

GitBox – User Manual

Git repository hosting inside a Dropbox folder

# Philosophy

As a developer I’ve come to appreciate code versioning a lot, especially the distributed version control systems. My favorite is git (thus the project, doh…), but it’s a matter of personal preference. I use git for most of my personal/hobby projects, theses and work; but there are some practical problems:

I use multiple machines and multiple operating systems, thus synchronizing my work between them and creating/managing backups is a pain without some third party source repository. For work, a company versioning system will do, and for open source projects github will do. But for personal research projects another solution was needed (private git hosting is too expensive, period).

The idea came (not mine, Google it) from the file synchronization service called Dropbox. It keeps a “special” folder on the file system in sync between different operating systems and machines (i.e. whatever you place in one will be available on all); as well as keeping a backup in their cloud service. It’s private and up to 16GB (through referrals) are free.

Thus the idea, that if we place a git repository inside the Dropbox folder, it will be available virtually anywhere, as well as backed up in case of a HDD failure. Whenever a commit is pushed into the repo, Dropbox would automatically sync it with the others. Instant private git hosting.

Although managing git repositories inside Dropbox aren’t too hard (it’s the same as if you’d manage it anywhere else remotely), you still have to remember a “handful” of commands as well as long paths. GitBox was born out of the necessity to make this process more user-friendly and automatic: creating, cloning and importing repositories should be one-liners.

Hope you like it; I’m open for suggestions, requests, bug reports and any feedback whatsoever. ☺

P.S. Less is more… I prefer stable products over large feature sets.

Péter Szilágyi

Table of Contents

[Philosophy 1](#_Toc327471694)

[1 Introduction 3](#_Toc327471695)

[2 Installation 3](#_Toc327471696)

[2.1 Supported Platforms 3](#_Toc327471697)

[2.2 Installation Steps 3](#_Toc327471698)

[2.2.1 Step 1 – Install Dropbox 3](#_Toc327471699)

[2.2.2 Step 2 – Install GitBox 4](#_Toc327471700)

[2.2.3 Step 3 – Configure GitBox – Linux, Mac OS 4](#_Toc327471701)

[2.2.4 Step 3 – Configure GitBox – Windows 4](#_Toc327471702)

[2.3 Verification 5](#_Toc327471703)

[3 Command Reference 5](#_Toc327471704)

[3.1 gitbox 5](#_Toc327471705)

[3.2 gitbox list [<group>] 5](#_Toc327471706)

[3.3 gitbox create <repo> 6](#_Toc327471707)

[3.4 gitbox clone <repo> 6](#_Toc327471708)

[3.5 gitbox import <repo> 6](#_Toc327471709)

[4 Samples and Tutorials 6](#_Toc327471710)

[4.1 Backup Code into the Cloud 7](#_Toc327471711)

[4.2 Access Repository from Multiple Machines 8](#_Toc327471712)

[4.3 Create Secondary Repository for Existing Project 8](#_Toc327471713)

[4.4 Organize Repositories into Hierarchical Groups 9](#_Toc327471714)

[5 Releases 10](#_Toc327471715)

[6 Acknowledgements 11](#_Toc327471716)

[7 License 11](#_Toc327471717)

# Introduction

GitBox is a cross-platform git repository manager for hosting private personal git repos inside Dropbox folders. Thanks to the nature of Dropbox, features include automatic synchronization between multiple machines and operating systems, as well as online backups.

Currently GitBox is capable of creating git repositories inside Dropbox folders, cloning repositories residing inside these folders, and importing existing git repos into Dropbox folders.

What GitBox is not, is a collaboration tool between multiple developers. Sharing a git repository through Dropbox will have serious consequences as there is no way to do atomic syncs.

# Installation

## Supported Platforms

GitBox differentiates between two levels of support: fully and partially supported platforms. Full support means that everything works out of the box. Partial support usually means that the tool itself works fully, but an installation or configuration step may require manual user intervention and is not fully automatic.

Fully supported platforms:

* Fedora
* Mac OS X 10.6+
* OpenSuSE 11.1+
* Ubuntu
* Windows XP SP2+, Vista, Windows 7

Partially supported platforms:

* Linux flavors (manual git installation is required)
* Windows before XP SP2 (requires manual configuration)

## Installation Steps

The following sections describe each installation step individually. Some of them are generic whilst others are operating system specific. In case of an OS specific step, it will be noted accordingly.

### Step 1 – Install Dropbox

As a prerequisite, GitBox requires you to have a valid Dropbox installation on your machine. If you haven’t already done so, in order to install Dropbox, you’ll need to first register an account for free at the Dropbox website. (Consider registering through [my referral link](http://www.dropbox.com/referrals/NTEwNDM1OTA5OQ) as support for this project, you’ll also get an additional 250MB of storage space). After registration you can [download](https://www.dropbox.com/downloading) and install their desktop client for various platforms.

### Step 2 – Install GitBox

Note, that this step is only needed once. If you already installed GitBox into your Dropbox folder on one machine, Dropbox will synchronize it with all of your other machines, so you can jump to the next step.

Download the GitBox package, which is a simple zip file available from the project’s [github download page](http://github.com/karalabe/gitbox/downloads). You need not worry about operating systems, since there is a single, cross-platform GitBox bundle. After acquiring the distribution zip package, GitBox should be extracted **into** your active Dropbox folder (the root itself is a good choice since GitBox is already in its own folder). Make sure you’re happy with the location before proceeding.

### Step 3 – Configure GitBox – Linux, Mac OS

To finish the installation in Linux, Mac OS and other flavors of \*nix, the setup.sh shell script in the setup folder should be executed.

$ cd <gitbox root>/setup  
$ bash setup.sh  
GitBox was successfully configured.

### Step 3 – Configure GitBox – Windows

In order to finish installation under Windows, the GitBox root folder containing the executable batch file should be added to the user’s path environmental variable.

For users of Windows XP Service Pack 2 and newer, Windows Vista and Windows 7, a setup script was also included in the bundle (<gitbox root>\setup\setup.bat), which will configure the path automatically. Simply run this batch file and you’re ready to go.

At the moment, users of previous versions of Windows need to add GitBox manually to their path variable through the Control Panel → System → Advanced → Environment Variables. The GitBox root folder should be appended to the path. Please note that only newly started programs will use the modified path.

## Verification

In order to verify that GitBox was correctly installed and configured, switch to a random folder on your computer and execute “gitbox” (under \*nix, do this from a shell).

At this point GitBox will check whether there is a valid git installation on the local computer and offer to install one if needed. After which a git bash console should be started up.

Should the gitbox script not be found, please go back and double check your paths and environmental variables.

# Command Reference

## gitbox

The command ensures that git is available (offers to install it otherwise) and prepares a console for a git workflow.

The command is specifically designed for extensive use under Windows, where git and related bash utilities aren’t readily available in the base command prompt provided by the operating system.

## gitbox list [<group>]

Lists all the git repositories tracked by GitBox. The optional <group> lists only a subgroup of the maintained repos.

$ gitbox list  
List of repositories tracked by GitBox:  
 + patches  
 - canvas  
 - nitrogen  
 - myapp  
 - gitbox  
  
$ gitbox list patches  
List of repositories tracked by GitBox:  
 - canvas  
 - nitrogen

## gitbox create <repo>

Creates a new empty git repository called <repo> inside the GitBox repository collection.

This command is to be used when a new repository/project is needed and there aren’t any existing repositories which can be imported into GitBox.

$ gitbox create myapp  
Creating empty repository...  
Initializing new repository...  
Repository successfully created.

## gitbox clone <repo>

Clones a git repository called <repo> from the GitBox collection into the current folder and configures a default remote called *gitbox* for committing code back into GitBox through git.

$ gitbox clone myapp  
Cloning repository...  
Repository successfully cloned.

## gitbox import <repo>

Imports an existing git repository into GitBox and also configures a remote called *gitbox* for the local repo. Note, this command is meant to be executed from within a git repository, which is to be imported.

$ gitbox import myapp  
Creating empty repository...  
Importing data into new repository...  
Repository successfully imported.

# Samples and Tutorials

The tutorials are split into a couple of common use-case scenarios that the tool was meant to support. Although they can be understood separately too, they were built one on top of the other.

Basic:

* Backup Code into the Cloud
* Access Repository from Multiple Machines

Advanced:

* Create Secondary Repository for Existing Project
* Organize Repositories into Hierarchical Groups

## Backup Code into the Cloud

The most important goal of GitBox is to allow the creation, constant maintenance and all time availability of backups, without the need of any paid third party hosting services or personally maintained servers.

This tutorial assumes that we either have a yet unversioned project that we would like to add versioning to and backup in the cloud, or we would like to start a brand new project tracked by GitBox. In either case, let’s call the project *myapp*.

The first thing to do is to create a new empty git repository inside GitBox and initialize it. This will place a readme file into the new repo to make git happy (it doesn’t like empty repos).

$ gitbox create myapp  
Creating empty repository...  
Initializing new repository...  
Repository successfully created.

Find a suitable working folder for the project and change to that specific directory. Here, clone the newly created repository. This will result in a subfolder being created with the name of the cloned repo (so in case you would like to clone to /work/uni/myapp, then issue the clone in /work/uni).

$ gitbox clone myapp  
Cloning repository...  
Repository successfully cloned.

From this point onward it’s a simple git workflow to back up things into the cloud:

$ echo “Some change” > somefile.txt  
$ git add somefile.txt  
$ git commit –m “Created some file”  
$ git push

Whenever you issue a git-push, your committed changes will be pushed into GitBox, and subsequently synchronized and backed up in the cloud. Be sure to wait until the Dropbox synchronization finishes before turning off the computer in order to have everything fully backed up.

## Access Repository from Multiple Machines

After backups, the second most important feature is the ability to work from multiple operating systems and/or machines without the need to manually copy files. This tutorial is a continuation of the previous one, where we already created and pushed our project into GitBox.

If we would like to access our repository from another operating system, computer, or simply after having it deleted from out original machine, we first list the available repositories tracked by GitBox to make sure we get the name right:

$ gitbox list  
List of repositories tracked by GitBox:  
 - gitbox  
 - myapp

Here I had two repositories tracked by GitBox: the myapp project we just created in the previous tutorial, and the repository of this project itself. After listing, we can simply clone the existing repository like we did with the empty one previously:

$ gitbox clone myapp  
Cloning repository...  
Repository successfully cloned.

We can now edit and push changes back to the repository like before. Whenever we push something into the repository, we must make sure that it gets fully synced before pushing changes from somewhere else to prevent repository corruption (this shouldn’t be a problem since a synchronization after a git push usually takes only a couple of seconds in the case of source code).

After we made sure syncing did indeed finish, we can either clone the repository at a new machine like before, or if we have cloned the repository previously but would like to update it with the newly pushed changes, we can pull them from GitBox:

$ git pull

## Create Secondary Repository for Existing Project

Another common use case scenario with git repositories is that we already have an existing repository that we would like to check into GitBox (e.g. We work on an open source project hosted on github, would like to implement some things without pushing to github, but still having the code backed up just in case).

For the sake of the tutorial, I will clone the ruby on rails repository as the existing project:

$ git clone git://github.com/rails/rails.git

In order to import an existing repository into GitBox, we have to be inside the specific repository we would like to import. Note, we specify the name with which to track the repository inside GitBox.

$ cd rails  
$ gitbox import rails  
Creating empty repository...  
Importing data into new repository...  
Repository successfully imported.

If we list the tracked repositories now, out new *rails* repo should appear:

$ gitbox list  
List of repositories tracked by GitBox:  
 - gitbox  
 - myapp  
 - rails

Finally whenever we make any commits, we can decide where to push the changes.

$ git push origin  
$ git push gitbox

Note, that the GitBox repository (remote) will always be called *gitbox*.

## Organize Repositories into Hierarchical Groups

The basic usage model of GitBox (creating and maintaining all the repositories in a single folder) eventually becomes a bit disorganized as the number of repos stored grows. To cope with larger sets, the hierarchical grouping of repositories was introduced: instead of having everything in a single folder, sub-folders within the repository storage can be also used.

Creating hierarchically grouped repositories can be done simply by specifying a path besides the name of the repo:

$ gitbox create applications/android/quickzone  
$ gitbox create applications/android/fractalflame  
$ gitbox create applications/iphone/quickzone

Importing too can be done using paths too for organizational requirements:

$ git clone git://github.com/karalabe/gitbox.git  
$ cd gitbox  
$ gitbox import tools/gitbox

If listed, the above would appear as:

$ gitbox list  
List of repositories tracked by GitBox:  
 + applications  
 + android  
 - fractalflame  
 - quickzone  
 + iphone  
 - quickzone  
+ tools  
 - gitbox

Cloning again works by specifying the path too in case of groupings:

$ gitbox clone tools/gitbox

One side note: currently there is no command for moving a repository after creation/import. The reasoning is the same as with repository deletion: manual execution is more conscientious than with a program. You can easily reorganize your repositories by moving them around inside the GitBox repos folder. The only thing to take care of is that after a move any repository already cloned will have an invalid path. Just re-clone the repo with the new path (do make sure all the changes have been committed beforehand).

# Releases

* Version 0.3.0: **2012-06-15**
  + Hierarchical repository grouping
  + Fix: Spaces in repository names at listing
* Version 0.2.1: **2011-04-01**
  + Display command reference if an unknown command is issued
  + Fix: Support paths and repositories with spaces
  + Fix: GitBox clone status message always reported success
  + Fix: Linux and Mac OS X setup script used ‘\n’ literals instead of new lines
  + Fix: Windows version only listed repos with ‘hidden’ file attributes
* Version 0.2.0: **2010-09-30**
  + Mac OS X port of GitBox
  + Automatic git installation in Ubuntu and Fedora
  + User friendly output messages
* Version 0.1.0: **2010-08-27**
  + Support for repository listing, creation, cloning and importing
  + Automatic git installation in OpenSuSE and Windows
  + Automatic GitBox configuration in \*nix and Windows XP SP2+, Vista and Windows 7

# Acknowledgements

Special thanks to the [msysGit](http://code.google.com/p/msysgit/) and [git-osx-installer](http://code.google.com/p/git-osx-installer/) projects and the people behind them for creating and maintaining the Git for Windows and Git for OS X bundles respectively, both distributed with and used by GitBox.

Furthermore I’d like to thank some people who in one way or another contributed to the existence of this project (alphabetical order):

|  |  |  |
| --- | --- | --- |
| * Attila T. Áfra * Bradley Wright | * Daniel Tull * Lőrinc Pap | * Pan Fan |

# License

The MIT License

Copyright © 2010 Péter Szilágyi <peter@karalabe.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.