Git Workflow

The different types of branches we may use are:

- `main`
  - production releases
- `develop`
  - AKA integration branch
- Release branches (`release-*`)
  - for finalizing a major/minor release, branched from `develop`
- Hotfix branches (`hotfix-*`)
  - for applying patches, branched from main (or release-* for older releases)
- Feature branches (`feature-*`)
  - for developing features or wild speculation, branched from `develop`

Feature branches

Created for feature development that may require several or more commits to produce a working tree. May make occasional merges from `develop` to keep it up to date. If it will be separate for an extended period of time, create a feature-branch-readme.txt in the top level directory dictating the reason for the branch.

May branch off from: `develop`

```
$ git checkout -b myfeature develop

```

Must merge back into: `develop`

```
$ git checkout develop
$ git merge --no-ff myfeature
$ git branch -d myfeature
$ git push origin develop

```

Release branches

- Release branches are created when the `develop` branch is at a stable point and release specific changes need to be made, such as bumping version numbers, etc. At that point, `develop` should be branched and the changes made before ultimately merging it into `main` and tagging the release.
- There should only be one active release branch at a time.
- Initial Release should be tagged as `x-alpha.1` or `x-beta.1`. Until the current release is wrapped up, merged into `main`.
- Development of the next release should take place on `develop`.
- When `develop` reaches another state of stability for release, another release branch is be created.

May branch off from: `develop`

```
$ git checkout -b release-1.2 develop
$ ./bump-version.sh 1.2
$ git commit -a -m "Bumped version number to 1.2"

```

Bug fixes made on a release branch may be merged back into `develop` continuously if needed, though ultimately they will be merged in when the release is finalized.

Must merge back into: `develop` and `main`

```
$ git checkout main
$ git merge --no-ff release-1.2
$ git tag v1.2

$ git checkout develop
$ git merge --no-ff release-1.2

```

Hotfix branches

Patches that need to be made to the most recent production release are applied to a hotfix branch off `main`. For older releases, hotfixes branch off a `release-*` branch (Explained below).

May branch off from: `main`

```
$ git checkout -b hotfix-1.2.1 main
$ ./bump-version.sh 1.2.1
$ git commit -a -m "Bumped version number to 1.2.1"

$ git commit -m "Fixed severe production problem"

```

Must merge back into: `develop` and `main`

```
$ git checkout main
$ git merge --no-ff hotfix-1.2.1
$ git tag 1.2.1

```

Merge into `develop` only if there is no current release branch, otherwise, merge into release branch instead.

Finally, delete. $ git branch -d hotfix-1.2.1

Hotfix branches (old release)

If `main` has moved on a point release (1.0, 1.1, 2.0, etc) and a hotfix must be applied to a older version (e.g 1.x):

- create a branch (e.g. `hotfix-1.1.1`), based on `release-1.x`
- fix the bug and merge `hotfix-1.1.1` back into `release-1.x`
- Do this for other older major releases as necessary

The release branch effectively becomes a `main` branch for a past version.