Deal with (g)it

Top 5 tips to make the best out of (g)it

#1 Sign your commits

Make visible you are the real owner of a commit.

Are these commit from Linus?

- this is a real github repo
- these are real commits

GitHub: Amog-OS/AmogOS

Fellow linux redditors may know this already

3

Anyone can be Linus!

- git config is not verifying anything

So you can set up your local config ...

```
git config --global user.name "torvalds"
git config --global user.email "torvalds@osdl.org"
```

... and be Linus:

```
touch proof
git stage proof
git commit -m "I am Linus!"
```

... at least thats what its going to look like

Verified commits to the rescue!

- signed with GPG

- visible on all git hosting platforms

- checkable with the git cli

GPG stands for GNU Privacy Guard and is a free tool to sign and encrypt things

5

Setting it up is easy - 3 steps to victory

1. create a GPG key (Tutorial)

2. add the key to your user profile on the git hosting platform (e.g. GitHub)

3. configure your local git installation:

```
# set gpg key
git config --global user.signingkey <key-id>

# autosign commits
git config --global commit.gpgsign true
```

Alternative: Use GPG Key Creation Assitant - created by me

#2 Keep your history clean

accountants don't use erasers or they end up in jail

But luckily we are not accountants!
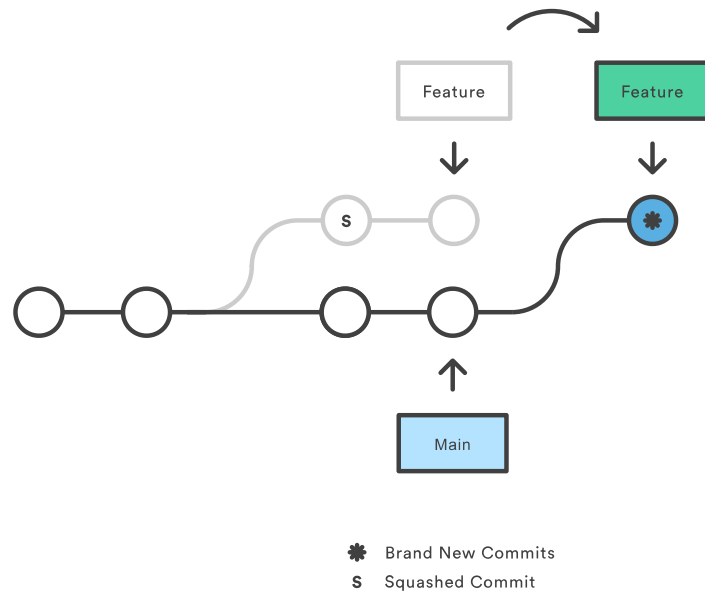
Squash your feature branches

- powerful feature allowing you to combine multiple commits

- supported by git hosters such as GitHub/GitLab on PRs

Image for git squash: atlassian.com

Feature   **Feature**

Main

❋ Brand New Commits
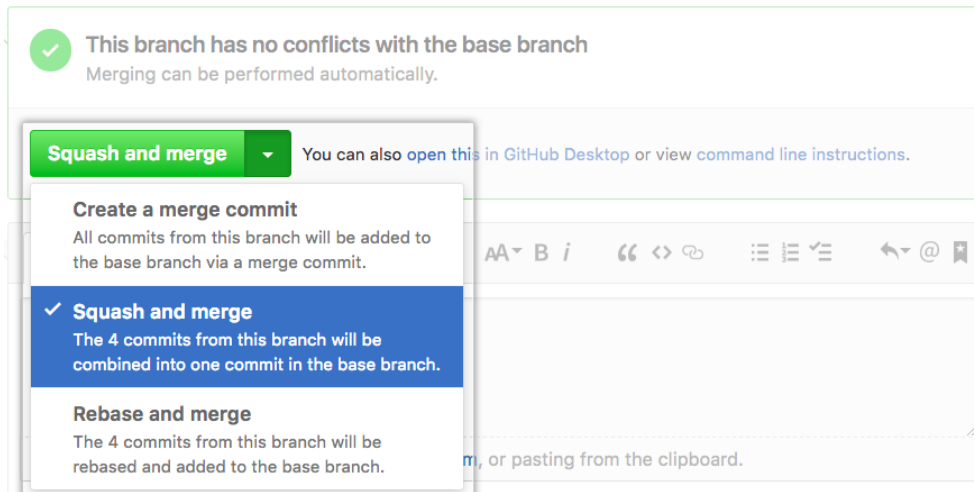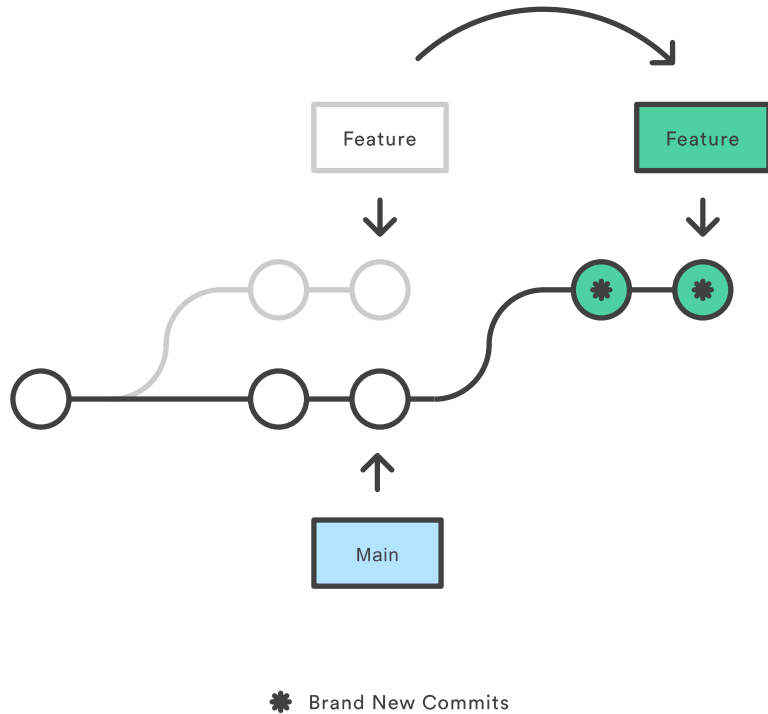
Rebase your changes

- local repo is not up to date
- main branch has evolved faster than feature branch
- smaller changes most of the time don't need to be merged

#3 Simplify your daily life

Git and your shell has more to offer than just builtin commands!

Add helper tools

- any command thats prefixed with `git-` can be a git command
- git-semver-tag becomes `git semver-tag`

Create alias in your git config

- entire shell command

- another subcommand

e.g. here are mine:

```
[alias]
p = pull
squash-all = "!f(){ git reset $(git commit-tree HEAD^{tree} -m \"${1:-A new start}\");};f"
sync = "!f() { git pull --rebase && git push; };f"
```

Alias entire git commands

- for the extreme lazy

- add to your shellrc

```
# use gpush
alias gpush = "git push"

# use commit "my message"
alias commit = "git stage . && git commit -m"
```

#4 Use branches and tags wisely

Nothing is immutable by default.

Choose the right branching concept

- trunk based development is a valid choice

- build your workflow around the real world, not some theory

- commiting to the main branch is not (always) a sin

- be creative and think out of the box

- don't see branching concepts as a religion

Tags are not immutable

- tags can be changed and deleted, making them also work as pointers

- that can break dependency pins

- only unique thing in git are commit hashes

#5 Rely on conventions

Who doesn't love conventions?

Commit with respect to conventional commits
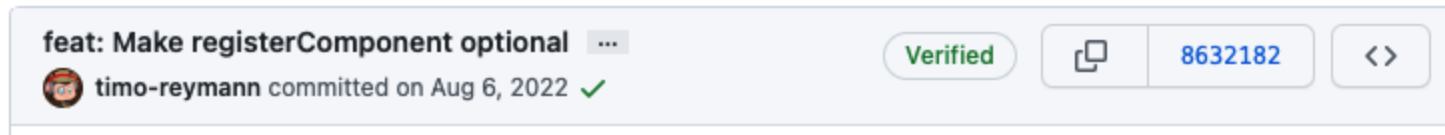
- be specific about your impact

- short and concise description

- explain details in the body

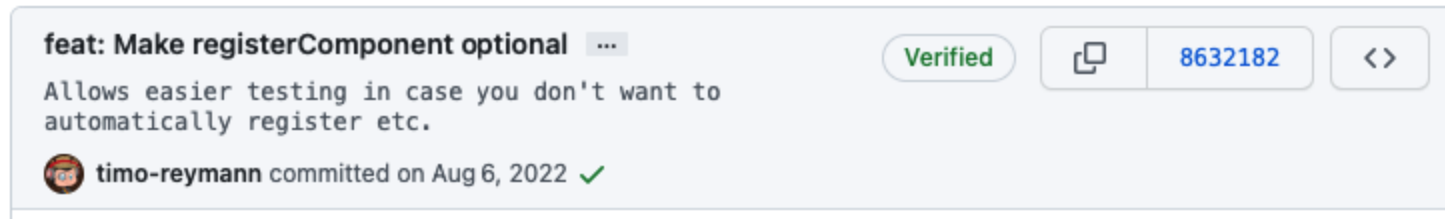The format is super simple:

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

[optional body]

[optional footer(s)]
```

Use the commit body

- the commit can have a body, use it when required

- collapses by default in web UIs and toolings

  - when viewing the history:

  - after clicking on the dots:

Use Conventional Comments on PRs

- no more undertone in your comments

- clear scope and less words to type and read

- take time for praise as well

The format, again, is super simple:

```
<label> [(decorations)]: <subject>

[discussion]
```

Recap

- sign your commits with GPG

- keep the history clean, dont hesitate to use erasers

- make your daily life easier with git and shell alias as well as third party packages and scripts

- be aware tags and branches are mutable all the way, only reliable thing are commit hashes

- use conventions for commits and PRs, don't forget about the message body

Q&A

Questions, concerns, ideas? - Now is the time

That's (g)it

Slides (Source):

github/timo-reymann/slides-deal-with-git

Slides (HTML):

deal-with-git.slides.timo-reymann.de

Twitter: @timo_reymann

GitHub: timo-reymann

In case of fire

1. git commit
2. git push
3. leave building