Git Installation Procedure

1. Git Installation Procedure

1.1. Prerequisites

- Workstation/laptop with Windows 10 or Mac
- Userid with admin access on workstation

1. This Git installation is recommended by the Git SCM[1] site for Windows Installations[2]. The Git SCM[3] site also supports other commonly used Operating Systems.

2. It is recommended to use openSSH.

3. The mac environments have a native ssh client.

4. For none windows systems, follow the Git installation instructions for the appropriate operating system. The remainder of this document can be used for any unix or linux environment.

5. For release notes, reference file:///C:/Program%20Files/Git/ReleaseNotes.html. You'll need to cut and paste this link into your browser address window after you complete the git installation. It is a link to your workstation.

1.2. Installing Git

1. go to link git-scm[4]. Depending on your workstation, you will see a section on this page with **Latest source Release** for your workstation.

   - For mac install:

---

[1] http://git-scm.com
[2] http://git-scm.com/book/en/Getting-Started-Installing-Git#Installing-on-Windows
[3] http://git-scm.com
[4] http://git-scm.com/

- ◦ You may need to hold down the control key when clicking on the pkg script.
- ◦ Run git command line in the native xterm under "Other" apps.
- • For Windows install:
  - ◦ All defaults with following considerations:
    - ▪ The Vim editor is the default editor. Nano, Notepad++, and VisualStudioCode are other options.
    - ▪ "Use "Git bash window Command Prompt" is recommended
    - ▪ "Use the OpenSSL library" is recommended
    - ▪ "Checkout Windows-style, commit Unix-style line endings" is recommended
    - ▪ "Use MinTTY (the default terminal of MSYS2)" is recommended
    - ▪ Following is recommended:
      - • "Enable file system caching"
      - • "Enable Git Credential Manager"
    - ▪ Following is **NOT** recommended:
      - • "Enable symbolic links"

2. Upgrade Git

Follow the same steps for installating Git. It is designed to install over an existing installation. Note the expected version release change after you are done.

2.1. Initial git Configurations

After you have installed Git successfully, you have full Git functionality on your workstation.

Be sure to use the Git Bash session you just installed for the following sections.

Try the following commands in your Git Bash window:

Run '**git help -a**' for a list of common Git commands.

In order to interface with the GitHub[5] system, the following needs to be performed:

- Identify or Set $HOME variable on your Workstation.
- Build and Configure SSH key Usage
- Identify yourself in Git:

  - Name
  - email
  - Setup your default editor
  - Setup the diff tool you will use
- Add SSH Aliases
- Verify Git remote "origin"

> The above bullets are covered in First-Time Git setup[6]. You may also find the GitHub Help page Set Up Git[7] useful.

The following sections cover required and recommended configurations.

2.2. Identify Yourself in Git

Enter your first and last name:

- $ git config --global user.name "Andy Wallace"

Enter your email address:

- $ git config --global user.email Andrew_Wallace@some-email

Setup your editor:

- $ git config --global core.editor vim

Setup you diff tool:

- $ git config --global merge.tool vimdiff

---

[5] https://github.com/
[6] http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup
[7] https://help.github.com/articles/set-up-git/

You can run the following command to see all of your Git settings:

- $ git config --list

> **Verify the following values have been configured:**
>
> 1. user.name
> 2. user.email
> 3. core.editor
> 4. merge.tool

For windows, verify your git workarea(s) on your windows workstation are on the C Drive in your home directory.

```
$ (current directory)
$ cd ~/
$ Andy@Office-PC MING@64 ~
$ pwd
/c/Users/Andy
$
```

2.3. Identify the $HOME variable on Your Workstation

Mac instructions:

1. $HOME should be defined already

Windows 10 specific instructions:

1. Right Click Windows pane (lower right) and select "System"
2. Search for "View Advanced System Settings"
3. Click on the **Environment Variables…** button on the System Properties window
4. Verify there is a **HOME** variable in the **System variables** list. This variable is referenced by SSH when supporting remote functionality.

   a. Should be set to C:\Users\[userid].
   b. Create (New button) or adjust (edit button) the **HOME** system variable accordingly.

2.4. Build and Configure SSH key Usage

You can reference Generating Your SSH Public Key[8] for more details on the following instructions.

Execute the following in your new Git Bash window.

• $ ls -la ~/

If your do not have a ~/.ssh directory, create it:

• $ mkdir c:\Users\[userid]\.ssh

> **Use the Git Bash window. Windows Explorer will not allow you to create the .ssh directory.**)

$ cd ~/.ssh (c:\Users\[userid]\.ssh)

• Set the ~/.ssh permissions to 740 **($ chmod -R 740 ~/.ssh)**
• Set the file permissions in ~/.ssh to 740 **(Just verify, should be done from previous step.)**

> Note the "~/" utilizes the workstation system **HOME** variable for your userid.

Generate ssh key pair using your GitHub userid:

• $ ssh-keygen -t rsa -f [Your userid] (userid is all lower-case, no mixed case.)

> It is recommended you enter nothing for the pass phrase.

The following two files will be generated:

• **[Your GitHub userid]** - Your **private** key file

---

[8] http://git-scm.com/book/en/v2/Git-on-the-Server-Generating-Your-SSH-Public-Key

- **[Your GitHub userid].pub** - Your **public** key file

⚠️ | **Never send your private key in an email or attach it to any ticket.**

Your interaction with the GitHub managed repositories will be more secure using your ssh key.

Setup your ssh Key on GitHub

1. Copy your public key into your paste buffer
2. Logon to GitHub [9]
3. On upper-right of window select pulldown for **Your Profile**
4. Select **Edit profile** button
5. Select **SSH keys and GPG keys**
6. Click on the **New SSH key** button and follow instructions.

Verify your git workarea(s) on your windows workstation are on the C Drive.

```
$ ajwal@HomeOffice ~
$ cd ~/ (or cd $HOME)
$ ajwal@HomeOffice ~
$ pwd
/c/Users/ajwal
$
```

2.5. Add SSH Aliases

To reduce typing and minimize ssh key issues, the following is done to provide ssh aliases for the Git System bare repository server(s). Add a config file under the \~/.ssh on your workstation for your GitHub userid as follows.

Edit (or create) ~/.ssh/config and add the following lines adjusted for your userid:

$ vim ~/.ssh/config

```
#############################################################
```

[9] https://github.com/

```
################################################################
### GitHub SSH Client Config file                        ###
###                                                       ###
### This code block for GitHub Access.                    ###
###                                                       ###
### Place this code block in file ~/.ssh/config on your   ###
### workstation. If ~/.ssh/config already exists, add     ###
### this code block to file ~/.ssh/config.                ###
###                                                       ###
### DISCLAIMER:                                           ###
###    This code block not designed to work with wildcard ###
###    definition for Host (Host *) in the ~/.ssh/config  ###
###    file.                                              ###
###                                                       ###
### Syntax format                                         ###
###                                                       ###
### Host [ssh alias names]                                ###
###        User [host user name]                          ###
###        Hostname [host dns]                            ###
###        Port 22                                        ###
###        IdentityFile ~/.ssh/[Your userid]              ###
################################################################
#                                                         ###
  Host github GitHub
        User git
        Hostname github.com
        Port 22
        IdentityFile ~/.ssh/"Your GitHub Userid"
#                                                         ###
################################################################
```

This file allows you to enter commands like this:

$ git clone github:cmguy/CM-Plan-Site

Rather than this:

$ git clone ssh://git@github.com/cmguy/CM-Plan-Site

You should now have three files similar to the following in your ~/.ssh directory:

```
ajwal@HomeOfice ~/.ssh
$ ls -la
total 20
drwxr-xr-x   15 ajwal 13899066     4096 Dec  2 10:14 .
drwxr-xr-x    1 ajwal 13899066    12288 Feb 17 12:12 ..
```

```
-rw-r--r--    1 ajwal 13899066      1679 Dec   2 10:12 cmguy
-rw-r--r--    1 ajwal 13899066       408 Dec   2 10:12 cmguy.pub
-rw-r--r--    1 ajwal 13899066      1749 Jul 17  2014 config


ajwal@HomeOfice ~/.ssh
$
```

> Be sure to read all comments whenever you enter git commands. They usually contain some indication of what you need to enter next.
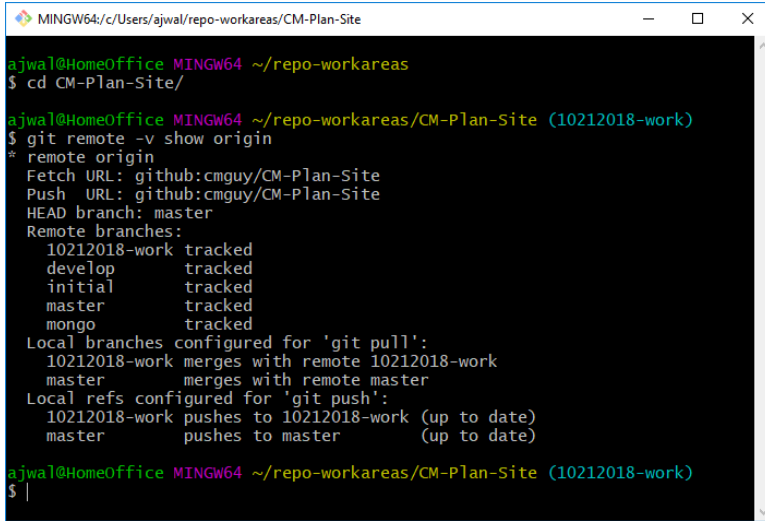
3. GitHub

Once you have established an account with GitHub, mimic the following on your workstation in your new Git Bash Session:

Now you have established a git repo local on your workstation from the Github system. You can verify the git remote origin in the local copy on your workstation.

The git remote `origin` should be setup for communication between your workstation repo and the GitHub system. Reference git remote [10] for more details.

---

[10] http://gitref.org/remotes/#remote

If you are unable to mimic the above Git bash sessions on your workstation, review the **Add SSH Aliases** section of this document.

4. Git Introduction

If you are new to Git, refer to the following links:

* https://git-scm.com/doc
* https://www.youtube.com/user/github/videos

5. GitHub

There are two protected main branches in the cmguy/CM-Plan-Site repo, *develop*, and *master*. Reference a successful-git-branching-model[11] for details.

To update the *develop* or *master* branch, GitHub pull requests should be done. This initiates a colaborative code review session with the repo owner. To do this:

1. git push the branch you wish to merge

2. Do a GitHub pull request. Reference **"Show me how"** at Push to Github & Create a Pull Request[12] for a specific example.

3. Click on pull request and add reviewers

---

[11] http://nvie.com/posts/a-successful-git-branching-model/

[12] https://services.github.com/on-demand/github-desktop/push-pull-request-github-desktop

4. Review and address comments from reviewers

5. Merge pull request

6. Git Conflict Reporting

Reporting on git merge conflicts before actually doing a merge provides a view into additional deltas that need to be considered before doing a merge.

A script tool, **report-conflicts.bsh** has been prepared for DevOps personnel to identify all conflicts to all main, and outstanding release branches for a given feature branch.

> Reference the **Branching and Merging Workflow** section of the Git ./user-guide.pdf[User Guide].

The **report-conflicts.bsh** is designed to be run from any DevOps contributor's workstation or laptop.

Installation Instructions

1. Create a "bin" directory under your ~/ directory

2. Add C:\Users\userid\bin to your workstation or laptop User Variable Path.

3. Get local to your new bin directory and install the report-conflicts.bsh script to it.

   - cd ~/bin

   - cp ~/repo-workareas/CM-Plan-Site/app/bin/report-conflicts.bsh .

4. Setup directories to be used only by the report-conflicts.bsh.

   - mkdir ~/repo-workareas/conflict-reports

   - mkdir ~/repo-workareas/conflict-reports/log

5. Open the script report-conflicts.bsh with vim and set the RepoHome variable to "/c/Users/userid/repo-workareas/conflict-reports".

6. Execute the following for execution instructions:

   - report-conflicts.bsh -h

1. Reference the **"Identify the $HOME variable on Your Workstation"** section above to get you to the "Environments Variables" window to update your User Variable **Path**.