Revcord Git Styles and Practices

Git comments:

1. All comments should use present tense and not past tense. Think what does the commit do. Use words like “create”. Do not use “created”.
2. Use sentences in all comments. End with a period.

Git configuration:

1. Configure git to use your full name, first name then last name. One space between names.

Things to think about:

1. Revcord’s tag name standards.
2. When do we assign a new tag.
3. Revcord needs a document that contains a description of every tag. What the tags ares and why it was created.

Most Common Git Commands.

Check status of staged files.

git status

Add all new or modified files to the staging area.

git add –-all

git add –-all –-dry-run <<< Lists what will be done but does not do it.

Commit all staged files.

git commit –m <”text deascription”>

git commit –m <”Initial file upload.”> --dry-run

Push committed file to a remote repo.

git push

Assign remote repository.

git remote add <name> <url>

git remote add integrations https://github.com/Revcord/integrations.git

Delete a file from a repo.

git rm –cahse <file name>

git rm –dry-run –cache <file name>

To save local changes on GitHub.

git status

git add –-all

git commit –m “text”

optional: git remote add integrations <https://github.com/Revcord/integrations.git>

git push

To clone a new repository

git clone https://github.com/Revcord/projects.git c:\users\rplaza\files\git\projects

Access a Git Repository For the First Time

Revcord repositories.

applications

<https://github.com/Revcord/applications.git>

documents

<https://github.com/Revcord/documents.git>

integrations

<https://github.com/Revcord/integrations.git>

projects

<https://github.com/Revcord/projects.git>

ng911

<https://github.com/Revcord/ng911.git>

To access a remote repository for the first time, a few steps must be taken.

1. Install the Git system.
2. Select and create a local base directory where all of your Git repositories will be located. This directory will be the base directory where eventually sub-directories will be created for each Revcord repository. Eventually directories and files of each repository will be below the repository directory.

As an example, I created the directory: c:\users\rplaza\files\git

1. Open a command window to type Git commands.
2. In the command window, perform initial setup Git commands to be sure your user name and email are set correctly. These commands are remembered and only need to be performed once.

git config –-global user.name <“user name”>

git config –-global user.email <email address>

Examples of these commands are shown below:

git config –-global user.name “Rob Plaza”

git config –-global user.email rplaza@revcord.com

1. In the command window, type the following command to retrieve the files from the repository.

Note: If the directory specification contains a space character, enclose the specification in double quotes.

Note: Since this is the first time you are retrieving files, the process will take a while.

git clone <repository url> <directory name>

An example of this command is shown below:

git clone https://github.com/Revcord/projects.git c:\users\rplaza\files\git\projects

1. During some git commands, git may ask for your git username and password. There is a way to tell git to remember these entries so you do not need to enter them every time. To do this you need to install the credential helper called “wincred”. If you have already installed “GitHub for Windows” the credential helper is installed, you just need to activate it.

Type the following command:

git config –-global credential.helper wincred

Important Note:

**It is very important that we do NOT store huge files in GitHub.**Large files will slow the library for all users.

We should NOT store:

1. zip files.
2. Large .msi install files.

Git Command List

Config --- Setting up git.

To list all global configurations:

git config –-global –-list

To show/set user name:

git config –-global user.name

git config –-global user.name “<username>”

To show/set email address:

git config -–global user.email

git config –-global user.email <email address>

To set pretty colors:

git config –-global color.ui true

To configure lines correctly for Windows:

git config –-global code.autocrlf true

Status --- Git Status.

git status

Stage --- Stage files.

Stages a specified file:

git add <filename>

Stage all new or modified file:

git add –-all

Stages all .txt files in the curent directory:

git add \*.txt

Stages all .txt file from a specified directory:

Git add <directory>/\*.txt

git add docs/\*.txt

Stages all files in a specified directory:

git add <directory>/

git add docs/

Stages all .txt files in the entire project:

git add “\*.txt”

Commit --- Commit files to the local repo.

Commit all staged files:

git commit –m <”text deascription”>

Stage and commit in one command:

git commit –a –m <”text description”>

Reset --- Rollback commits.

Undo last commit and put to staging:

git reset HEAD <filename>

Blow away any changes since the last commit.

git checkout -- <filename> <optional additional filename>

Undo last commit and move everything from commit to staging.

IOW: Set the current timeline to the parent of the current timeline. Stage the deleted stuff.

git reset –soft HEAD^ <filename>

Undo last commit and delete all changes. Blow it away.

IOW: Set the current timeline to the parent of the current timeline. Blow away stuff.

git reset –-hard HEAD^ <filename>

Undo last two commits.

IOW: Set the current timeline to the grandparent of the current timeline. Blow away stuff.

git reset –-hard HEAD^^ <filename>

Log --- View a timeline log of events.

git log

git log --until=1.minute.ago

git log --since=1.day.ago

git log –-since=1.hour.ago

git log –-since=1.month.ago –-until=2.weeks.ago

got log –-since=2000-01-01 –-until=2012-12-21

Diff --- Show what has changed in staged files.

Shows the differences of the current unstagged files.

git diff

Shows the differences of the current stagged file.

git diff –-staged

Shows the difference between unstaged and latest committed:

git diff HEAD

Shows the difference between unstagged and parent of latest commit:

git diff HEAD^

Shows the difference between unstagged and parent of latest commit:

git diff HEAD^^

Show differences between two branches:

git diff <branch name> <branch name>

Blame --- Who made changes.

git blame index.html –-date short

Whatever has been staged is added to last commit. Use –m command to change the comment message.

git commit –-amend

git commit –-amend –m <”comment text”>

To configure a GitHub remote repository

git remote add origin <https://github.com/Revcord/git-real.git>

<<< Use the URL that GitHub gave us.

NOTE: Suggest using “revcord” below to distinguish Revcord’s repository.

git remote add revcord <https://github.com/Revcord/git-real.git>

git remote –v <<< Gets list of all repositories that git knows about

To push commits from local reposiitory to the remote repository.

Pushes all “master” (or branch) files to the “origin”

git push

Full command.

git push –u origin master

git push –u <depository name> <branch name or “master”>

The –u option specifies the info. If the info has not changed since the last push, you do not need to use the –u option.

Follow this link to learn about password caching so only have to enter a password once.

<https://help.github.com/articles/set-up-git>

To pull sync code from the remote repository to the local repository.

git pull <<< No options needed. Syncs your local repository with the remote repository.

Copy the remote branch to your local repository

git fetch

To remove a remote repository form those known by the local git..

git remote rm <remote name>

To get a local copy of the remote repository

git clone <https://github.com/codeschool/git-real.git>

git clone <remote depository URL>

View the remote repositories that the system knows about

git remote –v <<< The –v means verbose.

git remote show <remote repository name> <<< verbose remote information

Check which branch we are set to use.

git branch

git branch –r <<< show all remote branches

Create a new branch

git branch <branch name>

git branch cat

git checkout –b <branch name>

git checkout –b admin

<<< Creates and checksout a branch (in one command).

Make a specified branch the current branch.

git checkout <branch name>

git checkout cat

git checkout master

Merge a branch.

git merge (branch name>

git merge cat

Delete a branch.

git branch –d <branch name>

git branch –d cat

git branch –D <branch name>

<<< Forces the delete event if there are committed file errors>

Push a local branch to the remote repository.

git push <remote repository> <local branch name>

git push revcord shopping\_cart

Gets rid of a remote branch in the remote repository.

git push <remote repository name> :<branch name>

git push origin :shopping)cart

Show remote branches.

git remote –v show <remote name>

git remote –v show origin

Remove stale references.

git remote prune <remote repository name>

git remote prune origin

List all tags.

git tag

To add a new tag.

git tag –a <tag name> -m <”comment”>

git tag –a v0.0.3 –m “Comment for this tag”

To push the tags to the remote.

git push –tags

Checkout using a tagged version.

git checkout <tag name>

git checkout v0.0.1

Rebase

git fetch

then

git rebase <branch name>

git rebase –-continue <<< Continue after resolving proble,.

git rebase –-skip <<< Skip patch.

git rebase –-abort <<< Stop the rebase run.

Remove a file, stage the removal. Then do a commit to perform the actual removal.

git rm <file name>

git rm readme.txt

Stop watching a file

git rm –-caches <file name>

Fork --- Fork remote repository.

Use this command if you have “read” permission but do not have “write” permission to use a repository. The fork command makes a copy of the repository and assigns it to you. You will have full permission to the new repository.

Use the GitHub web page to do the fork operation.

After you fork the repository, go to a command window and clone the repository to make a download a copy of the repository.

git clone <url of the repository>

git close https://github.com/PeterBell/dojo\_rules.git