Workflow Model

Gitflow utilizes the core feature of Git, which is the power of branches. In this model, a repository has two core branches:

1. **Master/Main—**This is a highly stable branch that is always production-ready and contains the last release version of source code in production. (For the purposes of this article, we will be referring to this branch as “main”).
2. **Develop—**Derived from the main branch, the development branch serves as a branch for integrating different features planned for an upcoming release. This branch may or may not be as stable as the main branch. It is where developers collaborate and merge feature branches.

Main and Develop Branches

The main and develop branches form the base of any repository in Git.
The main branch contains a version of the code that is in production, and the develop branch contains a version that is due to be released in an upcoming version. Execute the following commands in the command prompt to check out the main branch in your system.

```
git checkout main
git push -u origin main
```

You will be prompted for username and password; enter them to proceed. Next, you will need to push the develop branch to a remote repository (i.e. from your system to sync with GitHub). Since the develop branch is only on your local machine, it has to be moved to a remote repository by executing the following commands.

```
git checkout develop
git push origin develop
```

Now you have a repository containing main and develop branches copied from your local machine.

Apart from those two primary branches, there are other branches in the workflow:

1. **Feature—**This derives from the develop branch and is used to develop features.

Feature Branch

The feature branch splits from the develop branch and merges back to the develop branch after a feature is complete. The conventional naming of this branch starts with **feature/***. This branch is mostly created and used by developers collaborating with teams. The purpose of the feature branch is to develop small modules of a feature in a project.

You might wonder why developers can't work directly from the develop branch. Why do they need to branch off to a feature branch? To explain this, consider a scenario where you are developing a feature and management decides to drop that feature, as it is no longer required or there is less feasibility of implementing it.

At that time, if you are working in the develop branch directly, it would create a lot of conflicts and possibly break the existing code. Also, to do this you would need to manually delete or comment out code.

Instead, if you branched off a separate feature branch, you could silently discard and delete that branch without affecting the develop branch. Not only does this help develop features, which require the trial-and-error technique, but by using a separate branch you also get an extra level of stability in the develop branch because code from the feature branch undergoes several levels of code reviews and quality assessment before merging into the develop branch.

The lifetime of a feature branch ends once it merges with the develop branch. If multiple developers or teams are working on the same feature, it's easier for them to collaborate by working on a common feature branch.

To start a feature branch (the name of the feature branch will be the name of the feature; we are using **feature1** in this example).

```
git flow feature start feature1
```

Once done, the status of the changes can be checked for newly added or modified files.

```
git status
git add .
git commit -am "Your message"
```

The following commands publish the feature to the remote repository.

```
git flow publish feature1
git push
```

Once a feature is complete and the code has been reviewed, you can complete your work in a branch by issuing the command below. Upon execution, the code will be merged to the development branch automatically and the feature branch will be deleted from the remote repository.

```
git flow finish feature1
```

If you need to delete a branch, you can execute:

```
git branch -d feature/feature1
```

2. **Release**—This also derives from develop branch but is used during releases.
3. The release branch derives from the develop branch and merges back into the develop and main branches after completion of a release. By convention, the naming of this branch starts with `release/*`. This branch is created and used when features are completed and finalized for a versioned release.
4. Why can't we directly release from the develop branch? Because the sole purpose of the release branch is to isolate a version of a release that is final but needs some quality assistance and stability from the upcoming version. If we branch off from the release branch, then other developers who are assigned to develop features for an upcoming release and are not involved in the release stability process can continue to development and merge their features into the develop branch without waiting on or affecting the current release process. The release branch helps isolate the development of an upcoming version and the current release.
5. The release branch's lifetime ends when a particular version of a project is released. Once this branch merges into the develop and main branches, it can be deleted. And once you have done this, you can tag a main branch with a particular release version—let's say **v1.0**—to create a historical milestone.
6. The following example explains how a release branch can be created and published using the Gitflow extension from the command prompt.

1. Start a release branch.

```
git checkout develop
git pull
git flow release start release1
```

2. Commit newly added or modified changes and push to the remote repository.

```
git add .
git commit -am "Your message"
git flow publish release1
git push
```

3. Merge changes to the develop branch.

```
git checkout develop
git merge release/release1
```

4. After a release, merge changes to the main branch.

```
git checkout release/release1
git pull
git flow release finish release1
```

-or-

```
git flow release finish -m "Your message" "release1"
git checkout main
git push --all origin
```

7. **Hotfix**—This derives from the main branch and is used to fix a bug in the production branch that was identified after a release.

The hotfix branch is derived from the main branch and merged back after completion to the develop and main branches. By convention, the name of this branch starts with `hotfix/*`. This branch is created and used after a particular version of the product is released to provide critical bug fixes for the production version.

The reason we do this is because one problem you might face when branching off from the develop branch is that some of your developers would have already started work for the upcoming release while you are in the middle of the current release. Your release would contain the next version of features, which are not finalized, but you only need to provide bug fixes for the current version. Instead of branching off from develop branch, you can branch off from the main branch, as that branch contains only the current version of the code in production. This way, branching off from the main branch will not affect your production or development version of the product.

The hotfix branch can be deleted once a critical fix for the current production version is released and merged with the main and development branches. Once you have done this, you can again tag the main branch with an iterative subversion of the release; let's say v**1.1**.

This example shows how the `hotfix1` branch can be created and published using the Gitflow extension from a command prompt.

1. Start a new hotfix branch and commit changes after modifications.

```
git checkout develop
git flow hotfix start hotfix1
git status
git add .
git commit -am "Your message"
```

2. Publish the branch to the remote repository.

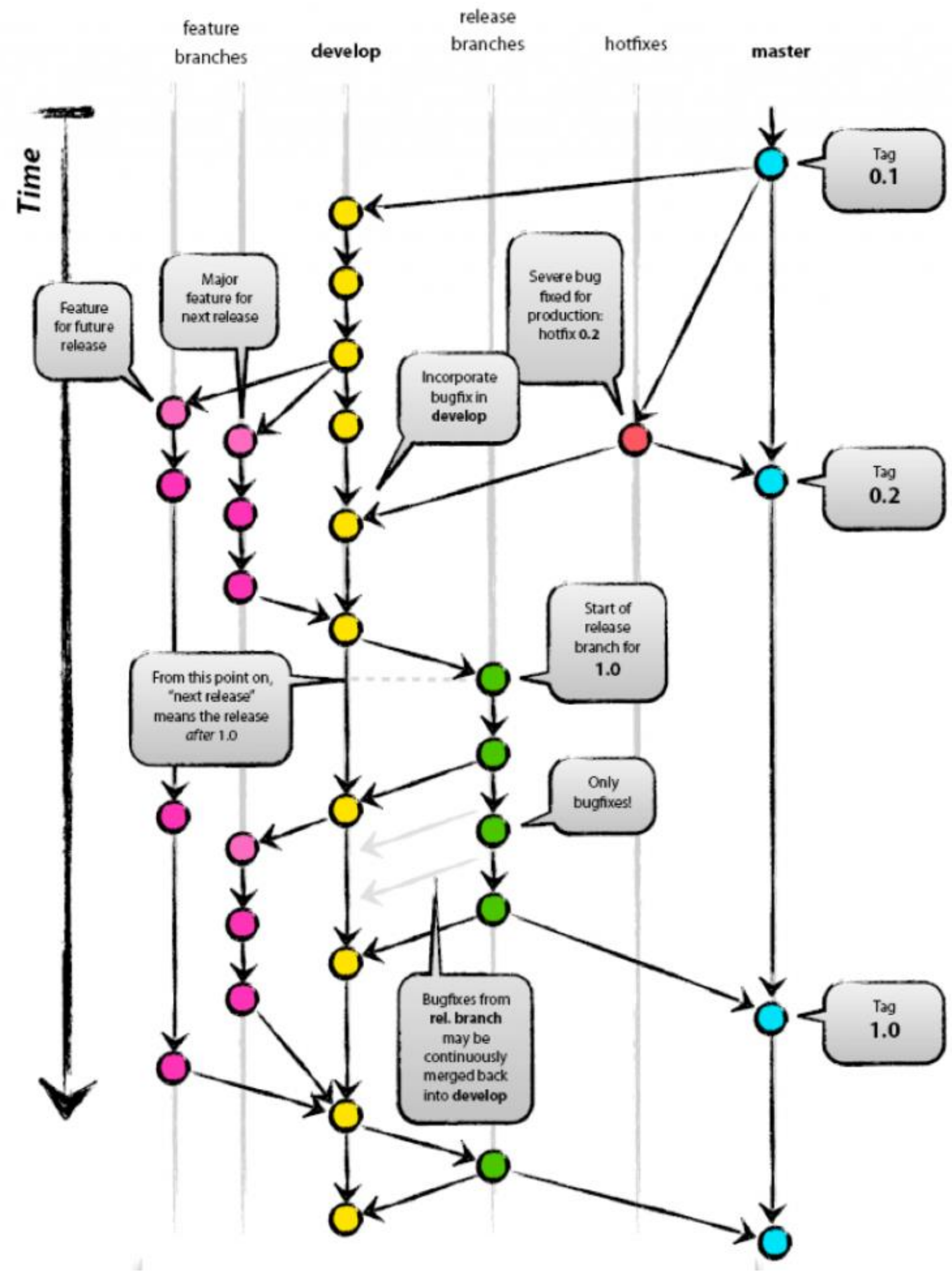
```
3. git flow publish hotfix1
```

4. Merge changes to remote repository.

```
git flow hotfix finish hotfix1
```

-or-

```
git flow hotfix finish -m "Your message" "hotfix1"
git status
git checkout main
git push --all origin
```


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Severe bug fixed for production: hotfix 0.2
Major feature for next release
Feature for future release
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Setting up Gitflow in Your Repository

The following set of commands takes you through the Gitflow release branch process lifecycle.

To work with the release branch, first initialize a Gitflow repository.

```
$ git flow init

Initialized empty Git repository in C:/_tools/temp/.git/
```

Only two branches exist after initialization, and there are no tags.

```
$ git branch -a

* develop

master


$ git tag -l
```

Create a feature branch to represent some work that will make the project feature complete. Notice this adds a new branch named "feature/feature_branch."

```
$ git flow feature start feature_branch

Switched to a new branch 'feature/feature_branch'


$ git branch -a

develop

* feature/feature_branch

master
```

When the feature is finished, the feature branch is deleted.

```
$ git flow feature finish feature_branch

Switched to branch 'develop'


$ git branch -a

* develop

master
```

Use the “git flow release start” command to create the release branch.

```
$ git flow release start 0.1.0

Switched to a new branch 'release/0.1.0'


$ git branch -a

develop

master

* release/0.1.0
```

When the release is stable, run the “git flow release finish” command.

```
$ git flow release finish '0.1.0'

Already on 'master'

Deleted branch release/0.1.0 (was f9cdbf0).
```

After the “git flow release finish” command runs, the release branch should be deleted and a new Git tag exists in the repository.

```
$ git branch -a

develop
```

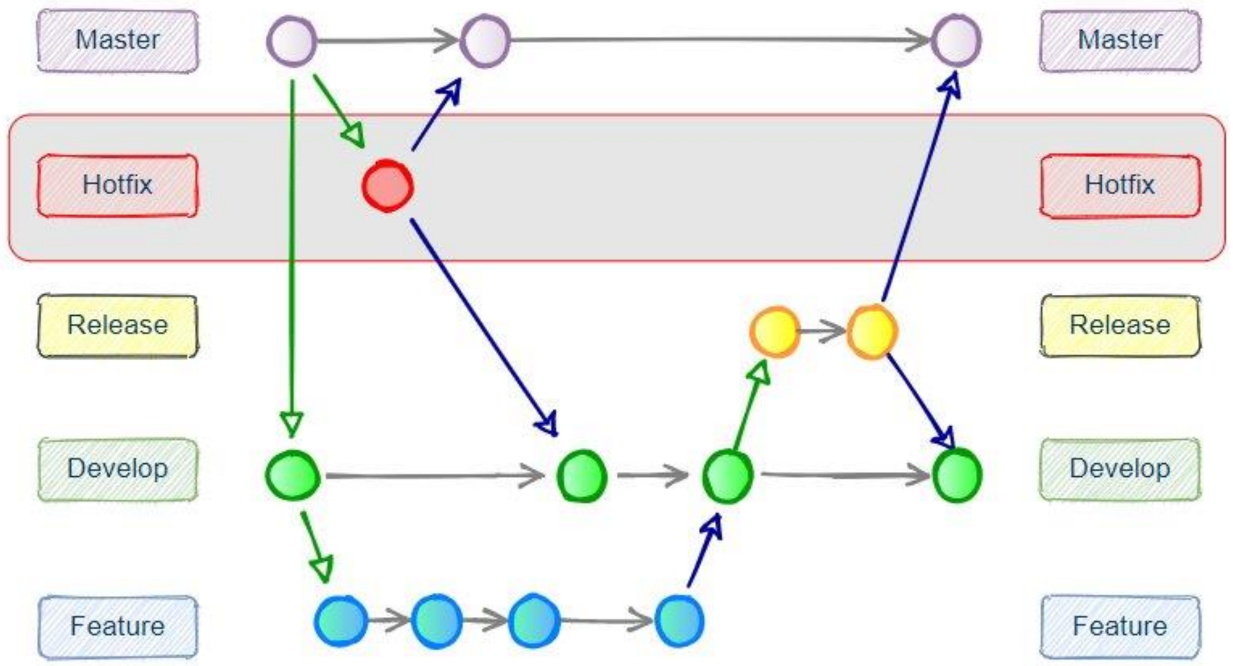
```
* master


$ git tag -l

0.1.0
```

Gitflow release branch process overview

In summary, the key features of the Gitflow release branch are:

- The release branch represents a complete feature set.
- The only commits on the release branch are for bug fixes and important chores.
- The Gitflow release branch is created off the development branch.
- Gitflow release is merged into master and also back into development.
- The release branch is the shortest lived of all Gitflow branches.

Gitflow Workflow Diagram


Master
Master
Hotfix
Hotfix
Release
Release
Develop
Develop
Feature
Feature