Help Topic: Getting Started with Git

Fall 2024

Michael Benjamin, mikerb@mit.edu
Oscar Viquez, oviquezr@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

Getting Started with Git

Git is a version control software that allows users to develop source code in a distributed manner, keeping track of changes, conflicts and versions. You will be highly encouraged to use Git or a similarly capable version control tool during the course. But for now, we focus just on installing Git and knowing just enough to check out software from the web to be used in the course.

How to tell if Git is already installed

Your machine may already have a version of `git` installed. It is typically preinstalled on macOS, and may be preinstalled in some Linux distros as well. To check, try:

```
$ which git
/usr/bin/git   (or perhaps /bin/git on Linux)
```

If you get a response like the one above, you're all set. If `git` is not installed you will get a response like the one below:

```
$ which git
git: command not found
```

Or no response at all:

```
$ which git
$
```

If the `which` command is telling you it cannot find the `git` command, you probably just don't have `git` installed. However it may be that you simply don't have the directory containing the `git` executable in your shell path. To rule this out, try the following (if using macOS, or GNU/Linux):

```
$ cd /usr/bin
$ which ./git
./git
```

To check if git is already installed via another method (i.e. MacPorts), try:

```
$ cd /opt/local/bin
$ which ./git
./git
```

If you see the above, it is simply a matter of adding directory to your shell path. Do this now, or see the help topic on augmenting your shell path:

http://oceanai.mit.edu/ivpman/help/help_cmd_shellpath

Getting Git in GNU/Linux

If you're running GNU/Linux you probably have access to a package manager such as apt and already know how to use it to some degree. To install packages like Git, you need root privileges. Assuming your user account has root privileges, do the following, and enter your normal login password if prompted:

```
$ sudo apt install git
```

See the discussion above on how to confirm whether git has been successfully installed, and adjust your shell path if need be.

Getting Git in macOS

If you're running on a Mac (macOS) you can use MacPorts or Homebrew to install Git. If you haven't installed either of these package managers, see the separate help topic on this. To install packages like Git, you may need root (admin) privileges. Assuming your user account has the necessary privileges, do the following, and enter your normal login password when or if prompted:

```
$ sudo brew install git (if using HomeBrew)
```

Or if you are using MacPorts as a package manager:

```
$ sudo port install git   (if using MacPorts)
```

See the discussion above on how to confirm whether git has been successfully installed, and adjust your shell path if need be.

Using Git to clone a repository

The Git executable is git and takes a number of arguments on the command line for interacting with a software repository. The most essential is the command to clone, i.e., download, a repository from a server.

```
$ git clone <url> [<destination>]
```

Below is the git command for checking out the MOOS-IvP software repository:

```
$ git clone https://github.com/moos-ivp/moos-ivp.git
```

A checkout essentially results in the download of a directory from a server. The name of this directory in the server is the final component of the URL, which may include the .git suffix. The name assigned to the directory in the local system typically defaults to the base name of the repository, but can be defined using the optional destination argument – the last component in the example command above, given after the repository's URL. In the above example, the base name of the repository would be `svn-mirror`, and the last (optional) argument in the above example command line ensures the local copy will be listed under `moos-ivp` upon download.

Things to Try at the Outset

If the above is your first taste of git, and `moos-ivp` is the first repository (repo) you have worked with, here are a few things to try. Keep in mind that the `moos-ivp` repo is largely "read only" unlike the repos you will be working with later for labs and projects. Nevertheless, here are a couple things.

Getting Info on a Repo's Origin

After a repo has been cloned, you may find yourself later wanting to remember where it was cloned from. You can try the following:

```
$ cd ~/moos-ivp
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/moos-ivp/moos-ivp.git
  Push  URL: https://github.com/moos-ivp/moos-ivp.git
  HEAD branch: main
  Remote branches:
    moos-ivp-22.8               tracked
    moos-ivp-22.8.1             tracked
    moos-ivp-24.8               tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (up to date)
```

Here you can see the github URL `"https://github.com/moos-ivp/moos-ivp.git"`. This can be used anywhere to clone the repo from github as shown earlier:

```
$ git clone https://github.com/moos-ivp/moos-ivp.git
```

Looking at Recent Commit Log

Git has a history of commits and revisions, and it's all stored in the `.git` folder in the top level of the repo. The log can be viewed with the below command. In this case only the most two recent entries are shown, due to the `-n 2` arguments.

```
$ cd ~/moos-ivp
$ git log -n 2 --pretty=short
commit ed377ae7d8ab8f2ec9ce5c77aded8df91a0f73fc (HEAD -> main, origin/main, origin/HEAD)
Author: Mike Benjamin <mikerb@csail.mit.edu>

   moved from sprintf to snprintf for utils in MBUtils

commit b00b505ec7b533cce19b9bddf9eef48d99c2c763
Author: Mike Benjamin <mikerb@csail.mit.edu>

   minor mods to LegRun OpReg and Waypoint bhvs
```

Notice the full hash of the most recent commit:

`ed377ae7d8ab8f2ec9ce5c77aded8df91a0f73fc`

Looking at Recent Commit Hash

Each git commit has a unique hash value. A full has is 40 characters. The short vesion of the hash is first group of characters of the full hash, typically nine characters. The hash of the most recent commit can be obtained with:

```
$ git rev-parse HEAD
ed377ae7d8ab8f2ec9ce5c77aded8df91a0f73fc
```

Or the just the short part of the hash:

```
$ git rev-parse --short HEAD
ed377ae7d
```

Configuring your Git environment

When committing changes to a repository, Git creates an attribution record so the author can be identified. You can tell Git what name and email address you want associated with your commits by setting the global defaults:

```
$ git config --global user.email "your_email@example.com"
$ git config --global user.name  "Your Name"
```

Alternatively, you can remove the `--global` argument to set repo-specific defaults; the default behavior is equivalent to setting the `--local` argument explicitly. This is useful when you mange

both personal and work repositories, and you want to ensure the system employs the correct email address for each case.

To check the local and global configurations, you can use the following commands:

```
$ git config --local --list     #(invoked within a checked out tree)
$ git config --global --list    #(can be invoked anywhere; reads ~/.gitconfig)
```

More can be learned about Git online. There are numerous help pages, cheat sheets, and the official Git book is even free online. On this course website we do have a few help pages of our own you can check out:

- http://oceanai.mit.edu/ivpman/help/git_log_cmd
- http://oceanai.mit.edu/ivpman/help/git_update_cmd
- http://oceanai.mit.edu/ivpman/help/git_status_cmd
- http://oceanai.mit.edu/ivpman/help/git_add_remove_cmds
- http://oceanai.mit.edu/ivpman/help/git_mv_cmd
- http://oceanai.mit.edu/ivpman/help/git_commit_cmd
- http://oceanai.mit.edu/ivpman/help/git_setup_your_own_repo