Tips and Tricks

Useful GIT-Tools and Commands

Useful Git-Tools

- gitk
- tig:

Path on mistral: /sw/rhel6-x64/vcs/tig-2.2.1/bin/tig

```
Path on workstations: /usr/bin/tig
short blog entry: https://www.atlassian.com/blog/git/git-tig
cheat sheet: http://ricostacruz.com/cheatsheets/tig.html
manual: http://jonas.nitro.dk/tig/manual.html
(Thanks to Ralf Mueller)
```

Useful Git Commands

- Checkout a version with a special hash tag:

git checkout 89aa38fc

You may have to “git submodule update” again. Looking at the status the first line of the message will be

```
HEAD detached at 89aa38f
```

This, of course, only works for hash tag inside your cloned repository, e.g. icon-aes.git.

- Checkout a new branch and branch off from a special hash tag:

git checkout -b my_new_branch 89aa38fc

- Delete a (local) branch

git branch -d my_new_branch

- Cherrypicking a range of commits

git cherry-pick 3020e317..9b9d68c8 origin/icon-aes-other-branch

Extend your prompt with the current branch

If the directory does not exist

```
mkdir -p $HOME/.config/git
```

change to

```
cd $HOME/.config/git
```

and copy

```
wget
https://raw.githubusercontent.com/git/git/master/contrib/completion/git-prom
pt.sh
```

edit `$HOME/.bashrc`; add

```
if [[ -f $HOME/.config/git/git-prompt.sh ]]
then
    source $HOME/.config/git/git-prompt.sh
    export -f __git_ps1
fi
export PS1='\[\e[32m\]\u@\h$(__git_ps1 "\[\e[36m\](%s)\[\e[0m\]")%\[\e[38m\]
\[\e[0m\]'
```

gives a prompt of the following form inside of a git repository

```
userid@hostname(git branch)%
```

and

```
userid@hostname%
```

else. (Hint: Try zsh with a good preconfigured .zshrc like `wget -O .zshrc` http://git.grml.org/f/grml-etc-core/etc/zsh/zshrc . Gives you tab completion for most git operations, remote (scp) autocompletion, and a lot more.)

*Copy of Notes on GIT

Add the current repository- and branch name to the commit message automatically

This needs to be applied by each developer. The tool of choice are the Git hooks. Hooks are scripts executed at a certain point of the Git workflow. They can be written in any language your shell supports. In this case we are interested in executing a script on the client side before we commit. We have in our repository's .git/hooks folder a set of examples, which you should ignore! Copy the following script

```
#! /bin/bash

# Just do nothing if there is a header already
expr "`head -1 $1`" : '[[:space:]]*\(\[[^]]*\]\)' > /dev/null && exit
```

```
# Prepend header with repo and branch name to current message
branch_name=$(git symbolic-ref --short HEAD)
branch_name="${branch_name##*/}"
repo_name=$(git config --get remote.origin.url | perl -pe
's/.*:(.*)\.git/$1/')
repo_name="${repo_name##*/}"
header=$(echo "$repo_name $branch_name" | awk '{ printf "[%s:%s]", $1, $2
}')
echo "$header $(cat $1)" > $1
```

into the .git/hooks folder and call the file `prepare-commit-msg`. Make it executable:

```
chmod +x .git/hooks/prepare-commit-msg
```

Every commit will automatically add

```
[your_origin repository name:your branch name]
```

eg.

```
[icon-cimd:icon-cimd-tcmalloc]
```

to the commit message.

In case you have several cloned repositories, you may move the script to a common directory to hold the hooks for all your repositories:

```
mkdir -p $HOME/.config/git/hooks
mv .git/hooks/prepare-commit-msg $HOME/.config/git/hooks
git config --global core.hooksPath $HOME/.config/git/hooks
```

If you do *not* want a common hook directory, you will have to add the script to every `.git/hooks` directory. In this case it still helps to copy the file prepare-commit-msg to a central place and simply copy it to the different clones.

Note that with the common directory in `core.hooksPath`, any remaining `.git/hooks` within your repositories will be ignored. If you need special hooks for some repository, you may add a fallback to `.git/hooks` by adding the following lines between the first and the second line of the common hook script:

```
# Check for local hook
hook_name=$(basename $0)
local_hook_name=$PWD/.git/hooks/$hook_name
[ -x $local_hook_name ] && exec $local_hook_name "$@"
```

*Copy of Cookbook

Seven rules of a great commit message

1. separate subject from body with a blank line,
2. limit the subject line to 50 characters,
3. capitalize the subject line,
4. do not end the subject line with a period,
5. use the imperative mood in the subject line,
6. wrap the body at 72 characters, and
7. use the body to explain what and why versus how.

*Copy of Notes on GIT

— *Monika Esch 2020/10/26 16:32 — Karl-Hermann Wieners 2022/04/20 13:00*