Git Workflow (Basic)

Change “master” to the name of your main development branch if it is not “master”

1. `git pull`
2. `git checkout -b <TASK-ID> origin/master`
3. Do the following periodically:
   - Commit changes (`git add` and `git commit`)
   - Get new changes (`git pull`)
4. `git checkout master`
5. `git pull`
6. `git merge --no-ff --no-commit <TASK-ID>`
7. `git commit`
8. `git push`

Advanced Workflow Changes/Notes

- Use `git pull --rebase` to pull down changes without an additional merge commit.
  - NOTE: Uncommitted changes need to be stashed first.
- To rebase instead of merge replace `git merge` . . . with:
  1. `git checkout <TASK-ID>`
  2. `git rebase master`
  3. `git checkout master`
  4. `git merge <TASK-ID>`
- Use `git commit -v` to show the diff of the changes while editing the commit message.
- Use `git rebase -i master` to rebase your current branch onto the newest commits in the master.

Finding a commit that introduced a bug (Manually)

1. `git bisect start`
2. `git bisect bad`
3. `git bisect good <commit_id>`
4. For each commit bisect makes you examine:
   - Run tests.
   - If tests fail: `git bisect bad`
   - If tests pass: `git bisect good`
5. When done: `git bisect reset`

Finding a commit that introduced a bug (Automated)

1. `git bisect start HEAD <commit_id>`
2. `git bisect run <test_script>`
3. When done: `git bisect reset`

The test script must return 0 upon success or anything from 1-127 excluding 125 for failure. Exit code 125 means the source cannot be tested, therefore that revision is skipped.

Gotchas/Tips

- Never rebase or amend commits that have already been pushed to the server
- Try to keep commits small. This makes moving changes between branches easier
- Create aliases for complex commands that are used often, such as showing the log history as a graph or the merge statement in the basic git workflow.
- Read the command output as it will give you instructions for what to do. For example, the output of “git status” will tell you how to un-stage a file and more.

Tags

Create an annotated tag
`git tag -a <tag_name> -m '<description>'`

List current tags: `git tag -l`

Delete a tag: `git tag -d <tag_name>`

Push tags to the server:
`git push origin --tags`

Delete remote tags:
`git push origin :refs/tags/<tag>`

Additional Information

Git Book: `http://git-scm.com/book`

Notes about git commit messages:
`http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html`

Some git tips: `http://mislav.uniqpath.com/2010/07/git-tips/`

Git log formatting tips: `http://www.jukie.net/bart/blog/pimping-out-git-log`

Merge vs. Rebase: Several comparisons of the two different techniques for combining changes from topic branches into the master branch.

- Use gitk to understand git merge and rebase

Configuration

Aliases:
```
[alias]
    co = commit -v
    me = merge --no-ff --no-commit
```

Merge Option:
```
[merge]
conflictstyle = diff3
```

Commits

Delete last N commits; unstage the changes:
```
git reset HEAD~N
```
Delete last N commits; keep staged changes:
```
git reset --soft HEAD~N
```
Delete last N commits:
```
git reset --hard HEAD~N
```
Modify the last commit: `git commit --amend`

Modify a series of commits:
```
git rebase -i <start_rev>
```
- Read the git book for more information before trying this command!

View the details of a specific commit:
```
git show <commit_id>
```
Copy a commit to the current branch:
```
git cherry-pick <commit_id>
```
Remove a file from every commit:
```
git filter-branch --tree-filter
’rm -f <filename>’ HEAD
```
- Use this command only when necessary!
- Other developers should not be using the tree when this is occurring and download a new clone after the changes have been made.

Go back to a previous version of a single file:
```
git checkout <version> <filename>
```

Log/History

View the log for the current branch: `git log`

View the log as a graph:
```
git log --graph --oneline
```
Git log with graph and more information:
```
git log --graph
--pretty=format:’\t%Cred%h
%Cgreen(%ad)%Creset
|%C(yellow)%d%Creset
%s %C(bold blue)<%an>%Creset’
--abbrev-commit --date=short
```
Show who modified a file: `git blame`

Uncommitted Changes

Show diff of unstanged changes: `git diff`

Show diff of stanged changes:
```
git diff --stage
```
Stage a file to be committed:
```
git add <filename>
```
Stage files interactively: `git add -i`

Remove deleted files from git: `git add -u`

Unstage a file: `git reset HEAD <file>`

Reset the current working directory:
```
git reset --hard
```
- WARNING: Unsaved changes will be lost!

Discard changes to a single file
```
git checkout -- <file>
```
Stash changes temporarily: `git stash`

Apply stashed changes: `git stash apply`

Clear the stash: `git stash clear`

Branches

List local branches: `git branch`

Show details about all branches:
```
git branch -a
```
Show which branches have not been merged:
```
git branch --no-merged
```
Delete a local branch:
```
git branch -d <branch_name>
```
Switch to a branch:
```
git checkout <branch_name>
```
Create a branch from a tag:
```
git checkout <branch_name> <tag_name>
```
Checkout a remote branch:
```
git checkout --track origin/<br_name>
```
Push a branch to the server:
```
git push origin <br_name>
```
Rename a branch: `git branch -m <new_name>`

Other

Open the git gui: `git gui`

Get a list of files that changed:
```
git log --name-only --pretty=oneline
--full-index <start_rev>..<end_rev>
| grep -vE ’^[0-9a-f]{40} ’
| sort | uniq
```
Create an archive from a git repo:
```
git archive <revision>
```
- Supports the following formats:
  - tar
  - tar.gz
  - zip

Get a “version id” for the current commit:
```
git describe --always --tags
```

Repositories/Remotes

Clone a new repository:
```
git clone <repository_url>
```
Show information about remotes:
```
git remote show origin
```
Show the URLs for the remotes:
```
git remote -v
```
Add a remote: `git remote add <name> <url>`

Remove a remote: `git remote rm <name>`

Download changes for all branches:
```
git fetch
```
- Good for reviewing changes before applying them. E.g. `git diff master origin/master`
- Will need to do a `git merge` or `git rebase` to apply the changes.

File Management

Delete a file: `git rm <file>`

Move a file: `git mv <from> <to>`

Clean untracked files from current dir:
```
git clean -f
```