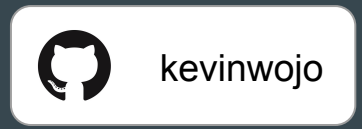
Git with the flow ... man

~~The HUBzero Git Flow~~

...

Kevin Wojkovich <kevinw@purdue.edu>



Audience

- HUBzero Core Developers
- HUBzero Hosted Hub Developers
- External developers

Objective

- You will gain an understanding of how HUBzero's git repository is structured.
- You will understand the HUBzero Monthly Release Cycle.
- You will understand how to contribute changes to:
  - A HUBzero-supported instance
  - The HUBzero Core (open source)
- You will refresh yourself on git fundamentals.

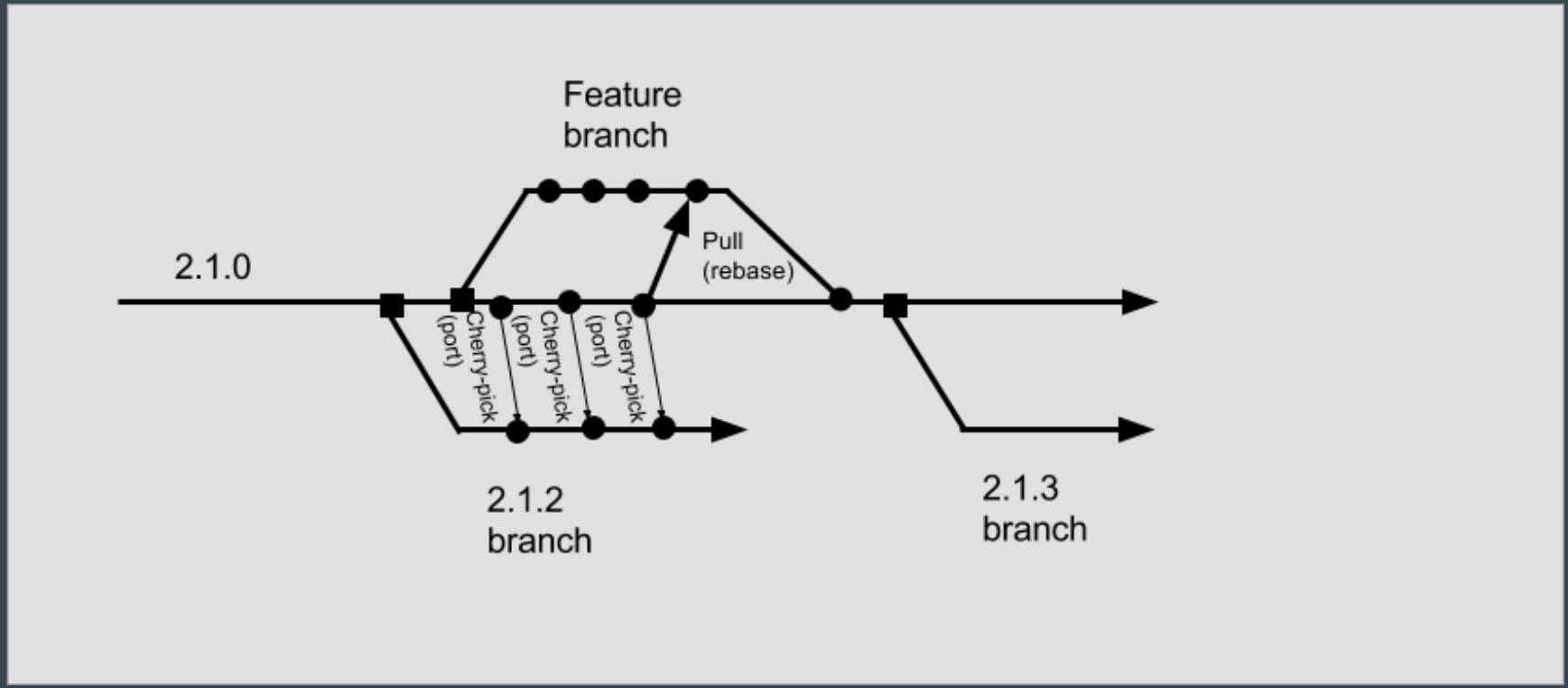
Understanding HUBzero's Branches

- 2.1.0 -> 2.1 branch
  - We made a decision after the branch was created to use it as our development branch.

- 2.1.x -> stable branches
  - These are production branches. They will receive critical fixes after the release until the new stable branch is created. All changes are reflected in 2.1.0

- master -> 2.2x
  - This branch has major changes such as:
    - Removing Joomla dependencies
    - Moving the HUBzero Framework back into the CMS repository

Understanding HUBzero's Branches



Monthly Release Cycle

Week #	Sun	Mon	Tue	Wed	Thurs	Fri	Sat
1		bug-fix	bug-fix	Target Release	Clean up	Priority Mtg.	
2(+)		develop	develop	develop	develop	develop	
3		develop	develop	develop	QA prep & develop	develop & RC1	
4		Testing & Bug-fix	Testing & Bug-fix	Testing & Bug-fix	Testing & Bug-fix	Testing & Bug-fix	

QA Prep: All QA / Scan hosts are re-cloned to mirror production.

Release Candidate 1 (RC1): New development is ‘frozen’ at end of business. Items that did not make the release are flagged and reprioritized for the future. A change list will be prepared and distributed to Testing Team.

Develop: Items (tickets and projects) prioritized for development are worked on & high / critical issues as-needed.

Bug-fix: Tickets generated from testing round are addressed and any high / critical issues as-needed.

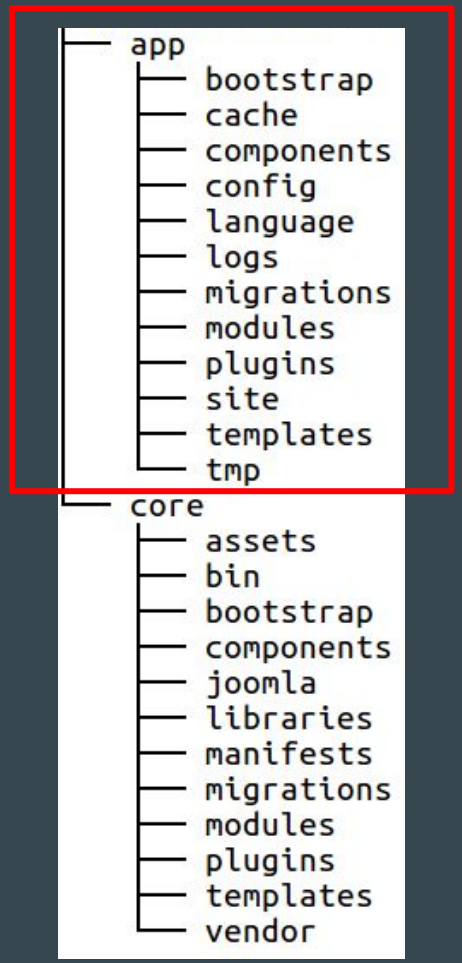
Testing: Expectation that students, community managers, and hub owners verify fixes and newly introduced features. Will supplement with automated testing in future.

Clean-up: Tickets are closed, asana is updated, major breakages are hot-fixed.

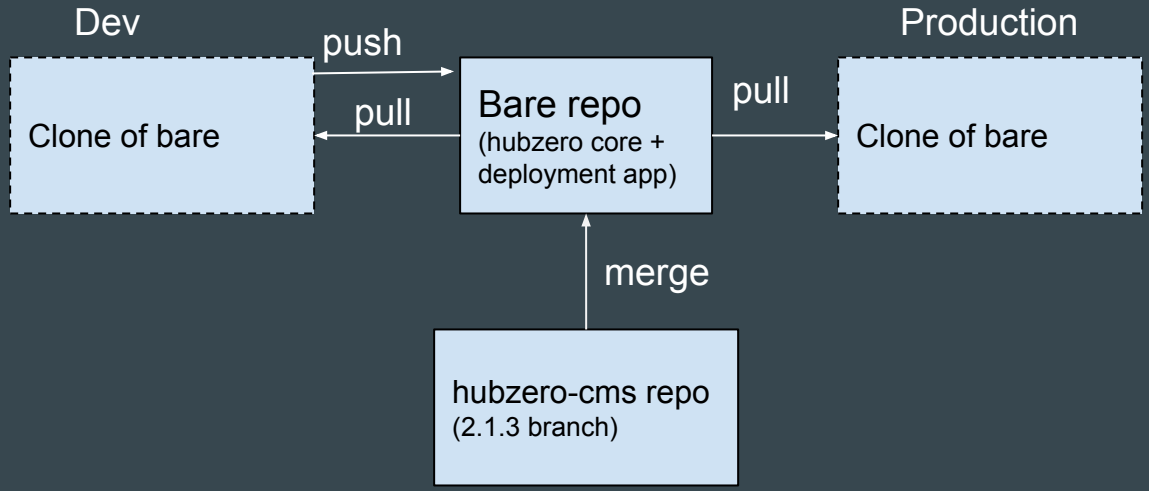
Hosted Hub Flow (app directory)

HUBzero CMS Structure

- The CMS has two main directories:
  - app - hub-specific code
    - Custom extensions
    - Template
    - Configuration
    - Uploads (app/site)
  - core - common hub code
    - Libraries (core/vendor)
    - Standard distribution extensions
    - Default templates
    - Composer

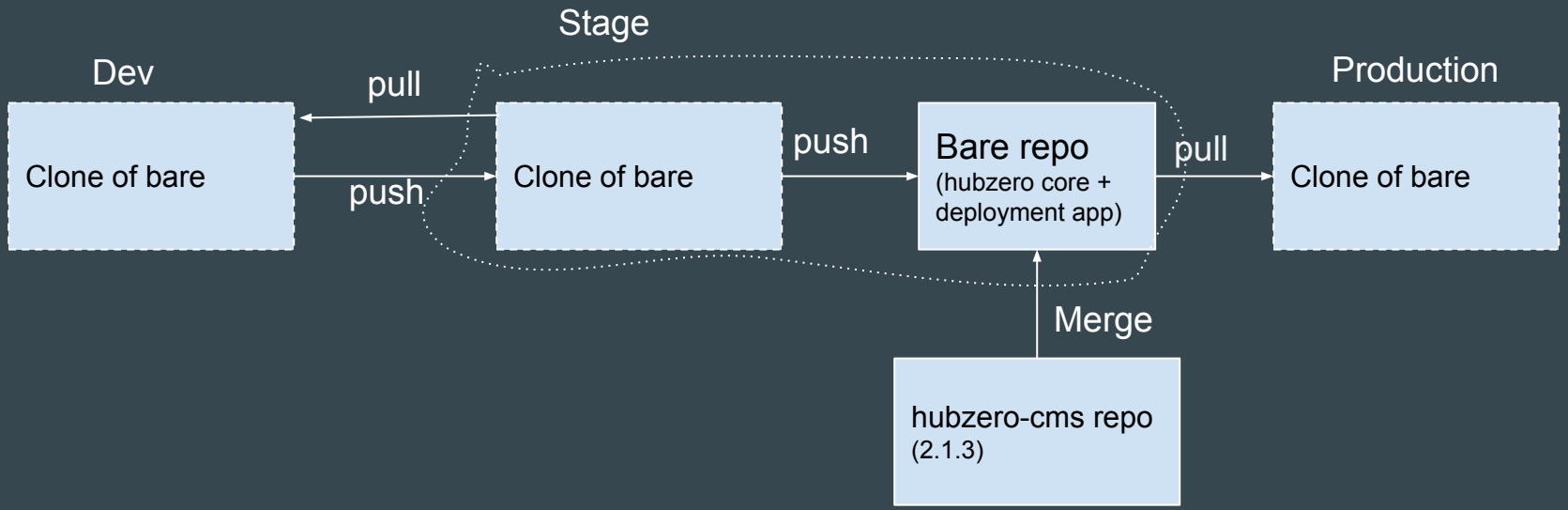


Hub Update Flow - dev/prod configuration



https://github.com/hubzero/hubzero-cms

Hub Update Flow - dev/stage/prod configuration



https://githhub.com/hubzero/hubzero-cms

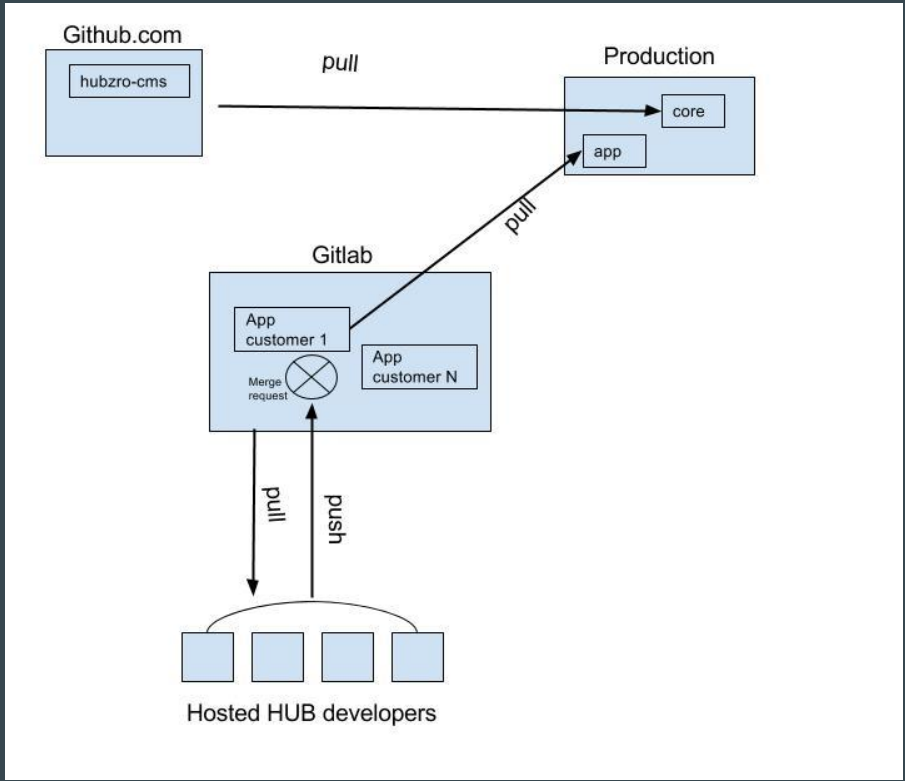
Contributing Changes in the App Directory (dev/stage)

For Today:

1. Sync your local branch with stage/prod.
    a. Git pull --rebase
2. Make your changes.
3. Test your changes on dev.
4. Commit the changes.
    a. git add <files>
    b. git commit -am “Message”
5. If you have a stage machine, git push.
6. If you have a stage machine: test on stage.
7. Submit ticket to pull onto production on next release cycle*

* When RC1 is created, it is difficult to promote changes to production.

Separation of App Directory



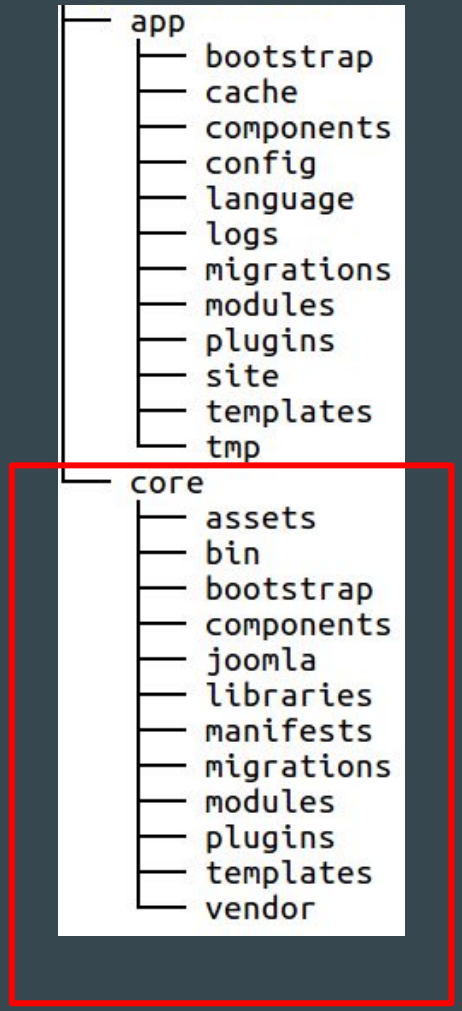
In the future:

1. Make changes.
2. Push to your gitlab repository.
3. Submit Merge Request
4. Wait for approval and automatically pushed to production.
    a. Automated checks for style and linting.
    b. HUBzero Developer Review
        i. Security
        ii. Stability
        iii. Quality
    c. Not dependant on release cycle.

HUBzero Core Contributions

HUBzero CMS Structure

- The CMS has two main directories:
  - app - hub-specific code
    - Custom extensions
    - Template
    - Configuration
    - Uploads (app/site)
  - core - common hub code
    - Libraries (core/vendor)
    - Standard distribution extensions
    - Default templates
    - Composer



Materials

- An account on github.com
- A development environment
  - Preference given to personal development environment
  - AWS, Vagrant, (Docker coming soon), VMware, or your own hub from packages.
- Git
  - yum/apt-get install git
- Recommended: tig
  - yum/apt-get install tig

Making a Core Contribution with Github

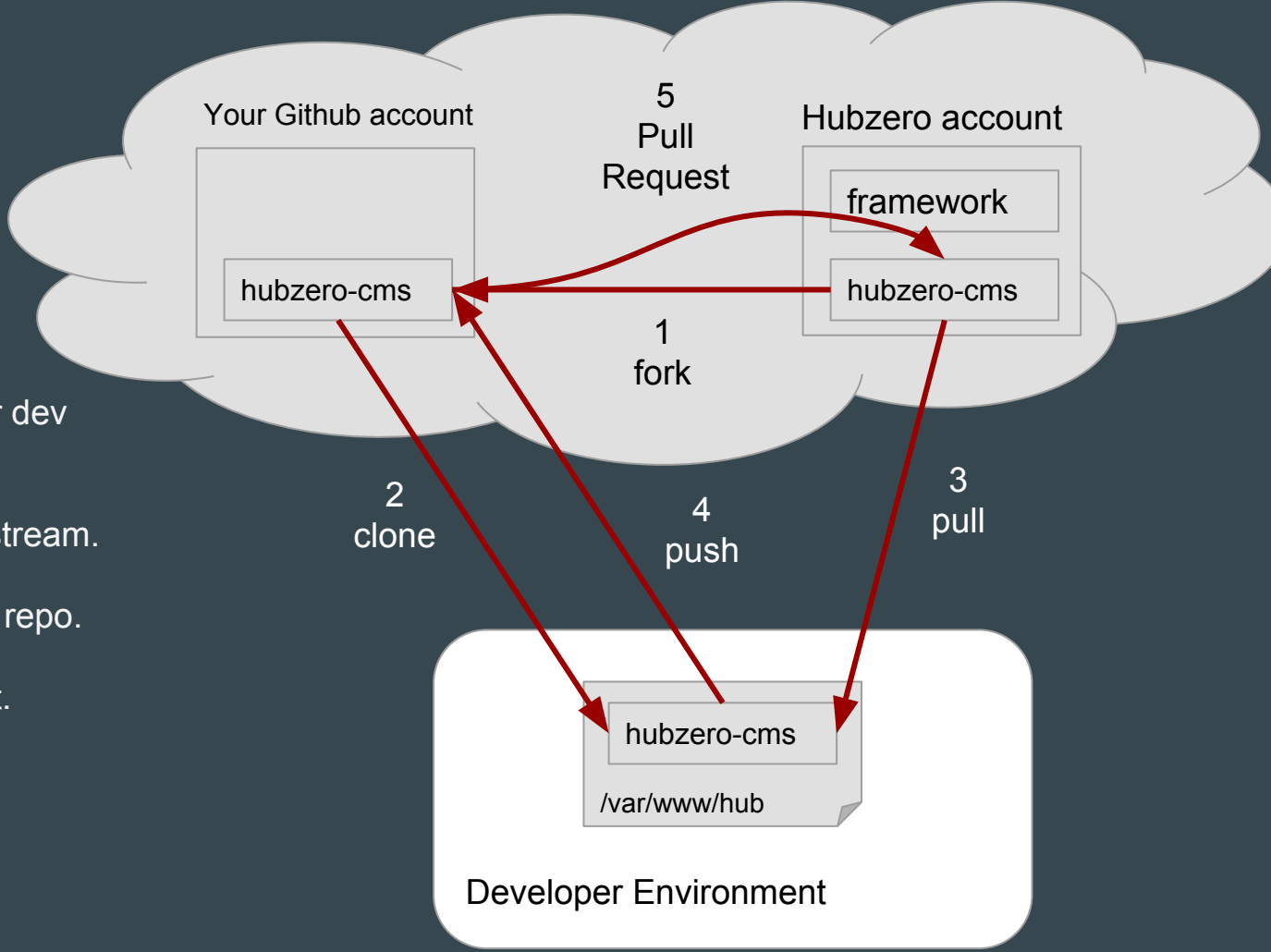
1. Fork the hubzero-cms repository.
    a. Creates <username>/hubzero-cms
2. Clone your fork.
3. Add upstream to pull down latest changes.
4. Update your local branch.
5. Make a feature branch.
6. Make your changes.
7. Keep your feature branch up-to-date.
8. Push to your fork.
9. Submit a pull request.
    a. Compares your fork with the original.
10. Switch back to local branch.
11. Update your local branch.

This flow is for those who want to contribute code to the ‘core’ directory of the hubzero-cms repository:
https://github.com/hubzero/hubzero-cms

These instructions also carry over to the hubzero framework repository:
https://github.com/hubzero/framework

Github Flow (simplified)

1. Fork the repo.
2. Clone the repo to your dev Environment.
3. Pull changes from upstream.
4. Push changes to your repo.
5. Submit a Pull Request.
6. Repeat steps 3 - 5.



Wrap Up

- Understand the organization of the Hubzero CMS repository.
- Understand the setup of Hosted Hub environment.
- See how to make changes to app directory extensions on Hosted Hub environment.
- See how to submit changes to Hubzero Core (hubzero/hubzero-cms).
- Understand the Hubzero Release Cycle.
- Preview proposed separation of app & core directories in the CMS.
- Reinforce some git basics.

Useful Git[hub] Documentation

- Forking a repo & keeping it up-to-date
  - https://help.github.com/articles/fork-a-repo/
- Submitting a pull request
  - https://help.github.com/articles/creating-a-pull-request/
- Tutorial on almost everything you can do with git
  - http://www.vogella.com/tutorials/Git/article.html
- The git Book
  - https://git-scm.com/book/en/v2
- HUBzero Documentation
  - https://hubzero.org/documentation/2.1.0/webdevs/index.contributions

Git Command Cheatsheet

- git pull --rebase <remote> <branch>
  - Rewrites history to keep tree flat.
- git remote add <name> <url>
  - Adds a remote called name from a url or file system path.
- git fetch --all
  - Updates information of all remotes.
  - Does not modify the branch.
- git add <files>
  - Stages a file for commit.
- git rm <files>
  - Remove a file from history in this commit.
- git commit -m “Message”
  - Forms a commit out of staged files.
- git remote -v
  - Shows the remotes of your repository.
- git push origin <branch>
  - Allows you to push to the origin remote.
- git push upstream <branch>
  - Only Senior-Level HUBzero Team Members have access to do this on hubzero/hubzero-cms.
- git checkout -b <feature-branch>
  - Creates a branch from the currently checked out branch and switches to the new branch.
  - same history at this point
- git reset <ref> --hard
  - Resets to a known point.
- git log
  - Shows the commit log.
- git status
  - Shows the current state of your repository.

Questions?