UPskill

Digital Skills & Jobs

Version control best practices

Metodologias de Trabalho em Equipa

P.PORTO

apdc
ASSOCIAÇÃO PORTUGUESA PARA O DESENVOLVIMENTO DAS COMUNICAÇÕES

INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL

CONSELHO COORDENADOR DOS INSTITUTOS SUPERIORES POLITÉCNICOS

Why should I care about it?

- Makes your life easier
- Makes life easier for your team members
- Transparency
- Less decision making

Commit

- Keep your commits smaller
- Commit only files changed
- Keep commit message concise and meaningful

```
git commit –m "Added more stuff"
```

```
git commit –m "Add Car Controller"
```

Commit

- Split commit messages into Title and Description if it is too complicated
  - The title and the description are separated by the first blank line after the title. So you can:

```
git commit –m "Implementation

               Add Car Controller"
```

or

```
git commit –m "Implementation" –m "Add Car Controller"
```

Commit

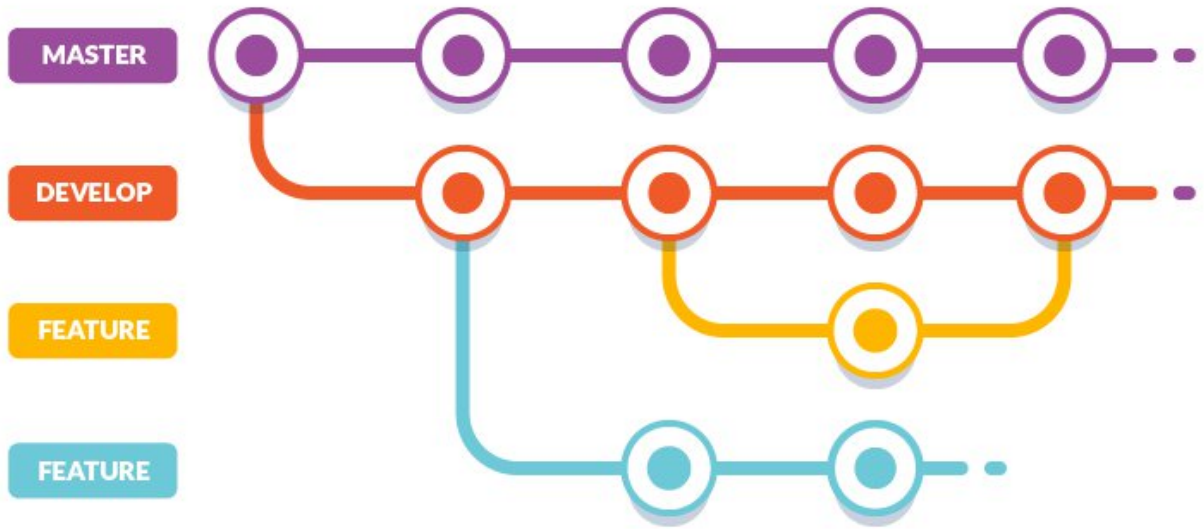
- Use a descriptive commit message
  - Indicates the purpose of the change;
  - Allow to look for changes related to a concept;
- Use verbs like **Fix**, **Add**, **Change**, **Update**
- Commit Messages should be imperative

Commit

- Don't commit generated files
  - Compiled binary files (such .o or .class)
  - Generated reports (pdf, xml, txt, ..)
  - Database files
  - IDE configuration files
- Make each commit a logical unit
  - Each commit should have a single purpose and should completely implement that purpose.
- Avoid indiscriminate commits
  - Empty commits (with no explicit files supplied). Commit every changed file.

Define a Git Branching Model

- Example:
  - Master
  - Develop
  - Feature
  - Releases
  - Hot Fixes



Branches and Purposes

- **Master**
  - Single source of truth
  - Always contains working codebase

- **Develop**
  - Holds current features built
  - Should contain code ready for testing or QA
  - Develop branched should be merged for Release branches

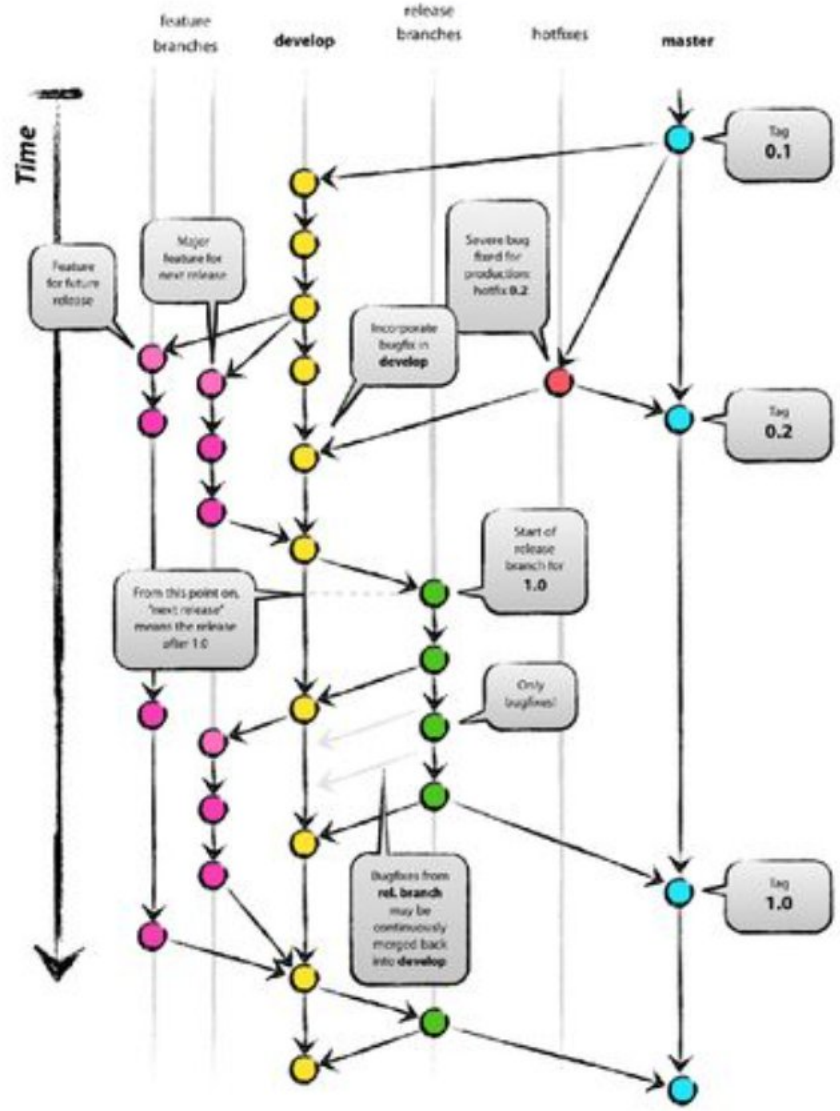
Branches and Purposes

- **Feature**
  - Holds feature under development
  - Different teams are working on different features and have their own feature branch
  - Should always be merged to develop branch and branched from develop branch

Branches and Purposes

- **Releases**
  - Holds the snapshots of version of codebase released
  - Release branches should be merged to Master

- **Hot Fixes**
  - Critical bug fixes and patches
  - Should be branched from Master and merged to Master
  - Hot fixes branch updates should also be merged with Develop branch



Git Best Practices

Branching Model

- Branching model used depends on:
  - Project size
  - Team size
  - Company

- You can use any branching template you like, but be consistent with it

Other Git best practices

- Incorporate others' changes frequently
- Share your changes frequently

Git Clients

- There are several third-party tools for users looking for platform-specific experience
  - SourceTree (https://www.sourcetreeapp.com/)
  - TortoiseGit (https://tortoisegit.org/)
  - GitKraken (https://www.gitkraken.com/)
- Most IDE has Git support
  - Intellij Idea (https://www.jetbrains.com/help/idea/set-up-a-git-repository.html)
  - Netbeans (https://netbeans.org/kb/docs/ide/git.html)
  - VS Code (https://code.visualstudio.com/docs/editor/versioncontrol)

UPskill

Digital Skills & Jobs

Version control best practices

Metodologias de Trabalho em Equipa

P.PORTO

apdc
ASSOCIAÇÃO PORTUGUESA PARA O DESENVOLVIMENTO DAS COMUNICAÇÕES

INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL

CONSELHO COORDENADOR DOS INSTITUTOS SUPERIORES POLITÉCNICOS