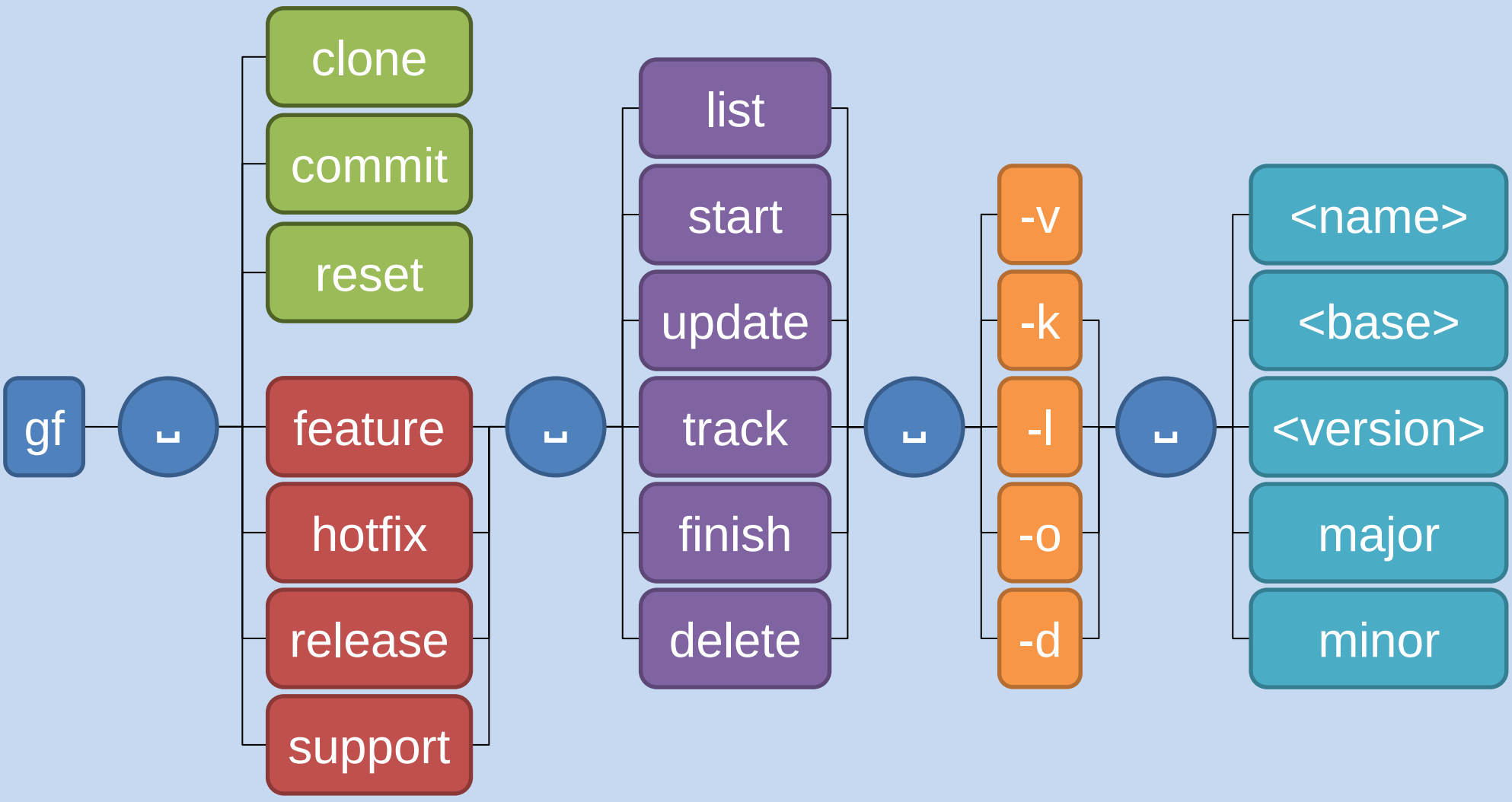
ITM Gitflow Script Package

Cheat Sheet

Overview of Gitflow Commands



➢ Clone Subcommand (clone + init + chmod)

gf clone <reference repository> <destination directory>

Clones the specified reference repository into the provided destination directory, e.g. <destination directory> = ".", initializes the ITM Gitflow script package for the cloned repository and modifies the access permissions of the target directory in such a way that only the owner of the repository is allowed to read, write and execute all content.

➢ Commit Subcommand (commit + push)

gf commit [flags <arguments>]

Basically, the "gf commit" command is a wrapper script combining a commit to the local repository with one to the tracked remote repository. For this, the standard "git commit" command is employed and, therefore, "gf commit" accepts the same flags and arguments as the "git commit" command does.

➢ Reset Subcommand (reset + push)

gf reset [flags <arguments>]

Basically, the "gf reset" command is a wrapper script combining a reset of the local repository with one of the tracked remote repository. For this, the standard "git reset" command is employed and, therefore, "gf reset" accepts the same flags and arguments as the "git reset" command does.

ITM Gitflow Script Package

Cheat Sheet

➢ Feature Branches

gf feature <action> [flags] [<name> [<base>]]

- **list**: List all existing local and remote feature branches.
  -v: Show some more information on the local feature branches.
- **start**: Create a new local feature branch and the corresponding remote branch.
  name: Name of the feature branch to be created.
  base: Base of the feature branch to be created (default = develop branch).
- **update**: Update an existing feature branch by merging another branch into it.
  name: Name of the feature branch to be updated.
  base: Name of the branch that will be merged into the specified feature branch (default = develop branch).
- **track**: Track an existing remote feature branch by creating a local copy of it.
  name: Name of the remote feature branch to be tracked.
- **finish**: Finish a feature branch, i.e., merge it into the develop branch and delete both the local and the remote feature branch afterwards.
  -k: Keep both the local and the remote feature branch after having merged them into the develop branch.
  name: Name of the feature branch to be finished.
- **delete**: Delete the local and the remote version of a feature branch without having merged them into the develop branch.
  -l: Keep the remote feature branch and only delete the corresponding local branch.
  name: Name of the feature branch to be deleted.

➢ Hotfix Branches

gf hotfix <action> [flags] [<argument>]

- **list**: List all existing local and remote hotfix branches.
  -v: Show some more information on the local hotfix branches.
- **start**: Create a new local hotfix branch and the corresponding remote branch. The new hotfix branch will be named automatically by the ITM Gitflow script based on the already assigned hotfix tags.
  version: Version tag of the released version to be fixed.
- **track**: Track an existing remote hotfix branch by creating a local copy of it.
  name: Name of the remote hotfix branch to be tracked.
- **finish**: Finish a hotfix branch, i.e., label it with the appropriate version tag, merge it into the develop and, if available, into a release branch and delete both the local and the remote hotfix branch afterwards.
  -k: Keep both the local and the remote hotfix branch after having merged and tagged them.
  -o: Only tag the hotfix branch without merging it into any other branch.
  -d: Merge the hotfix branch only into the develop branch (requires flag “-o” absent).
  name: Name of the hotfix branch to be finished.
- **delete**: Delete the local and the remote version of a hotfix branch without having merged them into any other branch.
  -l: Keep the remote hotfix branch and only delete the corresponding local branch.
  name: Name of the hotfix branch to be deleted.

ITM Gitflow Script Package

Cheat Sheet

➢ Release Branches

gf release <action> [flags] [<argument>]

- **list**: List all existing local and remote release branches.
  - **-v**: Show some more information on the local release branches.
- **start**: Create a new local release branch and the corresponding remote branch. The new release branch will be named automatically by the ITM Gitflow script based on the already assigned release version tags.
  - **major**: Create a release branch in order to prepare a major release, i.e. an upgrade (the first number of the version tag is incremented).
  - **minor**: Create a release branch in order to prepare a minor release, i.e. an update (the second number of the version tag is incremented).
- **track**: Track an existing remote release branch by creating a local copy of it.
  - **name**: Name of the remote release branch to be tracked.
- **finish**: Finish a release branch, i.e., label it with the appropriate version tag, merge it into the develop and the deploy branch and delete both the local and the remote release branch afterwards.
  - **-k**: Keep both the local and the remote release branch after having merged and tagged them.
  - **-o**: Merge the release branch only into the deploy branch.
  - **name**: Name of the release branch to be finished.
- **delete**: Delete the local and the remote version of a release branch without having merged them into any other branch.
  - **-l**: Keep the remote release branch and only delete the corresponding local branch.
  - **name**: Name of the release branch to be deleted.

➢ Support Branches

gf support <action> [flags] [<name> [<base>]]

- **list**: List all existing local and remote support branches.
  - **-v**: Show some more information on the local support branches.
- **start**: Create a new local support branch and the corresponding remote branch.
  - **name**: Name of the support branch to be created.
  - **base**: Base of the support branch to be created (a branch name or a version tag).
- **delete**: Delete the local and the remote version of a support branch without having merged them into any other branch.
  - **-l**: Keep the remote support branch and only delete the corresponding local branch.
  - **name**: Name of the support branch to be deleted.

ITM Gitflow Script Package

Additional Information

➢ Git branching model and ITM Gitflow script package

In January 2010, Vincent Driessen published an article called “A successful Git branching model”, which can be found under the following URL: http://nvie.com/posts/a-successful-git-branching-model. In this article, he presents a Git branching model developed by him, leading the way to a reliable and standardized, but also quite complex, workflow when using the Git versioning tool. A short summary on the basic concept is available here: http://blog.sourcetreeapp.com/2012/08/01/smart-branching-with-sourcetree-and-git-flow. When introducing the Git branching model at the ITM, following the intention to manage the Pasimodo source code according to the mentioned concept, the original model experienced some modifications in order to meet the special requirements demanded by the Pasimodo development process. The most important adaptions are:

- the branch providing the latest Pasimodo release versions is called “deploy”,
- another type of branches, the so-called “support branch”, is available for all other purposes,
- only one hotfix branch and only one release branch can be existent at the same time,
- hotfix branches will not be back-merged into the deploy branch, but kept as dangling tags(/branches),
- when using Gitflow specific commands always a push to the remote repository is performed and
- the version tag of a Pasimodo release consists of three parts: <Major tag>.<Minor tag>.<Hotfix tag>.

To gain the advantages provided by Vincent Driessen’s development model, it is necessary to observe all the rules established by its extensive guideline framework. In order to make the usage of the model much simpler, reduce the required maintenance effort and get new developers up to speed more quickly, the so-called ITM Gitflow script package, based on the Git-Flow command collection (https://github.com/nvie/gitflow), is introduced providing high-level repository operations for the presented Git branching model.

➢ Getting started with the ITM Gitflow script package

1) Read the article “A successful Git branching model” (http://nvie.com/posts/a-successful-git-branching-model) published by Vincent Driessen carefully and become familiar with the original Git branching model.
2) Take note of the modifications on the original Git branching model listed above.
3) Carefully read through the provided ITM Gitflow script package cheat sheet.
4) Create a local copy of the desired remote repository using the “gf clone” subcommand.
5) Start enjoying the flexibility and simplicity of usage provided by the ITM Gitflow script package.

➢ What to do when something goes wrong

1) Tell the other developers to stop interacting with the affected Git repository and outline your problem.
2) Record the current status of the inoperative repository and write a brief description of the problem.
3) Try to fix the problem by using Git standard commands and the ones introduced with the script package.
4) If it’s not working, kindly ask an admin to replace the current version of the repository with a backup.
5) Ask the person responsible for the ITM Gitflow script package even more kindly to solve the problem.
6) Go on to use the Gitflow package and don’t be afraid that something could go wrong again in the future.

➢ Some more information

- There are specific Gitflow “commit” and “reset” subcommands available, which perform an additional push at the end of the commit or reset process in comparison to the according standard Git commands. It could be a good idea to use both the specific and the standard subcommand depending on the situation.
- For all other actions except cloning and resetting a repository as well as committing changes make use of the standard Git subcommands, e.g. “git add”, “git move”, “git pull” and “git checkout”.