Git workflow

Table of Contents

1. Git basic concepts

- **Git provider**: Github, Gitlab, Bitbucket...

GitHub Bitbucket

GitLab Azure DevOps

1. Git basic concepts

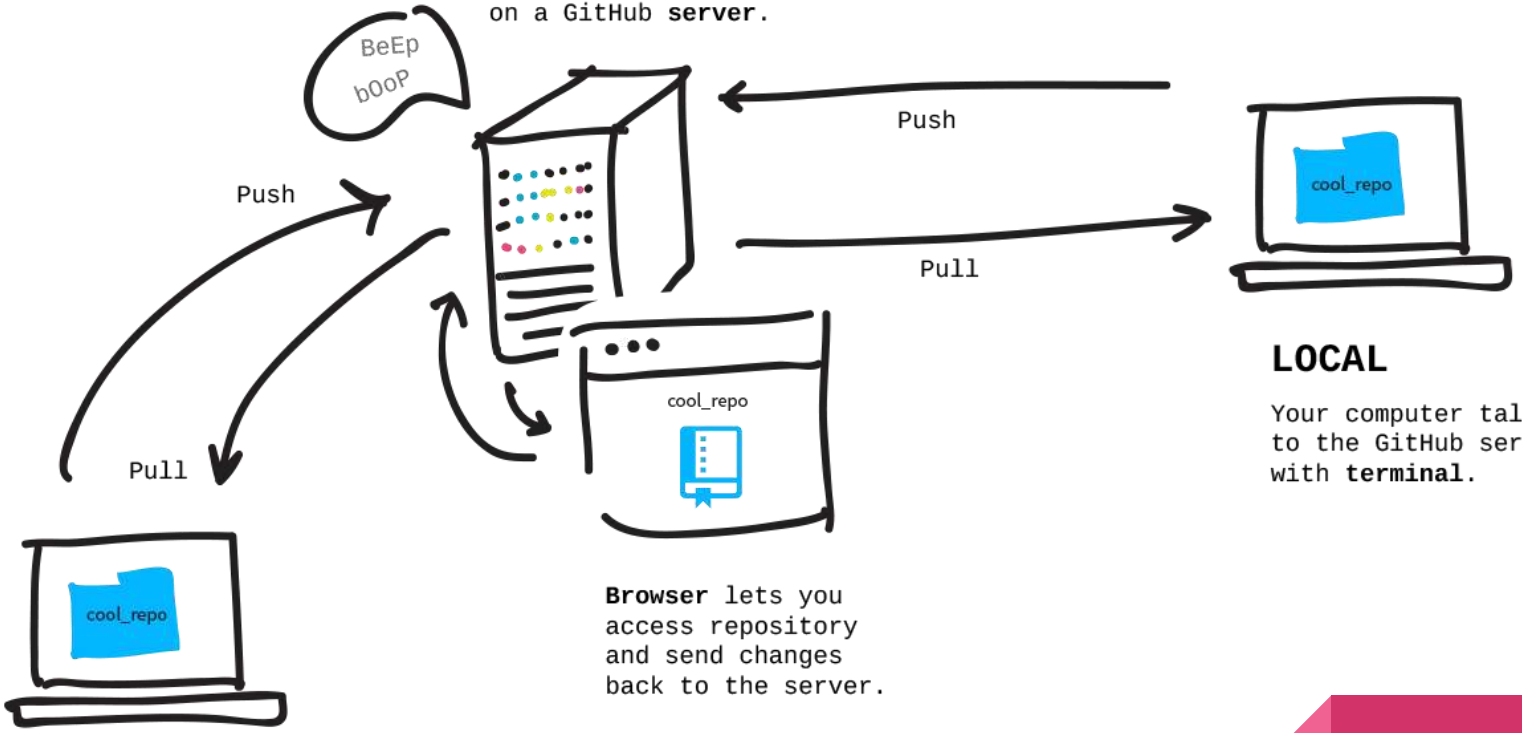
- **Git provider**: Github, Gitlab, Bitbucket...

- **Repository**: Git repo

- **Remote**: Remote Repository.

REMOTE

Repositories live
on a GitHub **server**.



LOCAL

Your computer talks
to the GitHub server
with **terminal**.

Browser lets you
access repository
and send changes
back to the server.

LOCAL

Someone else’s computer
talks to the GitHub server.

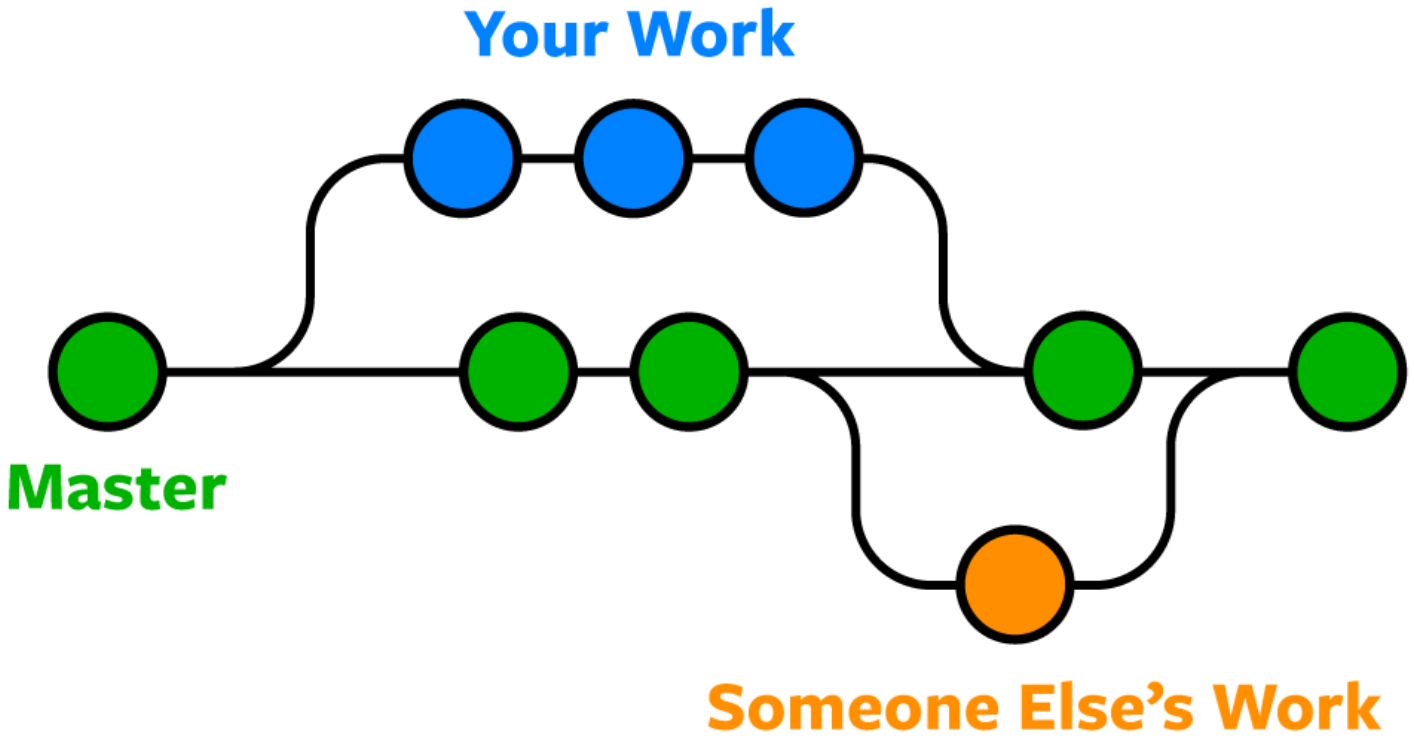
1. Git basic concepts

- **Git provider**: Github, Gitlab, Bitbucket...

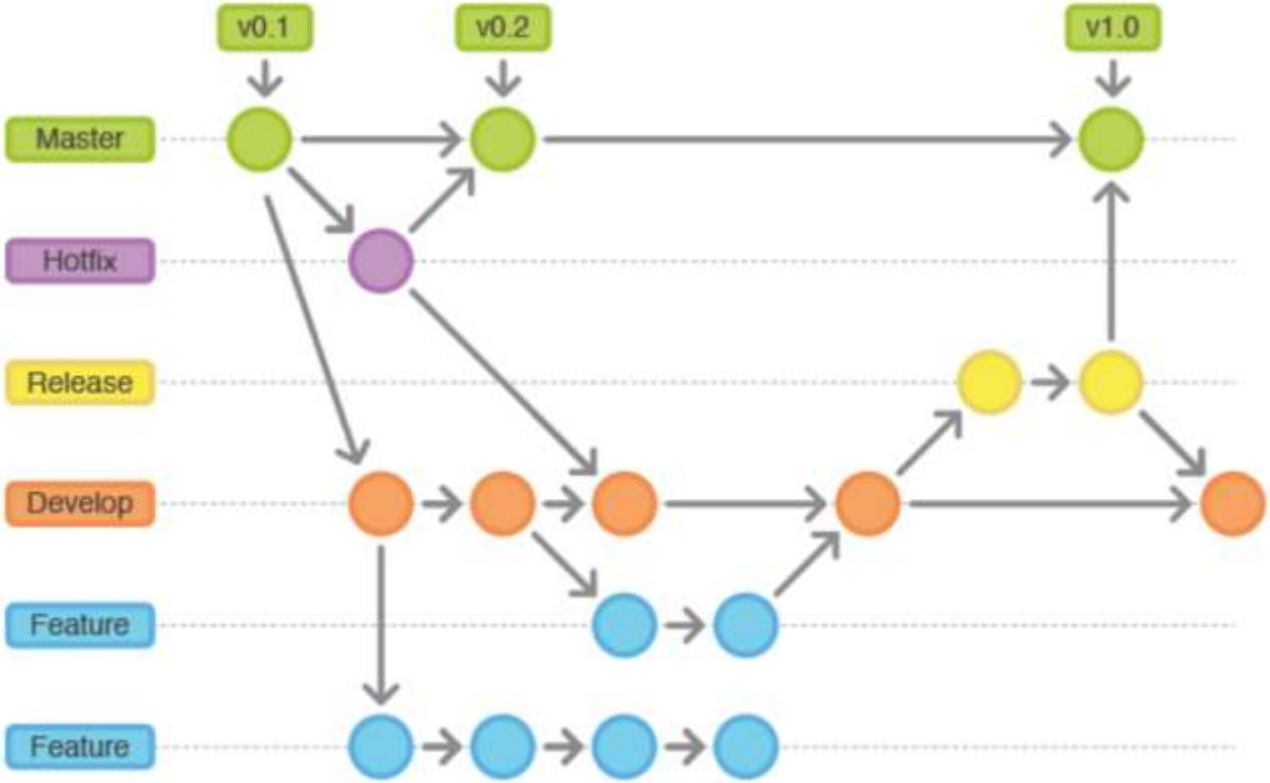
- **Repository**: Git repo

- **Remote**: Remote Repository.

- **Branch**: The branch of code when you work with git repo


Your Work
Master
Someone Else's Work

2. Git workflow



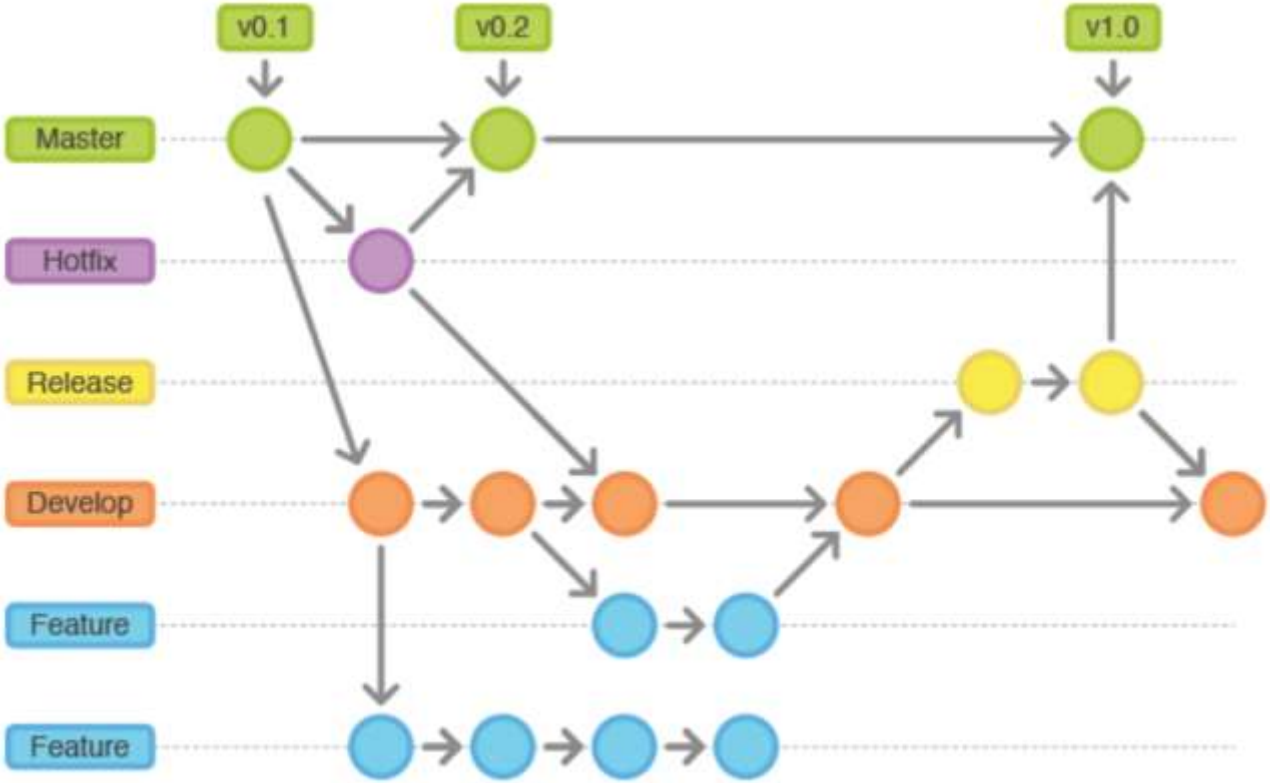
Gitflow Advantages

- Easy to manage the code from all developers
- We can make the rules when merge the code
- Easy to review code

2.1 Master branch

- Branch "master" is the main branch of the repository, always containing the source code that is in use on production.
- All source changes take place in another branch and are merged back into the "master".
- Do not commit directly, only owner can do merge/commit into master

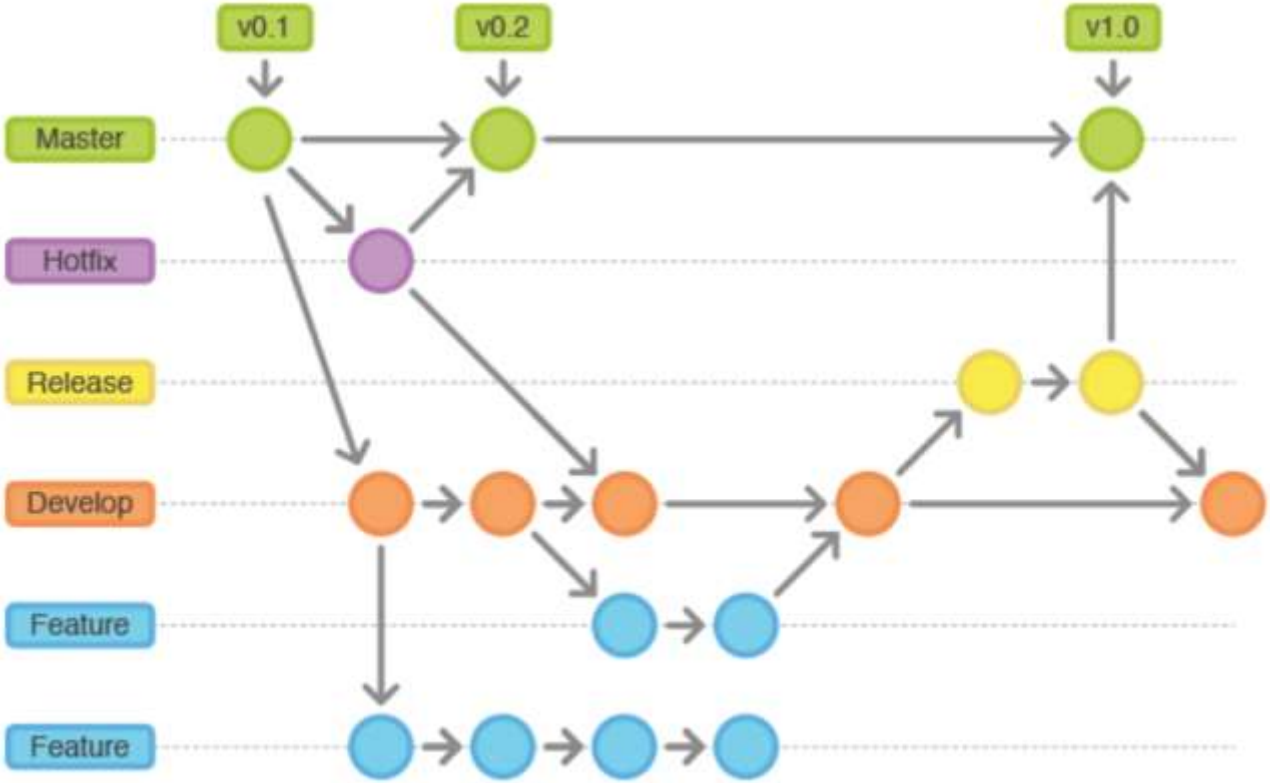
2.1 Master branch



2.1 Develop branch

- The "develop" branch is the main branch of the repository, which always contains the latest source code that is being developed for the next release.
- After changes, "develop" will be merged back into "master" to release to production.
- On the remote repository, only certain people are allowed to merge / commit into "develop".

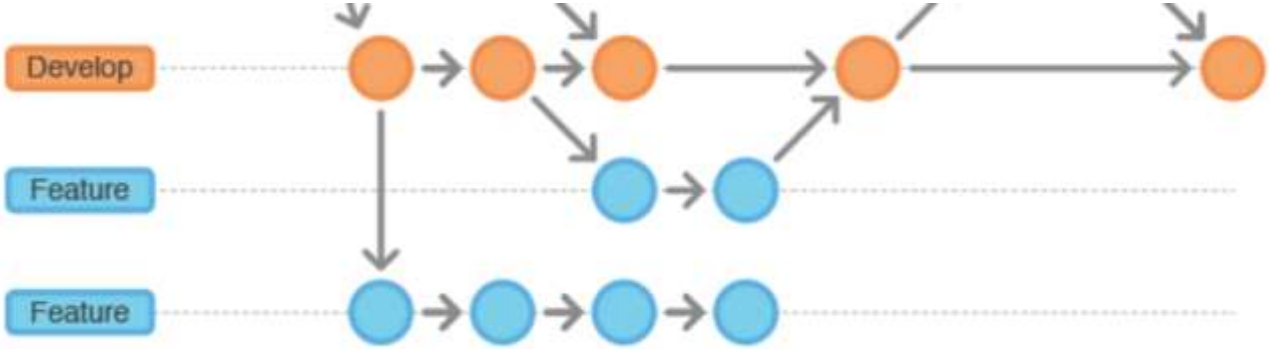
2.2 Develop branch



2.3 Other branches

- Feature branches
- Hotfix branches
- Release branches

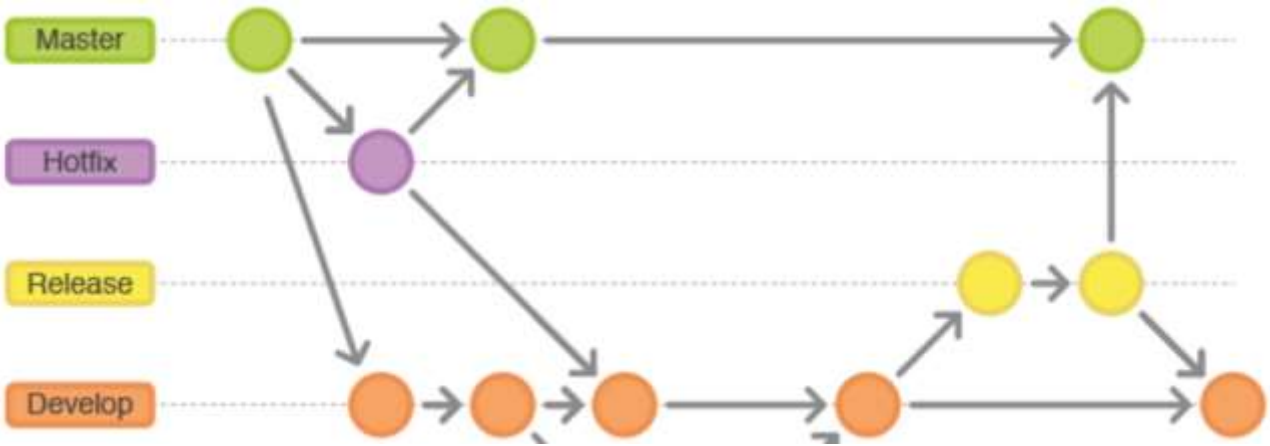
2.3.1 Feature branches



2.3.1 Feature branches

- Use to merge developer code for a particular function
- Be checked out from the "develop" branch
- When developing, you also need to checkout from the "feature" branch instead of "develop".
- When a function can be released, it will merge the "feature" into "develop" again.
- Only those with authority can merge code into feature branch, delete branch.
- Can be named according to convention "feature/TICKET-**-comment"

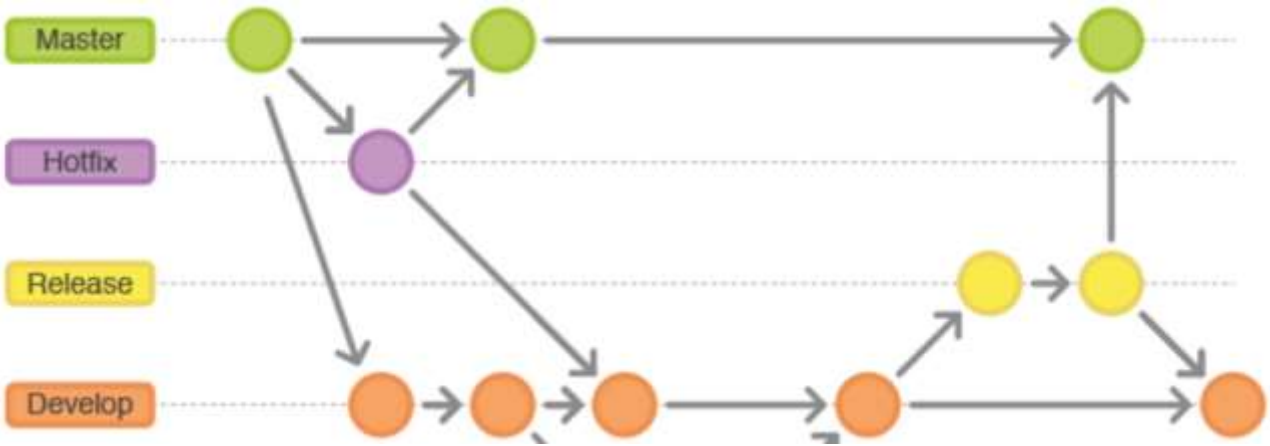
2.3.2 Hotfix branches



2.3.2 Hotfix branches

- When it is necessary to quickly fix a production error
- Checkout from "master" branch
- This branch needs to be merged and both "master" and "develop".
- Naming under the "hotfix/**" convention

2.3.3 Release branches

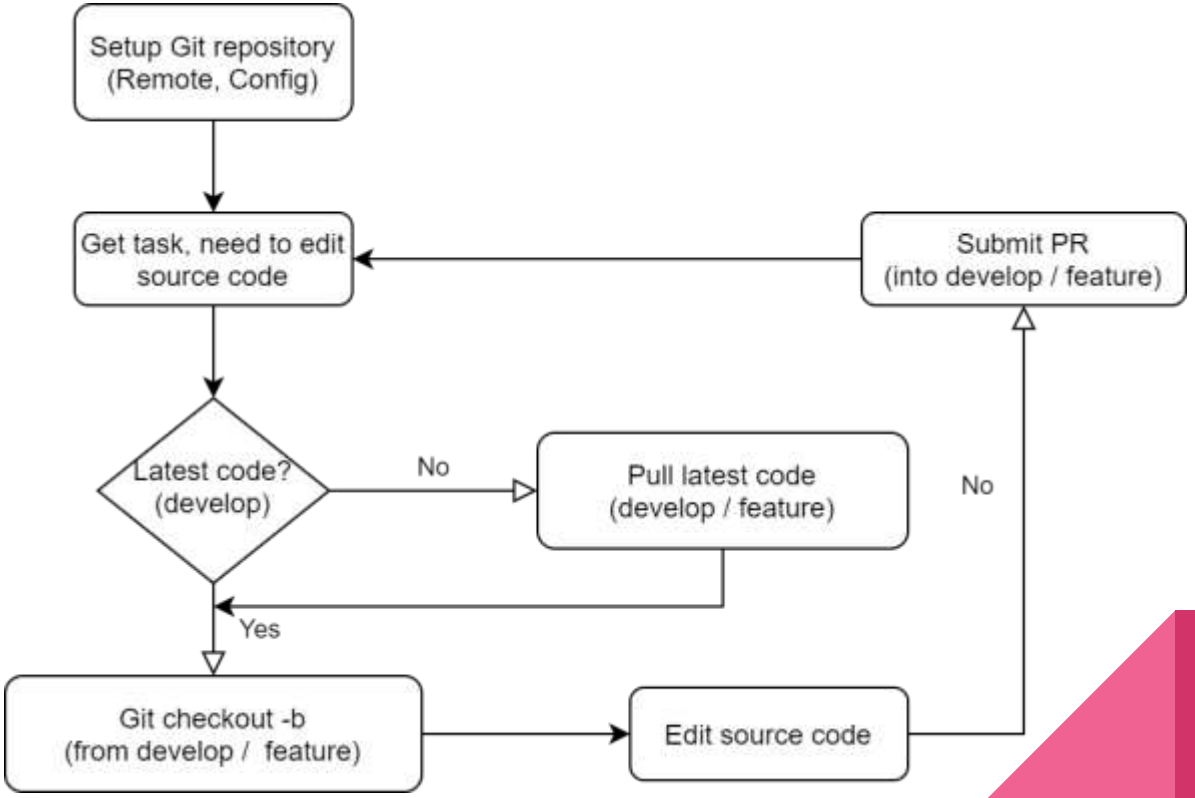


2.3.3 Release branches

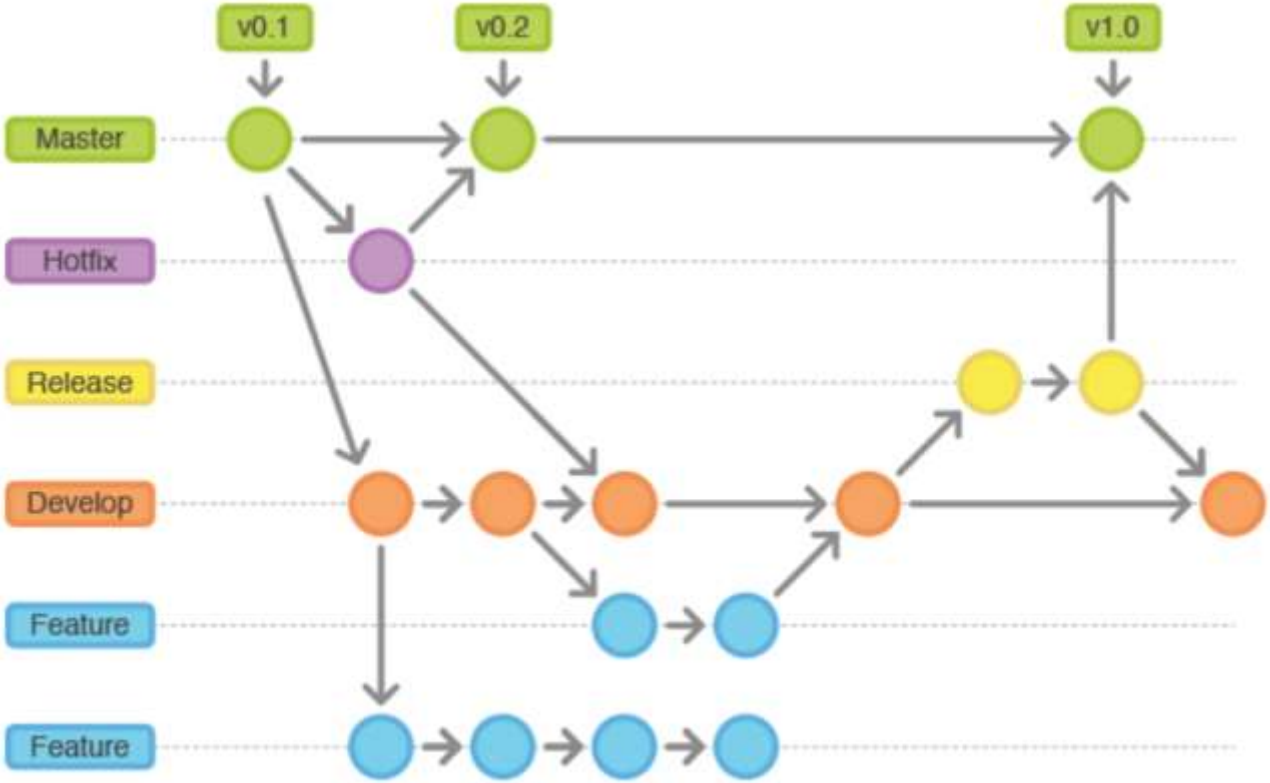
- Use to prepare for a new release to production
- Is separated from the "develop" branch. After the split, there will probably be more bug fix commits.
- Need to be merged back into both "develop" and "master".
- The name of the release branch should follow the "release-*" convention.

  Example: release-20200916, release-1.1.0

2.4 Dev workflow - Guideline



2. Git workflow (again)



3. Gitflow - Basic issue

- Conflict when creating Pull Request submit
- When starting to edit the code, checkout -b in the wrong branch
- Use git push -f to remote repo
- Use git rebase on the public branch
- Revert an unwanted commit
- Commit message doesn't make much sense

4. Commit Conventional

```
<type>[optional scope]: <description>

[optional body]

[optional footer]
```

- < >: Required
- [ ]: Optional

Shortened Commit

```
<type>[optional scope]: <description>

feat(bmr): add sql for meeting table
```

https://github.com/zeke/semantic-pull-requests

The pull request must be:

- type: lowercase - fix, feat, docs ...
- scope: is a noun, to classify the group of changes in commits
- description: a brief description of the changes in the commit

Advantages - Commit Conventional

- Simple set of general rules, easy to apply
- Ease of communicating modified content
- Compatible with Semantic Versioning
- Automatically generate CHANGELOGS
- Automatically determine new version based on commits
- People are easier to contribute: create commit, find and read commit history

“Type” - Commit conventional

- **feat:** adding a feature
- **fix:** fix bug for the system
- **refactor:** neither fix the bug nor add features or sometimes the bug is also fixed from refactor.
- **docs:** add / change document
- **chore:** minor modifications to the code
- **style:** changes that do not change the meaning of the code such as changing css / ui.
- **perf:** improved code in terms of processing performance
- **deps:** update versions for dependencies, packages

5. New Rules

- All repos must have PR Template

https://github.com/Seta-International/seta-policies/blob/main/pull-request-template.md

- PR editing interface must attach Screenshot, or Gif image perform test under local

- PR editing API must have a screenshot of Laravel Debugbar (*)

- Commit message written by Commit Conventional

- After editing is complete, you must comment on PR as "Done"

Basic Command - Checkout

Command:

```
git checkout -b feature/e2e-integrations

git checkout feature/e2e-integrations
```

- Create and checkout to branch

Basic Command - commit

Command:

```
git commit
git commit -m “feat: my first commit”
git commit -am “feat: my first commit”
```

- Create new commit for your code on your local machine

Basic Command - Pull / Fetch / Merge

https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch

Pull = Fetch + Merge

feature/e2e

git pull origin develop = git fetch origin develop + git merge origin develop

Basic Command - Push

The git push command is used to upload **local repository content** to a **remote repository**

```
git push <remote> <branch>
```

feature/e2e-integration

git commit => git push origin feature/e2e-integration

The end