****

**Beginner’s Guide**

Published 3/10/2014

CdsGit Revision 1.0

Keywords: CdsGit Beginners Guide Basic Usage Tutorial

Abstract: A beginner’s guide to using CdsGit in Cadence Virtuoso.

# Table of Contents

[1 Introduction 3](#_Toc384235713)

[2 Git Resources 4](#_Toc384235714)

[3 Your First Repository 5](#_Toc384235715)

[3.1 Cloning an existing repository 5](#_Toc384235716)

[4 The First Commit 8](#_Toc384235717)

[5 Updating the Repository 13](#_Toc384235718)

[5.1 Textbook way using “Fetch” + “Merge” 13](#_Toc384235719)

[5.2 Fast way Using “Pull” 14](#_Toc384235720)

[6 Resolving a Conflict 15](#_Toc384235721)

# Introduction

**Welcome to CdsGit!**

The following guide will help you get up and running fast with CdsGit. The goal of this guide is to provide a step by step introduction to using the tool. By the end of the guide, you should be able to perform all of the basic tasks involved in managing a CdsGit repository

# Git Resources

There are many online resources to acclimate you to using git. While you do not need them to finish this guide, it is strongly recommend to review the following websites:

* Interactive Git Tutorial **(STRONGLY RECOMMENDED!)**

[**http://try.github.io/levels/1/challenges/1**](http://try.github.io/levels/1/challenges/1)

* Git Documentation

<http://git-scm.com/documentation>

Chapters 1-3 are especially useful

# Your First Repository

## Cloning an existing repository

This example will highlight how to create your own copy of an existing repository, refered to as “cloning”. Cloning is the first step in contributing to a git repository.

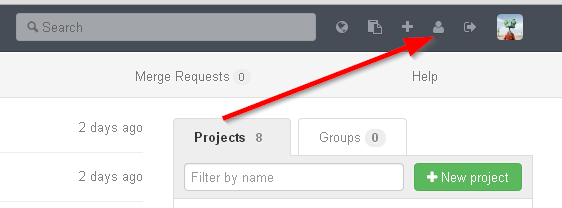
**Step 1) Setup SSH key**

Before you can clone remote repositories, you usually need to create and SSH key on your local machine and add it to your profile on the remote server. Your SSH key has the following purpose:

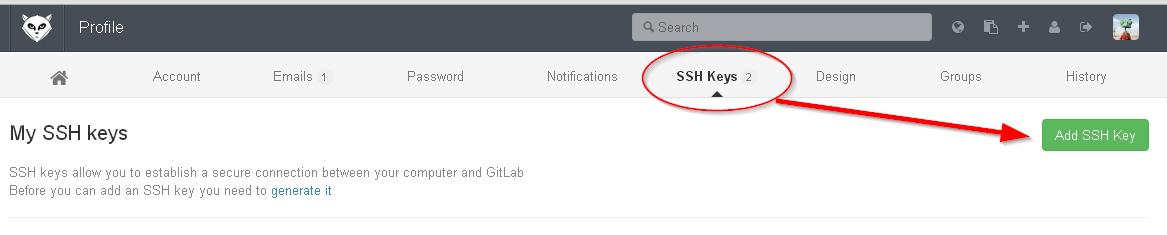
* Identifies to the remote server who you are.
* Allows the remote server to enforce permissions.

In this step, GitLab is use as an example for the remote server.

1. In the GitLab top toolbar, open up your profile be selecting the profile icon:

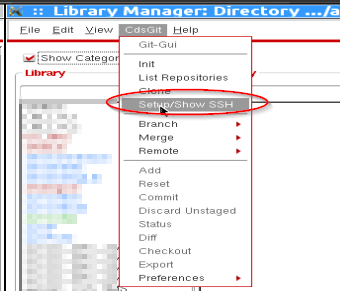


1. Select the SSH Keys tab and Click “Add SSH Key”

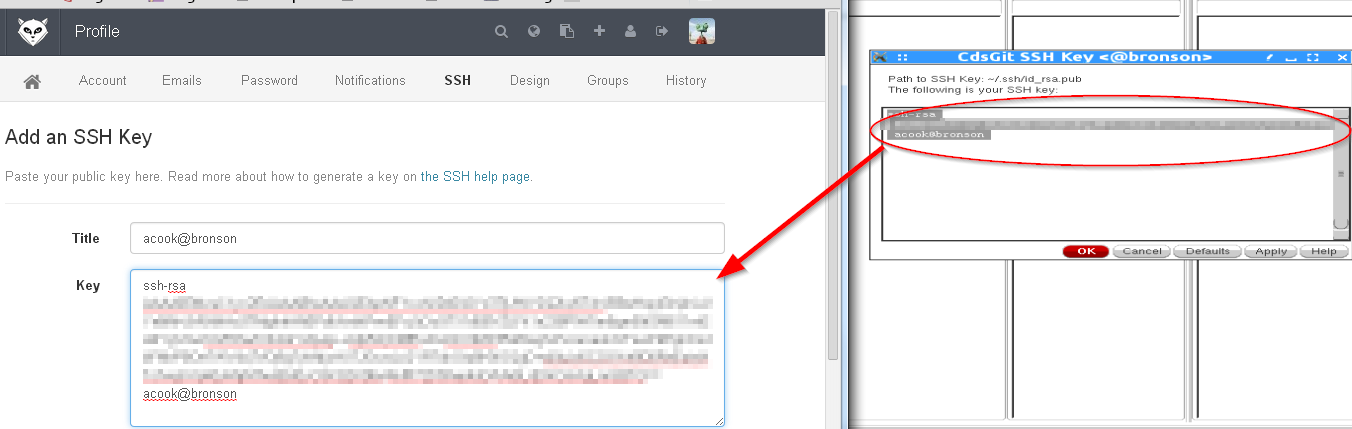


1. Startup Cadence (assuming CdsGit is properly installed).

In the Library Manager select CdsGit->Show./Create SSH Key



1. If creating an SSH key, you will be prompted in the terminal for a passphrase, leave it blank.
2. Copy all text from the key shown into your profile:



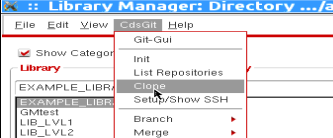
Your SSH key is now setup and you will be able to access the repository!

**For a given remote server, this step only needs to be repeated if you change computers.**

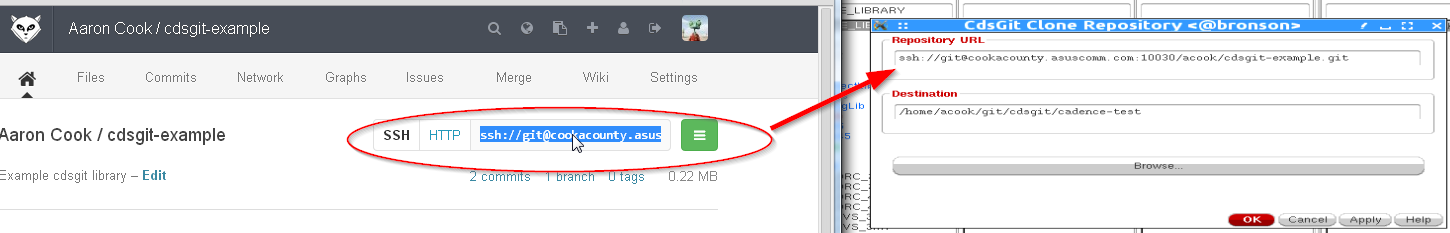
**Your SSH key is commonly stored in your home directory under ~/.ssh/id\_rsa.pub**

**Step 2) Clone the repository**

1. Navigate in GitLab to your project homepage
2. In the Library Manager, select CdsGit->Clone



1. Fill out the form and hit OK
   * The repository URL is the location of the repository you want to clone. The project homepage in GitLab shows this URL. SSH or HTTP can be used for the commination protocol (SSH recommended).
   * The destination is where the local repository will live. This is the path you will be making modification in.



1. CdsGit will display your information, if it correct press “Yes”
2. Wait for the cloning process to complete. If successful you will see:



**Had an error? Most of the time cloning will fail because:**

* + - You didn’t setup an SSH key
    - You do not have permissions to clone the repository
    - The path you are cloning to is read-only

1. Update your cds.lib to point to the new cloned library. The path is the destination you specified in the cloning form.

# The First Commit

The following section will show you how to:

* Make a modification
* Stage the modification to be committed
* Commit the staged modification.
* Push the committed modification to a remote server so that other’s may utilize it.

1. Make a modification

We will assume that you have a CdsGit manage library cloned and available in your Cadence Session. This example will show an example RC filter schematic where we will add a second stage.

Open the cell view, make modifications, then check and save.

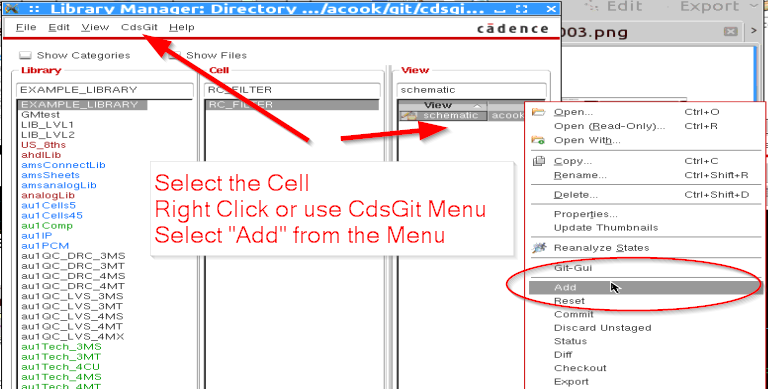
|  |  |
| --- | --- |
|  |  |
| Original | Modified |

1. Stage the modification to be committed

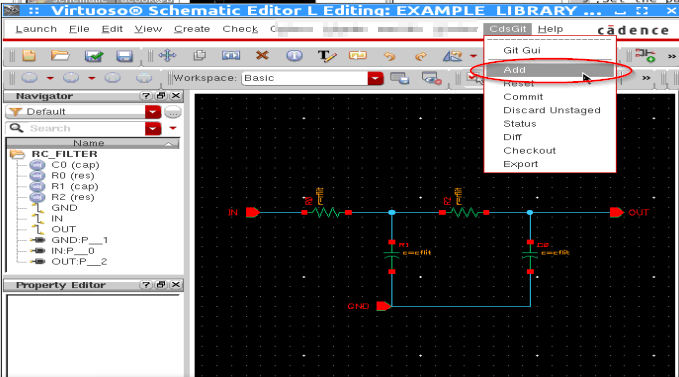
Only modifications that are staged will be committed. By default when you make a modification it is considered “unstaged”. By staging it, you are telling Git “Commit this change next time I make a commit”.

There are 2 ways in CdsGit to stage a modification

* From the library manage – Select the Library, Cell or View, Right Click, and Select “**Add**”



* From the Schematic/Layout editor – Open the CdsGit menu and Select “Add”



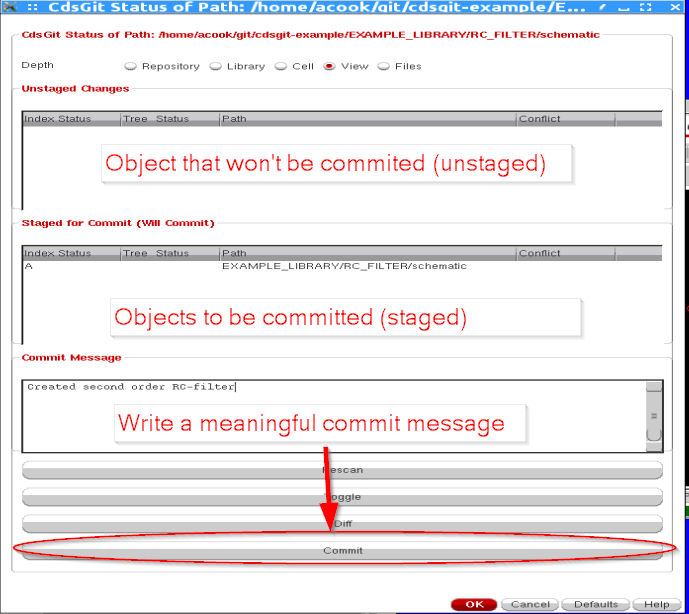
Tip – To Unstage, select “**Reset**”, you modification will now no longer be committed next commit.

1. Commit

Select Commit from the CdsGit menu. Again you can use either the Library manager or the schematic/layout menu.

Selecting Commit will open the Status form

* Review the staged/unstaged objects and verify you want to commit them
  + You can move items back and forth using the Stage/Unstage button or double clicking them
* Enter a meaningful commit message
* Click the “Commit” button.

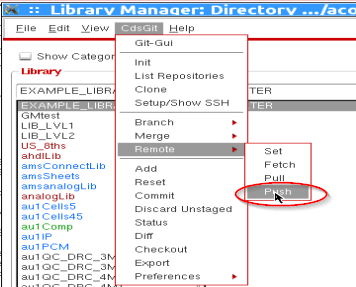


Your modification is not commited in the local repository!

1. Push the committed modification to a remote server

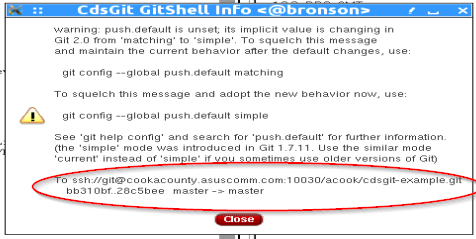
The change you have made is only committed in your local repository. For others to see the modification, you need to “push” your repository to the remote server, synchronizing the two.

From the Library Manager select CdsGit->Remote->Push



Wait for the push to finish. Currently there is no status indicator, look for one in future CdsGit releases.

If the push is successful you will see a message similar to:



Ignore the warning jargon. It is warning you that all branches are being pushed, legacy Git defaults were to push only the current branch.

Look to the bottom where it shows “To …..”

The log shows that the branch master was updated from revision bb310bf to 28c5bee. The push was a success!

**My push failed! Why?!?**

Common reasons for a push to fail

* You do not have permissions on the branch you pushed to.
  + GitLab allows admins to “protect” certain branches so that only certain users can update them.
* You did not setup an SSH key
* Someone else pushed an update since you last
  + You need to update your local repository and merge in the modifications.
  + See “Updating the Repository”
* **Merge Conflict!**
  + See “Resolving a Merge Conflict”

# Updating the Repository

In the last section, you saw how to make a commit and share it with others. The next step is to update your local work area with your team members commits. This step is referred to as “Pulling”.

Pulling is a shortcut for two important steps:

* Fetch – Updates your local repository with incoming changes, does not actually change any of the files in your workarea
* Merge – Update the files in your workarea with the changes received from the “fetch”.

Since Git fundamentals are outside the scope of the document, the details of fetching and merging will not be covered here. It is highly suggested to refer to the Git documentation referenced earlier.

## Textbook way using “Fetch” + “Merge”

The following method illustrates the two steps that the “Pull” command uses. The same method can be used to merge in remote or local branches.

1. To be sure you don’t lose data, commit all the local modifications in your workarea

The command “git stash” is a fast way to temporarily save your changes

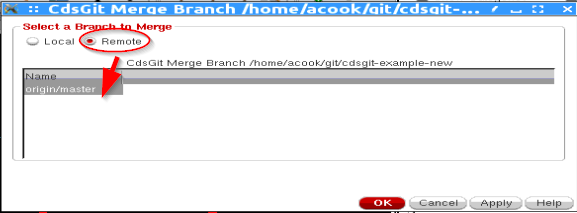
“git stash pop” will restore the last stashed changes

1. From the Library Manager CdsGit menu select CdsGit->Remote->Fetch

Wait while the local work area index is updated

1. From the Library Manager CdsGit menu select CdsGit->Remote->Merge

Select the remote branch you want to merge in



1. If the merge fails you will see:



Refer to “Resolving a Merge Conflict”

## Fast way Using “Pull”

Pull will automatically “fetch” from the remote server and merge in the matching remote branch.

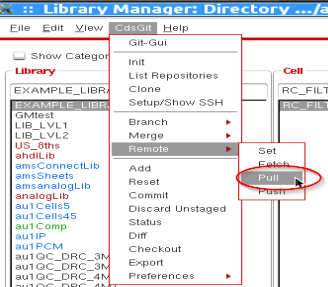
1. To be sure you don’t lose data, commit all the local modifications in your workarea

The command “git stash” is a fast way to temporarily save your changes

“git stash pop” will restore the last stashed changes

Stash support will be added to CdsGit in future releases, for now use the command line.

1. In the Library Manager CdsGit Menu select CdsGit->Remote->Pull



1. If no differences between the local and remote are found you will see:



Otherwise you will see the changes that were pulled (fetched + merged):



# Resolving a Conflict

Let’s propose a simple scenario of events:

* You clone a repository from a remote server
* A team member clones the same repository from the same remote server
* You modify a cell and commits
* Your team member modifies the same cell and commits
* Your team member is faster, and he pushes his change before you
* You attempt to push your change…..**ERROR!!**

You have created a merge conflict. Git does not know which version of the file is the one you really want, thus you must resolve the conflict and tell Git which version you want.

CdsGit simplifies the merge process by allowing you to quickly select whether you want your copy, or the “other” copy.

Currently here is how to resolve a conflict

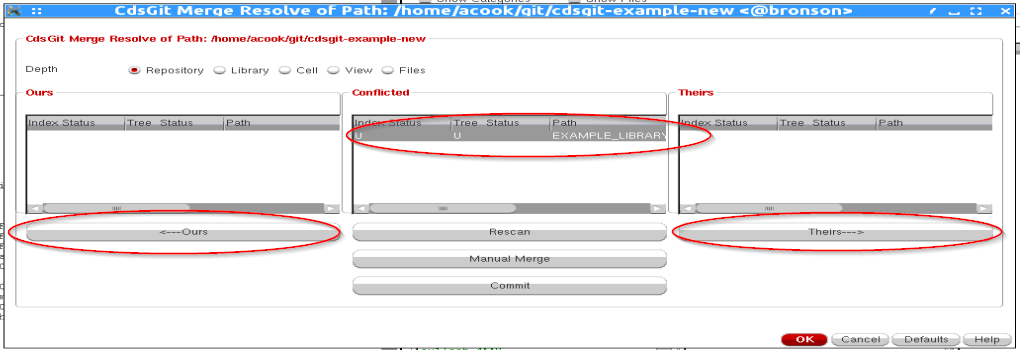
1. You ran the “Pull” command
2. Abort the attempted pull

Future CdsGit will automatically pop up the merge resolve form when a pull fails

1. Follow the “Fetch” + “Merge” procedure from “Updating the Workarea”
2. A form will pop up asking to resolve conflicts, select yes



1. For each conflict use the “Ours” or “Theirs” to resolve



* Currently you cannot undo. Select CdsGit->Merge->Abort and redo the merge if you select wrong
* Manual Merge is not implemented

1. Click the commit button once all conflicts are resolved. You can now push back to the remote