Feature Branches

Branch new features off from the develop branch:

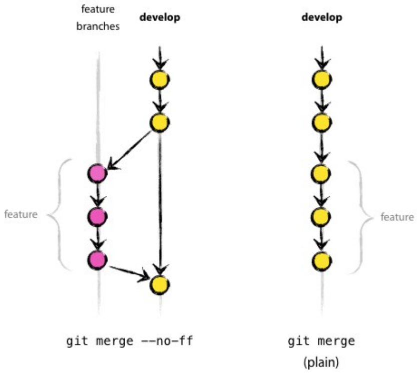
```
        $ git checkout -b myfeature develop
        Switched to a new branch "myfeature"
        [ update code and commit your changes for this feature …]
```

Incorporating a finished feature on develop

Merge finished features into the develop branch for the next release:

```
        $ git checkout develop
        Switched to branch 'develop'
        $ git merge --no-ff myfeature
        Updating ea1b82a..05e9557
          [...]
        $ git branch -d myfeature
        Deleted branch myfeature (was 05e9557).
        $ git push origin develop
```

The --no-ff *flag causes the merge to always create a new commit object, even if the merge could be performed with a fast-forward. This avoids losing information about the historical existence of a feature branch and groups together all commits that together added the feature*



Release Branches

May branch off from:	develop
Must merge back into:	develop and master
Branch naming convention:	release-*

Create a release branch

```
$ git checkout -b release-1.2 develop
   [...]
$ ./bump-version.sh 1.2 # some custom script
version bumped to 1.2.
$ git commit -a -m "Bumped version to 1.2"
[release-1.2 74d9424] Bumped version to 1.2
```

Finish a release branch

1. Merge the release branch into master :

```
        $ git checkout master
        Switched to branch 'master'
```

2. Next, tag that commit on master for future reference:

```
        $ git merge --no-ff release-1.2
        Merge made by recursive.
         [...]
        $ git tag -a 1.2
```

3. Merge the release branch into develop:

```
        $ git checkout develop
        Switched to branch 'develop'
        $ git merge --no-ff release-1.2
        Merge made by recursive.
         [...]
```

4. Delete the release branch; we don't need it anymore:

```
        $ git branch -d release-1.2
        Deleted branch release-1.2
```

Hotfix Branches

May branch off from:	master
Must merge back into:	develop and master
Branch naming convention:	hotfix-*

Hotfix branches are very much like release branches in that they are also meant to prepare for a new production release, albeit unplanned.

Create the hotfix branch

```
        $ git checkout -b hotfix-1.2.1 master
        Switched to a new branch "hotfix-1.2.1"
```

*Remember to bump the version number **after** branching off!*

```
        $ ./bump-version.sh 1.2.1 # some script
        version bumped to 1.2.1.

        $ git commit -a -m "version bump 1.2.1"
        [hotfix-1.2.1 41e61bb] version bump
        1.2.1
```

Then fix the bug in one or more separate commits.

```
        [ update code and commit your changes for this hotfix …]
```

Finally, summarize the hotfix commit series in a comment:

```
        $ git commit -m "Fixed problem"
```

Finish a hotfix branch

*When finished, merged back into master **and** back into develop to ensure that the bugfix is included in the next release.*

1. Update master and tag the release.

```
        $ git checkout master
        Switched to branch 'master'

        $ git merge --no-ff hotfix-1.2.1
        Merge made by recursive.
         [...]

        # use the -s or -u <key> flags to sign
        # the tag cryptographically.
        $ git tag -a 1.2.1
```

2. Include the bugfix in develop, too:

```
        $ git checkout develop
        Switched to branch 'develop'

        $ git merge --no-ff hotfix-1.2.1
        Merge made by recursive.
         [...]
```

3. Remove the temporary branch:

```
        $ git branch -d hotfix-1.2.1
        Deleted branch hotfix-1.2.1
```

Notes

This command will default git merge --no-ff

```
$ git config branch.master.mergeoptions  "--no-ff"
```

Other useful things to remember:

```
$ git push --all # push all branches
$ git pull --all # pull all branches
$ git push origin mybranch # push a specific branch
```