UCLA Git Walkthrough (for Moodle)

Very basic guide how to get GIT set up on Windows or OSX.

Documentation:

- http://docs.moodle.org/en/Development:Quick_Git_start_guide_for_Moodle_development

Setting up the Environment

- Windows: http://help.github.com/win-set-up-git/
- Mac OSX: http://help.github.com/mac-set-up-git/
- Linux: http://help.github.com/linux-set-up-git/

Configure Line Endings

- In order to avoid issues with line endings when cloning on to Windows machines, follow the directions here: Dealing with line endings

Setting up Git global configs

Before making commits, it is useful to add your name and email:

- `git config --global user.name "Your Name"`
- `git config --global user.email "Your email address"`

Set these git config settings

- `git config --global push.default current // only push current branch to remote and set upstream`
- `git config --global core.ignorecase false // makes sure that git is case sensitive`
- `git config --global pull.rebase true // rebase by default when doing a pull`

If you want to connect to github without SSH you need a token:

- Follow the directions here: Email and Github Tokens

Setting up your Github repository:

Clone the repository

- With Github Token
  - git clone https://YOUR_GITHUB_USERNAME@github.com/ucla/moodle.git ./YOUR_LOCAL_MOODLE_FOLDER/
- With SSH RSA-key
  - git clone git@github.com:ucla/moodle.git ./YOUR_LOCAL_MOODLE_FOLDER/

Work on a new feature/patch/test/update

Our workflow is similar to the workflow mentioned in this article: http://nvie.com/posts/a-successful-git-branching-model/

Name your git branch using our naming convention:

- type/jira ticket-short description
- The type, for now, should either be:
  - feature (something that has never existed before or an improvement to a current feature)
  - patch (a bug fix, can either come from internal or external sources)
  - tests (For Behat or PHPunit only branches)
  - update (reserved for core Moodle version and external plugin updates)

1. `git checkout master`

2. `git pull origin master`
3. `git submodule update --init --recursive`
4. `git checkout -b <type>/CCLE-<JIRA ticket number>-<shorten ticket description>`
5. Repeat the following steps as necessary:
    - - change file(s) -
    - `git commit -a -m "CCLE-#### - A useful short comment summarizing what you did."`

6. `git push -u origin <branch_name>`
7. Merge task onto TEST
    1. `git checkout development`
    2. `git pull origin development`
    3. `git merge --no-ff  <branch_name>`
    4. `git push origin development`

If you run into merge conflicts when merging to development or it's been a while since your development branch has been branched off of master, run the following command on your working branch:

`git rebase origin/master`

At this point, there could be more than one feature that is being tested! Once a feature/patch has passed testing, then merge it to the rc branch.

Creating rc branch

1. `git checkout master`
2. `git checkout -b origin/<release_number>-rc`
3. Now merge in several branches with fixes/features that passed review
    1. `git merge --no-ff <feature_branch_name>`
    2. `git push -u origin <release_number>-rc`

Once rc is ready, merge it into master so it can go to prod

The numbers M.m.v.rr should be the same as the numbers for the RC branch.

1. `git checkout master`
2. `git merge --no-ff origin/<release_number>-rc -m "Release <release_number>-gm: Description of what was in this release/use JIRA version description"`
3. `git push origin master`
4. `git tag M.m.v.rr-gm -m "Release <release_number>-gm: Description of what was in this release/use JIRA version description"`
5. `git push origin M.m.v.rr-gm`

Start On PROD machine ONLY
The numbers M.m.v.rr should be the same as the numbers for the RC branch.

1. `git fetch`
2. `git checkout M.m.v.rr-gm`

3. `git submodule update --init --recursive`
4. Validate things are working...
5. Finished with release cycle!

End On PROD machine ONLY

Performing a hotpatch

1. `git checkout master`
2. `git pull`
3. `git merge --no-ff patch/<branch_name>`
4. `git push origin`

Start On PROD machine ONLY
Make sure you are root user and in the Moodle directory

1. `git fetch`
2. `git checkout master && git pull`
3. `git submodule update --init --recursive`
4. Validate things are working...
5. Finished with hotpatch

End On PROD machine ONLY

Upgrading to new version from Moodle.org

1. `git checkout -b update/M.m.v`
2. `git remote add core https://github.com/moodle/moodle.git`
3. `git fetch core`
4. `git merge -Xtheirs -m "CCLE-<ticket> - Upgrade to Moodle M.m.v" <version tag>`
    1. Might first want to try without the -Xtheirs if upgrading between minor versions.
5. Resolve conflicts
    1. `git add <conflicted_file>`
    2. If needed, continue the merge
6. Finish the pull.
    - Push to development, test on TEST, push to STAGE then release on PROD.
7. Be sure to notate the new RC and GM tags with the updated Moodle version M.m.v.00

8. To upgrade to a major release of Moodle, follow the instructions in this guide:

   http://tjhunt.blogspot.com/2014/01/moving-ou-moodle-code-to-moodle-261.html

---

Revision #134
Created Tue, Feb 15, 2011 5:08 PM by Thompson, Nicholas David
Updated Wed, Apr 3, 2019 11:07 PM by Lorenzo, Rex