#31-make-changes-to-disk-files) if needed and continue editing, adding changes to your stage, and committing your work locally.

If you are done working locally for now, we can can process the push now, or if we need to we can do more work and commit it locally in our branch until we are ready to push the commits in our local working branch to the remote working branch.

**3.4 Push your work into the remote working branch**

When you have finished working on the issue, or have come to a point where you want to move the work you committed into the local git repo to devtopia as a backup and/or for others to see it, press the “push 1” at the top of the screen. Check the box next to your branch name only and press the **OK** button.

[Push your work into the remote working branch git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#34-push-your-work-into-the-remote-working-branch)

This will push the changes that you have made in your local <YourLogin>/Issue1-esriText branch into the origin/<YourLogin>/Issue1-esriText branch in the git-class repo on devtopia.

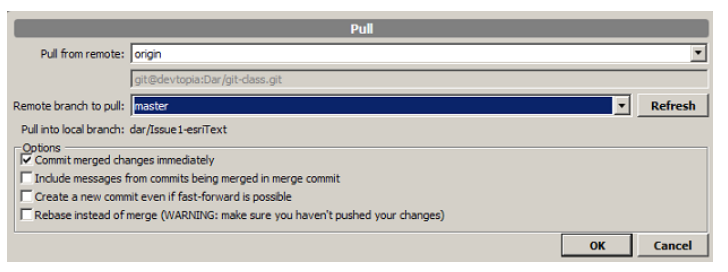
**Iterate on updates made to the remote working branch**

Go back up to [3.1](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia#31-make-changes-to-disk-files) if needed and continue editing, adding changes to your stage, committing your work locally, and pushing it up to the remote repo until your are done working on your issue.

**4. Periodically merge changes from the remote master branch to your local working branch**

Periodically you will want to check the state of the origin/master branch to see if commits have been made to it that you do not have in your local working branch. These will appear in SourceTree in a couple of places. For example you will see a number next to the **master** branch “… icon …”, and in the repo list a number like “tick dar/Issue1-esriText down-arrow” when commits have been made to master for you to pull. If there are changes, you will want to consider merging these commits to your local working branch.

Pull the remote origin/master by pressing “pull 1” pull button. Set the Remote branch to pull: to master. Because your current working files are the ones in your working branch, you will notice the Pull into local branch: is already set to origin/<YourLogin>/Issue1-esriText for you.



Press the **OK** button to complete the merge.

If there are merge conflicts, you will need to resolve and add them to your staged index. Then commit the merge into the local repo manually. The commit message for the merge commit will already be filled in for you.

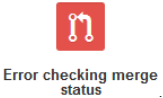
[pull origin master into local master git bash shell commands](https://devtopia/devtopia/git-class/wiki/git-class-for-working-in-devtopia-git-commands#41-pull-origin-master-into-the-local-master)

**Note:** you can also pull the origin/master to the local master if you are going offline and want to have the current origin/master commits available to you. If you don't want to checkout the master branch locally to do it, open a Terminal window and type git pull origin master:master.

**5. When done working on your task use a Pull request to merge the remote working branch into the remote master**

At this point you are all done with your work locally, it's been tested, there are no more iterations required and your working branch at devtopia, origin/<YourLogin>/Issue1-esriText, is current. Now it's time to get your changes into the remote master branch so that it is part of product. You will need to submit a Pull Request to perform this operation.

**5.1. Create a pull Request.**

In your browser, go to https://devtopia/<your\_account>/git-class/. Press the v arrow in and select the <YourLogin>/Issue1-esriText branch. Below your selection you should see a branch state that reads *This branch is 1 commit ahead and 0 commits behind master*. Directly to the right press the grey **Pull Request** button, not the Blue one. This will open the Pull Request dialog, but the merge check might look like . Next press the **Edit** button at the top right and set the base fork to your\_login/git-class if it is not set already. When it is set properly, the merge check should look like .

You should add a reference to the issue you are addressing in the Pull Request submit message. Simply add the text **Issue /#** for this git-class example it will be: <YourLogin>/git-class#1. This will create a bi-directional reference between GitHub Issue and the Pull Request that addresses it.

If you are submitting a Pull Request to another developer on your team, then before pressing the  button, it is recommended that you add the text Assigned to: @<account\_name> of the person on your team that you will assign the pull request to. This will let the other team members know they can look at the pull request only if they want to. Once you add the mention, you can press the  button. When it has been created, assign the Pull Request.

**Note:** If you don't know an persons account name, type @ and the first few characters of their name in the search dialog at the top of the page. You should see a list of accounts that includes the person you are looking for.  
**Additional Note:** In general it's a very good idea to @mention people in Pull request and GitHub Issues. It vastly increases communication, planning, and productivity.

**5.2. Process a pull Request.**

If you go back into your forked git-class repo you should now see a  button with a 1 after it. This means that your repo has 1 open pull request. Press the  button and select your pull request to start to process it.

Next to each commit in the pull request in the Pull Request is a commit hash URL that when pressed shows the diff for the files in that commit. If you move your cursor over the diff lines, you will see a blue callout box to the left with a white + in it. If you press this callout, you can create a Pull Request comment for that line change. Notifications will be generated for all comments so that the people involved can act upon them.

If there are issues with the code in the Pull Request, the developer has the opportunity to go back to their local workspace, fix the issue, commit it, and push the fix back to the branch on devtopia. When this happens the existing Pull Request will automatically be updated with the additional modification.

There are lots of things you can communicate in a Pull Request. Here is an [example](https://devtopia/apps-group/collector-android/pull/381)

Once everyone is satisfied with the state of the Pull Request, the person that is assigned will press the  button and then the  button.

The working branch on devtopia can be deleted by pressing the  button.

**Note:** You can recover the deleted working branch on devtopia if you return to the pull request and press the **Restore** link at 

**More information** on Pull requests is available in the **Collaborating** section of the [GitHub Help](https://help.github.com/categories/63/articles)

**Additional Note:** If you reject the Pull Request and close it, you can still delete the branch, but the button will be grey instead of blue.