

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10210 NE Points Drive, Suite 200 Kirkland, WA 98033 |  |  |  | [Web address] |

https://dev1.allyis.com:9443/Bonobo.Git.Server/Home

**git Usage & Setup**

Allyis Apps

Table of Contents

1. [Prerequisite Tools 2](#_Toc457479461)

[git](#_Toc457479462)

[KDiff3](#_Toc457479463)

[SourceTree](#_Toc457479464)

1. [Creating a Bonobo User 13](#_Toc457479465)

[What is Bonobo, and where is it?](#_Toc457479466)

[New Accounts](#_Toc457479467)

[Joining a Team](#_Toc457479468)

[Joining a Repository](#_Toc457479469)

[Retrieving Your Personal Repository URL for git](#_Toc457479470)

1. [Cloning The Golden Repository 15](#_Toc457479471)

[Adding the Bonobo Server to SourceTree](#_Toc457479472)

[Cloning Guide](#_Toc457479473)

[Optional: Instead, Creating a New Local Repo and Pulling](#_Toc457479474)

1. [git Actions 18](#_Toc457479475)

[Creating a Branch](#_Toc457479476)

[Creating a Commit](#_Toc457479477)

[Synchronizing Branches with Bonobo](#_Toc457479478)

[Merging Branches](#_Toc457479479)

[Feature Collaboration](#_Toc457479480)

1. [General git Rules 19](#_Toc457479481)

[Never Re-Write Golden Master History](#_Toc457479482)

[Merge Style](#_Toc457479483)

1. [Extra Niceties 19](#_Toc457479484)

[Creating a Gravatar](#_Toc457479485)

# Prerequisite Tools

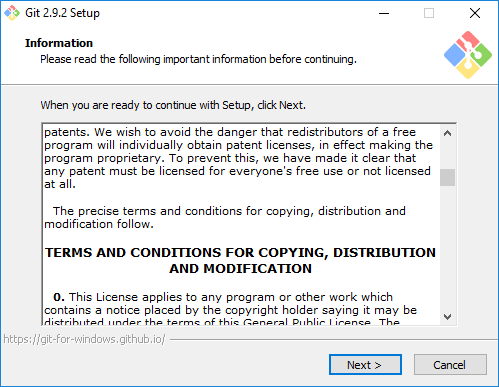
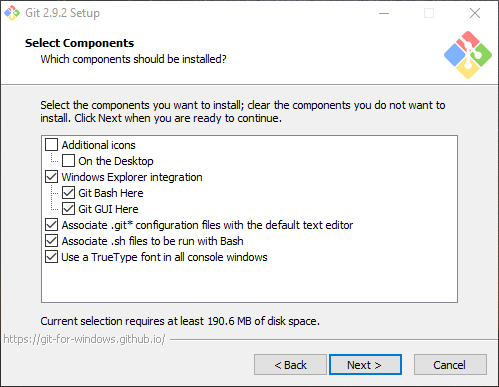
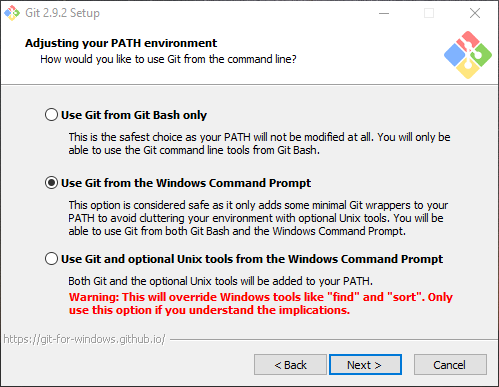
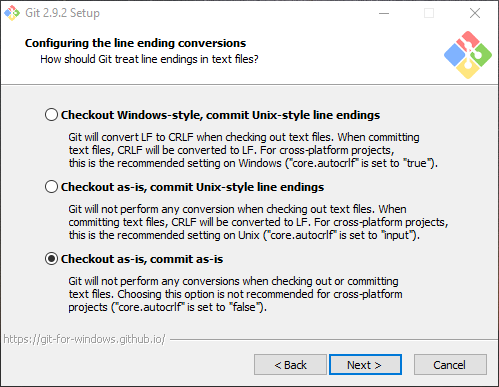
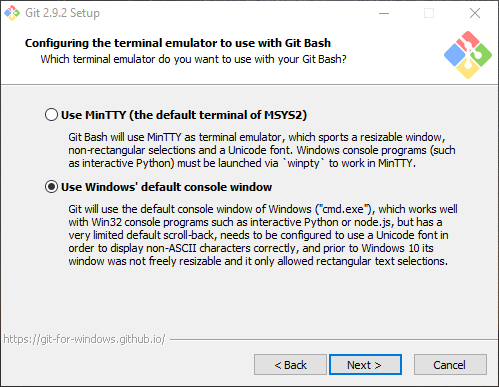
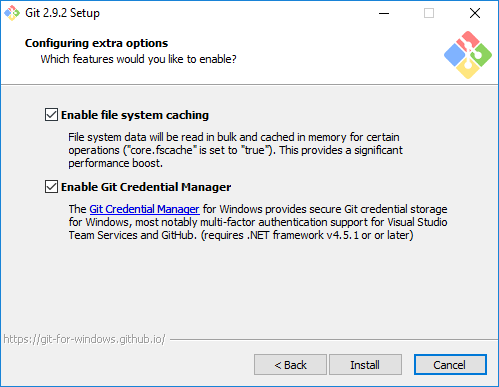
|  |  |
| --- | --- |
|  | It will be assumed throughout this document that the reader is using the following tools. If you do not already have them installed, this section will guide you through the process of installing and configuring the tools with the Allyis Apps standard configuration. |
|  | If you already have the tools installed, it is highly suggested that you verify their configuration matches what is described here. Check the Gotchas sections for configuration setting overviews. |

## git

### Download Location

The installation files can be found on the [git-scm website](https://git-scm.com/download/win). Be sure to download the current 64-bit installer. At the time of writing, the current installer is “Git-2.9.2-64-bit.exe”.

### Installation Guide

1. Run the installer and accept the license shown by clicking “Next >”.
2. On the “Select Components” screen, make sure the selected components match what is shown in the following screenshot. Specifically, make sure the boxes under “Windows Explorer integration” are checked. They should be pre-checked by default. Click “Next >”.
3. On the “Adjusting your PATH environment” screen, ensure the selected option is “Use Git from the Windows Command Prompt” as shown below. Click “Next >”.
4. On the “Configuring the line ending conversions” screen, ensure the selected option is “Checkout as-is, commit as-is” as shown below. Click “Next >”.  
   
5. On the “Configuring the terminal emulator to use with Git Bash” screen, ensure the selected option is “Use Window’s default console window” as shown below. Click “Next >”.
6. On the “Configuring extra options” screen, ensure the selected options match what is shown below. Click “Install”.  
   

#### Gotchas

1. Refer to the photo in step 4 of the installation guide. Did you set up git to “**Checkout as-is, commit as-is**”? We do not want git automatically making any whitespace changes, especially as we are all developing on Windows boxes. You will find yourself touching files in commits without meaning to if you neglect to select this option.  
     
   You can edit this setting after installation by using the following command in a git bash:

|  |  |
| --- | --- |
|  | git config --global core.autocrlf false |

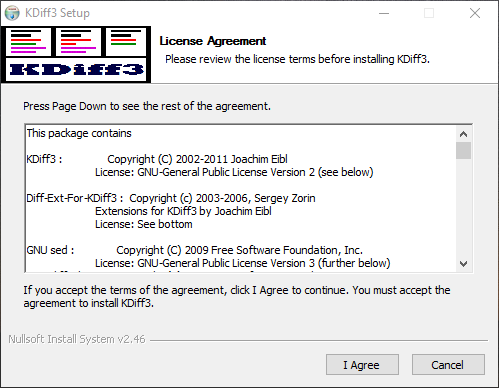
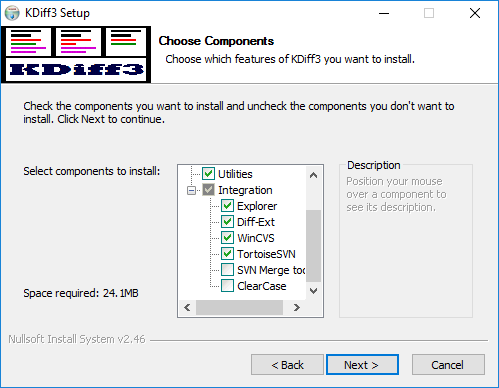
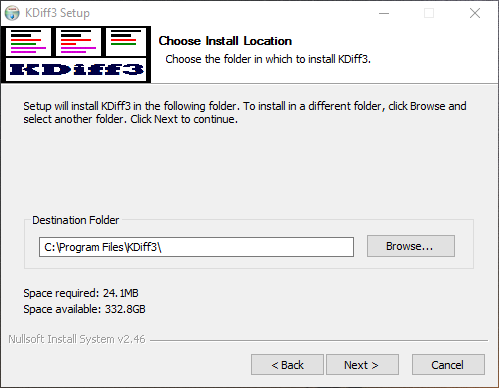
## KDiff3

|  |  |
| --- | --- |
|  | You may use a different diff tool if you prefer, but it is highly recommended you use KDiff3 if you have no strong feelings on the matter. This way, if and when you need help, your fellow developers will be familiar with the tools on your machine. |

### Download Location

The installation files can be found on the [project's sourceforge page](https://sourceforge.net/projects/kdiff3/files/). Be sure to download the current 64-bit installer. At the time of writing, the current installer is “KDiff3-64bit-Setup\_0.9.98-2.exe”.

### Installation Guide

1. Accept the license by clicking “I Agree”. 
2. On the “Choose Components” screen, ensure the selected option match those shown below. Click “Next >”. 
3. Select your destination folder. The default is fine. Click “Next >”, “Next >” again, and “Install”. 

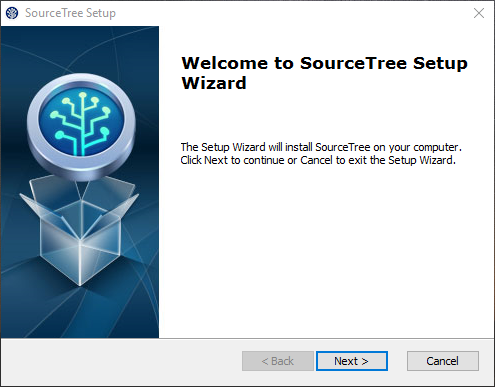
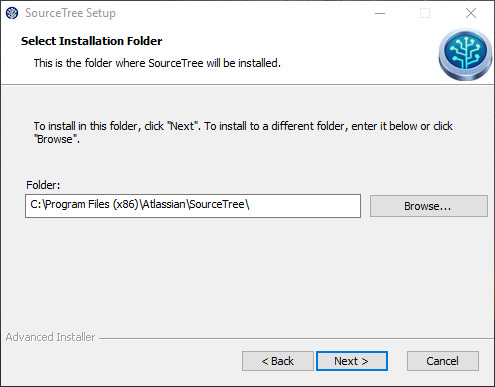
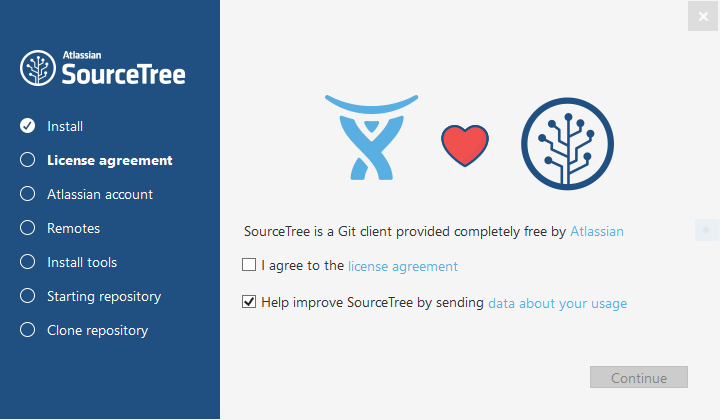
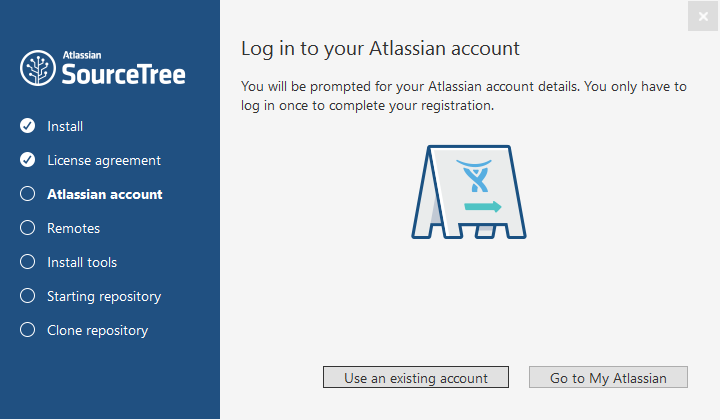
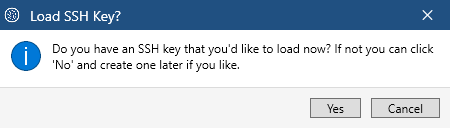
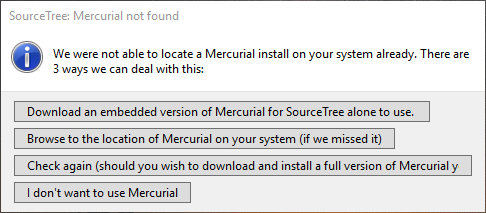
## SourceTree

|  |  |
| --- | --- |
|  | You may use a different git tool, or none, if you prefer, so long as you adhere to the conventions outlined in the “git Actions” section of this document. |
|  | However, it is necessary that you have SourceTree installed and configured, even if you do not use it, so as to make the inevitable situation of seeking of help from your fellow developers easier.   Moreover, the reader should note that this document assumes usage of SourceTree. Following along with any instructions will be complicated if you do not use this tool. |

### Download Location

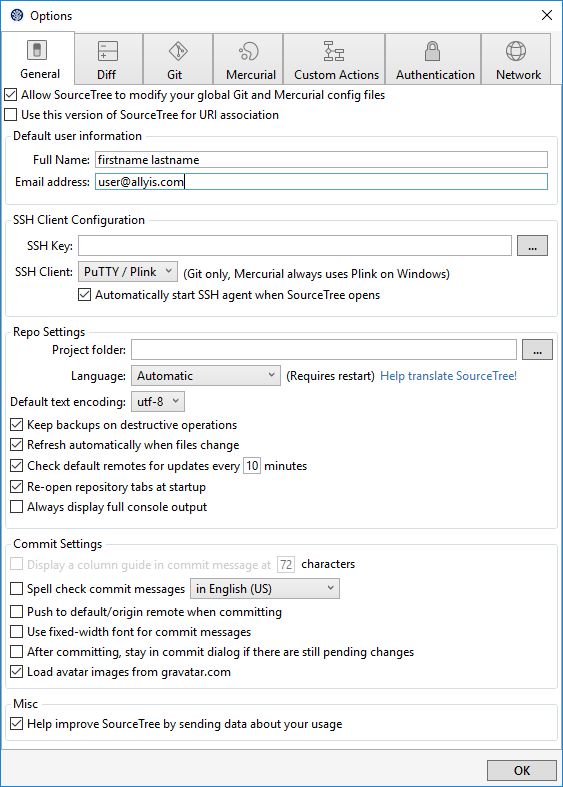
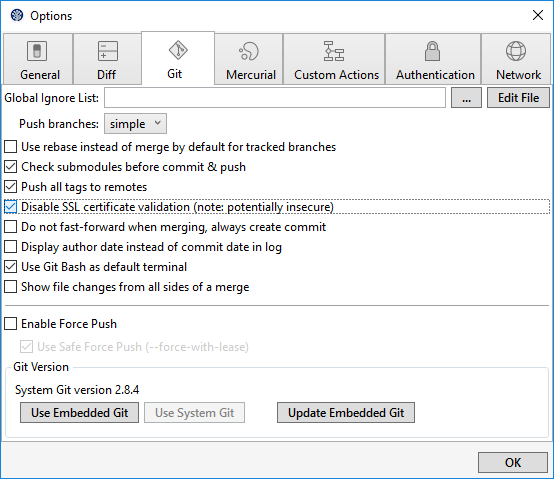
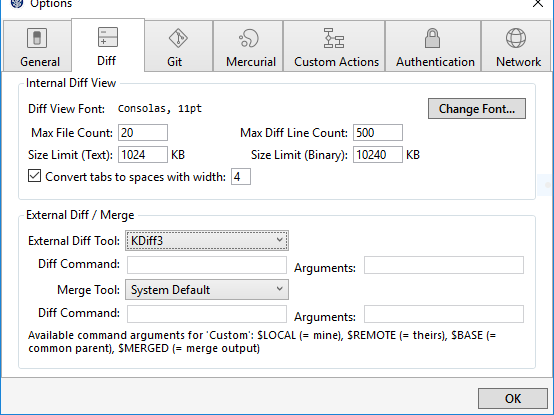
The installation files can be found on the [Atlassian's website](https://www.sourcetreeapp.com/). Be sure to download the current 64-bit installer. At the time of writing, the current installer is “SourceTreeSetup\_1.9.5.0.exe”.

### Installation Guide

1. Run the installer. You will be greeted with the screen shown in the following screenshot. Click “Next >”.  
    
2. Select your installation folder. Click “Next >” then click “Install”. 
3. Upon first launch, you will be prompted with the following screen. Check the box to agree to the license agreement and click “Continue”.
4. You will need to create an Atlassian account if you do not already have one. Click “Go to My Atlassian” to do so. Othewise, click “Use an existing account”. 
5. On the “Remotes” page, just click “Skip Setup”. You will be prompted to load an SSH Key. Click “Cancel”.   
   
6. You will then be prompted with the following screen. Click “I don’t want to use Mercurial”. 
7. Installation is now complete and the app will launch.

### Post-Installation Setup

|  |  |
| --- | --- |
|  | If you do not already have a Bonobo User set up for yourself, head to [Creating a Bonobo User](#_Creating_a_Bonobo) to do so. Then come back to this section to complete SourceTree setup. |

1. Open Tools>Options. Fill out the “Default user information” section with your name and email. Click “OK” to save.
2. Under the Git tab, make sure the Disable SSL certificate validation option is checked. Otherwise git will not be able to communicate with Bonobo. 
3. Under the Diff tab, make sure the External Diff Tool dropdown is set to KDiff3. 
4. SourceTree is now setup for use with the Allyis Apps repository on our Bonobo server. For instructions on how to clone the repo, head to the [Cloning The Golden Repository](#_Cloning_The_Golden) section of this document.

#### Gotchas

|  |  |
| --- | --- |
|  | Make sure SourceTree is set up to Disable SSl Certificate Validation. Refer to step 2 of the post-installation setup guide for more information. |

# Creating a Bonobo User

## What is Bonobo, and where is it?

|  |  |
| --- | --- |
|  | Bonobo is the Source Control server our golden repo lives on. It is located at <https://dev1.allyis.com:9443/Bonobo.Git.Server/Home>. You should consider bookmarking this address in your browser. |

## New Accounts

|  |  |
| --- | --- |
|  | Creating a new account requires logging in to Bonobo with the admin account. You can get the credentials from your team lead or have them help you with the following steps. |

1. Log in to the Bonobo server with the admin credentials.
2. Click on Users, and then on that page, click “Create new user”.
3. Fill out the information on the “Create New Account” page.
   1. The username convention is to use your Allyis email username.
4. Click “Create” to create the new user.

## Joining a Team

|  |  |
| --- | --- |
|  | Joining a Team requires logging in to Bonobo with the admin account. You can get the credentials from your team lead or have them help you with the following steps. |

1. Log in to the Bonobo server with the admin credentials.
2. Click on Teams, and then click the edit icon for the team you would like to add the user to.
3. Click the checkbox next to the user you would like to add under the Contributors section, and then click “Save”.

## Joining a Repository

|  |  |
| --- | --- |
|  | Joining a Repository requires logging in to Bonobo with the admin account. You can get the credentials from your team lead or have them help you with the following steps. |

1. Log in to the Bonobo server with the admin credentials.
2. Click on Repositories, and then click the edit icon for the Repository you would like to add the user to.
3. Click the checkbox next to the user you would like to add, and then click “Update”.

## Retrieving Your Personal Repository URL for git

1. Log in to the Bonobo server with your personal account.
2. Click on the repository you need a Personal URL for.
3. Copy the URL listed next to Personal URL. It will be of the form: https://{username}@dev1.allyis.com:9443/Bonobo.Git.Server/AllyisApps.git

# Cloning The Golden Repository

|  |  |
| --- | --- |
|  | “Golden Repository” is the term used to refer to the repository on the Bonobo server. Similarly, “Golden master” is the term used to refer to the master branch on the Golden Repository. The distinction is drawn to highlight the difference between your local repository and master branch, and the “true” repository and master branch. |

## Adding the Bonobo Server to SourceTree

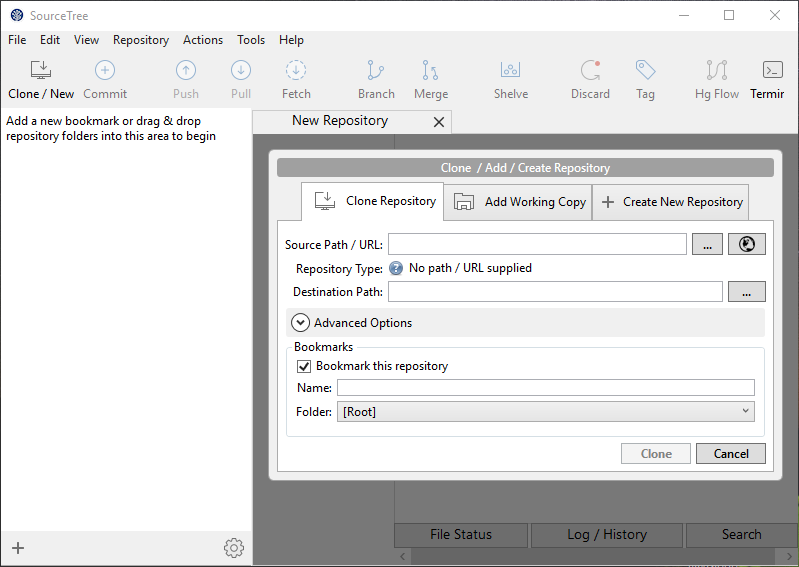
|  |  |
| --- | --- |
|  | If you do not already have a repository of any kind set up in SourceTree, it will not allow you to edit Repository settings, which is where you would add the Bonobo server as a remote. You must first create an empty local repository, or clone the Bonobo remote repository. |

1. Retrieve your Personal Repository URL. If you need help with this, follow the instructions in the [Retrieving Your Personal Repository URL for git](#_Retrieving_Your_Personal) section of this document.
2. Go to Repository>Repository Settings, and under the Remotes Tab, click Add.
3. Name the remote something like Bonobo or Dev1.
4. For the URL / Path, use your Personal Repository URL.
5. Ignore the settings under Optional Extended Integration, and click OK.

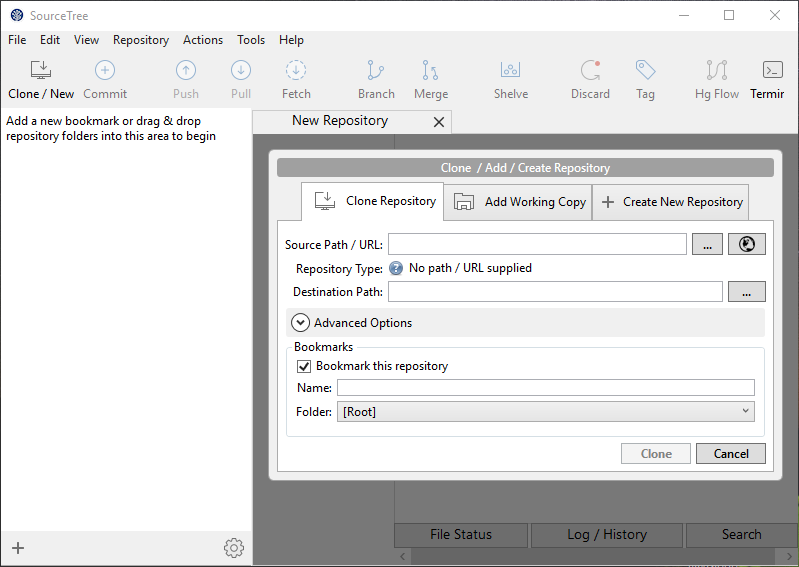
### Gotchas

|  |  |
| --- | --- |
|  | If you have not disabled SSL certificate validation, you will not be able to add the Bonobo server as a remote. Follow the instructions in the [Post-Installation Setup](#_Post-Installation_Setup) section of this document to do so. |

## Cloning Guide

1. Retrieve your Personal Repository URL. If you need help with this, follow the instructions in the [Retrieving Your Personal Repository URL for git](#_Retrieving_Your_Personal) section of this document.
2. Click the Clone/New button in SourceTree. You will be prompted with the following screen:
3. Fill out the Source Path / URL textbox with your Personal Repository URL.
4. Select a destination path for your local repository, and click Clone.

## Optional: Instead, Creating a New Local Repo and Pulling

1. Click the Clone/New button in SourceTree. You will be prompted with the following screen: 
2. Switch to the Create New Repository tab.
3. Select a destination path and name the repository, click Create.
4. Go to Repository>Repository Settings, and under the Remotes Tab, click Add.
5. Name the remote something like Bonobo or Dev1.
6. For the URL / Path, use your Personal Repository URL.
7. Ignore the settings under Optional Extended Integration, and click OK.
8. Now, Click Pull, and under Remote Branch to Pull, select master. Click OK.

# git Actions

## Creating a Branch

### Naming Convention

|  |  |
| --- | --- |
|  | When creating a new branch, always name the branch using the convention “{username}/{branchname}”. This is what enables SourceTree to organize branches per developer. |

### Guide

1. First, switch to the branch you want to branch off of by double clicking the branch in the BRANCHES list.
2. Then click the Branch button and name the branch using the specified convention.
3. If you want your branch to start at the branch head, leave “Working copy parent” selected. Otherwise select “Specified commit” and then click the three dots button to open a commit tree where you can select the commit the new branch should branch off of.
4. If you leave “Checkout New Branch” checked, you will automatically be switched to the new branch you just created.

## Creating a Commit

### Conventions

|  |  |
| --- | --- |
|  | Commit text should be an action statement regarding what that commit does to the code base. Examples include: “Implements feature A” or “Fixes bug B” or “Creates a new object C” |

### Guide

#### Staging Files

Before you are able to commit, you must stage the files you wish to be included with that commit. This is done by highlighting the uncommitted changes bubble in the Log / History, and then selecting the files in the Unstaged files list to stage. Typically, you should be staging all files, but make sure git isn’t trying to have you stage any bin or obj files.

#### Creating a Useful Commit Message

After staging the files to be included with the commit, click the commit button to be prompted with a text field to enter a commit message. Follow the guidelines in the [Conventions](#_Conventions) section of this document. Provide as much detail as is reasonable, including links to the Trello card associated with your changes if such a card exists.

## Synchronizing Branches with Bonobo

### Fetching and Pushing Branches and Commits

|  |  |
| --- | --- |
|  | When preparing to push a commit to the server, especially when that branch is the master branch, always first perform a fetch. This will allow you to identify if there will be any merge conflicts BEFORE you push to the Golden Master. |
|  | If there are unsynced commits, pull them. If there are conflicts that you must go fix, do not forget to test your code, and then repeat the fetch on the likely probability that another developer pushed to master while you were resolving merge conflicts. |
|  | Only when the fetch returns no unsynced changes to your current branch shall you push your commits. Do so immediately to minimize the likelihood of someone pushing between your fetch and your push. If you wait, fetch again. |

## Merging Branches

### …To Local Master and Pushing to Golden Master

#### …When the feature branch has 5 or fewer commits AND is not a ‘small feature’

In this scenario, you should perform a standard merge. The process for doing so is as follows:

1. Make sure your local master is synced with the Golden Master.
   1. Perform a fetch to check if your local master has any unsynced commits.
      1. If so, pull them from Golden Master into your local master.
   2. If the fetch returns empty, your local master is up to date.
2. Merge your local master into your feature branch. Pay close attention to the directionality here.
   1. In SourceTree, you merge one branch into another by doing the following:
      1. Switch to the branch you want to merge INTO.
      2. Right-Click on the branch to merge into your current branch.
      3. Click on “Merge {branchname} into current branch” in the contextual menu that appears. Click “OK”.
   2. If there are merge conflicts, resolve them. Since you merged your local master into your feature branch, you will still be on your feature branch for resolving the conflicts. Commit any resolutions to your feature branch, and fetch master again to make sure your local master is still in sync with the Golden Master.
3. Once your local master has been successfully merged into your feature branch, and all conflicts have been resolved and committed to that feature branch, you may merge your feature branch into your local master.
   1. Switch to your local master branch by double clicking it in the branches list.
   2. Right-Click on your feature branch and select “Merge {branchname} into current branch” in the contextual menu that appears. Click “OK”. If you followed the steps up until now properly, there will not be any conflicts.
4. Now that you have merged your feature branch into your local master, fetch once more to make sure your local master is still not behind Golden Master.
   1. If there are unsynced commits to pull, pull them and resolve any conflicts. Once you are done resolving any conflicts, fetch again to ensure you are still in sync.
5. Now, your local master should be ahead of the Golden Master, and not behind. Push your commits to the Golden Master.

#### …When the feature branch has more than 5 commits

In this scenario, you will perform a ‘Soft Reset’ merge. The steps for doing so are as follows:

1. Follow steps 1, 2, 3, and 4 from the […When the feature branch has 5 or fewer commits AND is not a ‘small feature’](#_…When_the_feature_1) section of this document.
2. Before pushing your local master to the Golden Master, you will perform a soft reset.
   1. Right-Click on the commit in the Log / History that is immediately prior to your feature branch merge. It will be the commit that was the local master head before you merged your feature branch into your local master.
   2. Click on “Reset current branch to this commit” in the contextual menu.
   3. **Make sure the using mode is set to Soft.** 
   4. Click “OK”.
3. Your feature branch in its entirety will be a singular blob of uncommitted changes on top of your local master with no linkages to your feature branch. Commit those changes with a commit message matching the form “Delivering changes from {branchname}”.
4. Perform a fetch to ensure your local master is not behind the Golden Master.
   1. If there are unsynced commits to pull, pull them and resolve any conflicts. Once you are done resolving any conflicts, fetch again to ensure you are still in sync.
5. Now, your local master should be ahead of the Golden Master, and not behind. Push your commits to the Golden Master.

### … To Any Branch Other than the Master Branches

This situation does not matter. Merge however you like.

## Feature Collaboration

|  |  |
| --- | --- |
|  | When you are co-developing a feature with a co-worker, follow the following branching format to make both of your lives easier. 3 branches will be made using the form shown in the image below. |

Your Coworker’s Feature Branch

Your personal Feature Branch

Shared Feature Branch

Master Branch

### Creating A Shared Feature Branch

Create a branch off of master named “Shared/{featurename}”. Treat this branch as if it was a master branch. Do not edit on it directly.

### Creating Your Personal Development Branch

Create a branch off of your newly created Shared Feature Branch using the naming scheme “{username}/Personal{sharedFeatureBranchName}”. This will be the branch for you to use for implementing the feature. Your co-developer will have their own.

### Sharing Progress with your Feature Co-Developer via the Shared Feature Branch

Merge changes to the Shared Feature Branch as if it was a master branch. Your co-developer will retrieve your commits through pulling and merging this branch into their personal feature development branch.

### Merging the Shared Feature Branch Back to Master and Golden Master

Make sure the Shared Feature Branch is in a complete state, and follow the instructions for delivering changes to master as outlined in the […To Local Master and Pushing to Golden Master](#_…To_Local_Master) section of this document

# General git Rules

## Never Re-Write Golden Master History

|  |  |
| --- | --- |
|  | You may commit to your local master and change history (via deleting commits or likewise) on it within the range that it is ahead of the Golden Master all you wish. However, once a commit has been pushed to the Golden Master, it is set in stone. |
|  | If you realize you made a mistake in a commit that was pushed to the Golden Master**, DO NOT** try deleting it and recommitting it in a fixed state. Own the mistake and create a new commit that fixes the issue. |

## Never Work on the Master Branch, Always Make a Branch

|  |  |
| --- | --- |
|  | When making changes locally, never have your working copy be on the Master Branch. This opens you up for merge conflicts if one of your fellow developers pushes anything to the Golden Master, and removes your freedom to commit freely. |
|  | Instead, when making changes locally, **ALWAYS** first make a feature branch for your edits. |

## Merge Style

|  |  |
| --- | --- |
|  | When merging a feature branch to master, **ONLY** do a standard merge if the feature branch contains only 5 or fewer commits, **UNLESS** it is a ‘small feature’ when at which you will still not do a standard merge. |
|  | In all other cases, perform a **‘Soft Reset Merge’** to prevent cluttering up the history of the master branch. For more information about how to perform a Soft Reset Merge, refer to the […When the feature branch has more than 5 commits](#_…When_the_feature) section of this document. |

## Never Delete a Branch from Bonobo from Which You Have Merged Changes to the Golden Master

|  |  |
| --- | --- |
|  | You may delete your local copy of the branch, but the server version shall be considered a matter of permanent record kept for archival purposes. |

# Extra Niceties

## Creating a Gravatar

SourceTree makes use of the Gravatar system to retrieve avatars for the Log / History. To get one for yourself, go to [Gravatar's website](http://en.gravatar.com/) to create an account. You can make the account using your personal, non-work, email address if you would like to make use of your Gravatar in other contexts and then add your work email as an additional email linked to the account.