Git Best Practice

**TABLE OF CONTENTS**

[Best Practices 1](#_Toc1720439405)

[Naming 1](#_Toc563187559)

[General conventions 1](#_Toc1140858885)

[Service Users 1](#_Toc218798029)

[Resource and data source arguments 1](#_Toc896516024)

[Variables 2](#_Toc8454459)

[Outputs 2](#_Toc852868051)

[Resources: 2](#_Toc9478135)

# Foreword on main / master

The term **main**/**master** is inter-changeable, through this document it's be referred to as **main**, also **trunk** is a old name from a previous technology called svn that's very occasionally used

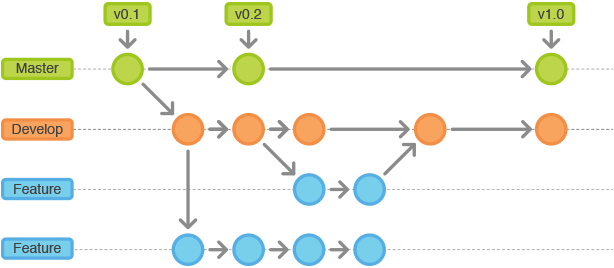
More info here on the change

<https://www.zdnet.com/article/github-to-replace-master-with-main-starting-next-month/>

# Best Practice

## Code Commit Process

Developers **SHOULD** follow the popular gitflow method for committing new code and features, see below for exception\*



<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

More information in this nice description of git flow  
<https://nvie.com/posts/a-successful-git-branching-model/>

### Cliff notes

* Codebase have three types of branches (main *for releases*, develop *for current development work*, and multiple feature/AB-XXX-xxxx branches *for individual ticket work*)
* Developers **MUST** not commit new work directly to main branch
* All new work **MUST** be in created in a separate branch (feature/TICKET-HUMAN-NAME aka feature/AB-379-tool-integration)
* Once work finished **MUST** be pull into a pull request to go to develop branch and sent for code review
* After Code Review the pull request is merged

### Exceptions \*

* Developers **MAY** commit documentation only or minor non-code changes directly to develop (should still reference ticket change is for)
* Release branch is a bit pointless, as long as you tag and know what you're doing, you can just tag develop and merge it into prod, instead of making a release branch and merging that (assuming main is a direct descendant of develop)

## Commit Message Format

* All git commits **MUST** have *meaningful* messages attached
* All git commit message **MUST** follow the following format
* TICKET\_NUMBER: HUMAN DESCRIPTION
* example of **GOOD** comment for a Azure Dev Ops Ticket
* AB#379: Tool Integration Library
* example of **GOOD** comment for a Jira Ticket
* AIMLCOE-379: Tool Integration Library
* example of a **BAD** comment for a Azure Dev Ops Ticket
* Tool Integration Library
* example of a **VERY BAD** comment for a Azure Dev Ops Ticket
* updated some files
* Reason for adding the ticket number and meaningful description is for provisioning a audit trail and helping a merge main if  
  conflicts arise, and linking the tickets to their retrospective board items.

*Special Case: House Keeping Commits*

* Developers **MAY** commit documentation only or minor non-code housekeeping changes directly to without a ticket, these should follow the safe format with mock ticket comment such as AD#HOUSEKEEPING: xxx or AIMLCOE-HOUSEKEEPING: xxx

## Pull Requests

* Name **SHOULD** follow code commit standards
* aka "AB#427: AWS Sam Testing" not "AWS Thing" or "quick pr"
* **SHOULD** only be merged once all code review points addressed

## Git Ignore

* All repos **MUST** have a git ignore
* Compiled reports and other assets such as executables **MUST** be git ignored
* *exception: where the point of the repo is to host said exe or reports*
* Gitignore **SHOULD** be set to ignore all developer IDE files
* Gitignore **SHOULD** be set to ignore all os specific files (aka .thumbdb of mac os)
* Gitignore **SHOULD** be set to ignore all development temp files or local config files
* Developers **SHOULD** not override gitignore to commit any of the above files
* If any of the above files are accidentally committed, they **SHOULD** be immediately deleting and any contained credentials committed immediately cycled

## Git Commit Hooks

* All code repos **SHOULD** use git commit hooks to run unit testing and linting tools to help ensure poor quality code is not committed
* *notes: 'help ensure' as obviously hooks can be avoided, and local developer setup / uncommitted files can cause issues even with hooks in place*

## Git Storage Standards

* developers **MUST** not commit passwords or credentials into git
* *even if they're only for dev environments...*
* developers **SHOULD** not commit large blobs of data such as video or raw media assets file (aka premier or lightwave files)
* *media asset files should be stored else, git is for code or small compiled assets such as gif/jpegs*
* *large blob files can break git performance and waste alot of space if frequently updated*
* all commits **MUST** be professional, nothing should be committed that breaches the company code of conduct
* *no name calling or personal blaming comments, hateful variable naming, etc*

*Exceptions*

* Developers **MAY** commit large final media assets if they are essential to the purpose of a application, aka sound effect mp3s for button presses but do so with caution, better storage solutions probably exist

## Git Maintenance Tips

Warning: These over lap above

* Avoid including files in source control that are specific to your development machine or process.
* *aka .vscode ide files, or mac os thumb file, use .gitignore to hide these*
* *some excellent gitignore examples* [*https://github.com/github/gitignore*](https://github.com/github/gitignore)
* Delete local and remote feature branches after merging
* Perform work in a feature branch
* When working in a feature branch, rebase frequently to incorporate upstream changes
* Use a pull request for code reviews
* Avoid committing large files into git

*based of* [*https://github.com/thoughtbot/guides/tree/main/git*](https://github.com/thoughtbot/guides/tree/main/git)

# Further Reading

* <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>
* <https://git-scm.com/book/en/v2>

## Git popular integrations reading

<https://azure.microsoft.com/en-gb/blog/linking-your-github-commits-with-azure-boards/>

<https://bigbrassband.com/git-integration-for-jira/documentation/linking-git-commits-to-jira-issues.html>