

# gitops-Training

## Agenda

### 1. Installation (GIT)

- [GIT auf Ubuntu/Debian installieren](#)
- [GIT unter Windows installieren](#)

### 2. Kubernetes

- [Installation micro8ks \(Ubuntu\)](#)

### 3. Commands (with tips & tricks)

- [git add + Tipps & Tricks](#)
- [git commit](#)
- [git log](#)
- [git config](#)
- [git show](#)
- [Needed commands for starters](#)
- [git branch](#)
- [git checkout - used for branches and files](#)
- [git merge](#)
- [git tag](#)

### 4. Advanced Commands

- [git reflog](#)
- [git reset - Back in Time](#)

### 5. github

- [GitHub Actions](#)
- [Github Pages](#)

### 6. Nix kaputtmachen - so gehts

- [Die 5 goldenenen Regeln](#)

### 7. Tips & tricks

- [Beautified log](#)
- [Change already committed files and message](#)
- [Best practice - Delete origin,tracking and local branch after pull request/merge request](#)
- [Change language to german - Linux](#)
- [Reference tree without sha-1](#)

### 8. Exercises

- [merge feature/4712 - conflict](#)
- [merge request with bitbucket](#)

### 9. Snippets

- [publish lokal repo to server - bitbucket](#)
- [failure-on-push-fix](#)
- [failure-on-push-with-conflict](#)

## 10. Extras

- [Best practices](#)
- [Using a mergetool to solve conflicts](#)

## 11. Help

- [Help from commandline](#)

## 12. Documentation

- [GIT Pdf](#)
- [GIT Book EN](#)
- [GIT Book DE](#)
- [Third Party Tools](#)

## Installation (GIT)

### GIT auf Ubuntu/Debian installieren

#### Installation

```
sudo apt update  
sudo apt install git
```

#### Language to english please !!

```
sudo update-locale LANG=en_US.UTF-8  
su - kurs  
  
## back to german  
  
sudo update-locale LANG=de_DE.UTF-8  
su - kurs  
  
## Reference:  
https://www.thomas-krenn.com/de/wiki/Locales\_unter\_Ubuntu\_konfigurieren  
  
## update-locale does a change in  
$ cat /etc/default/locale  
LANG=en_US.UTF-8
```

## **GIT unter Windows installieren**

- <https://git-scm.com/download/win>

## **Kubernetes**

### **Installation micro8ks (Ubuntu)**

#### **Reference:**

- <https://ubuntu.com/tutorials/install-a-local-kubernetes-with-microk8s#2-deploying-microk8s>

## Commands (with tipps & tricks)

### git add + Tipps & Tricks

#### Trick with -A

```
## only adds from the folder you are in recursively
## but not above (you might miss some files, when you are in a subfolder
git add .

### Fix -A
## adds everything no matter in which folder you are in your project
git add -A
```

## **git commit**

### **commit with multiple lines on commandline (without editor)**

```
git commit -am "New entry in todo.txt

* nonsense commit-message because of missing text-expertise"
## enter on last line
```

### **Change last commit-mesage (description)**

```
git commit --amend
## now you can change the description, but you will get a new commit-id
```

## git log

### Show last x entries

```
##  
## git log -x  
## Example: show last 2 entries  
git log -2
```

### Show all branches

```
git log --all  
## oder wenn alias alias.lg besteht:  
## git lg --all
```

### Show first log entry

```
## Step 1 - log needs to only show one line per commit  
git log --oneline --reverse  
  
## Step 2: combine with head  
git log --oneline --reverse | head -1
```

### Multiple commands with an alias

```
git config --global alias.sl '!git log --oneline -2 && git status'
```

## git config

### How to delete an entry from config

```
## Important: Find exact level, where it was added --global, --system, --local
## test before
## should contain this entry
git config --global --list

git config --unset --global alias.log
```



## **git show**

**Show information about an object e.g. commit**

```
git show <commit-ish>  
## example with commit-id  
git show 342a
```

## Needed commands for starters

```
git add -A
git status
git log // git log -4 // or beautified version if setup as alias git lg
git commit -am "commit message" // "commit message" can be freely chosen
## for more merge conflict resolution use only
git commit # to not change commit - message: must be message with merge
## the first time
git push -u origin master
## after that
git push
git pull
```

## **git branch**

### **Create branch based on commit (also past commit)**

```
git branch lookaround 5f10ca
```

### **Delete unmerged branch**

```
git branch -d branchname # does not work in this case  
git branch -D branchname # <- is the solution
```

## **git checkout - used for branches and files**

### **Checkout (change to) existing branch**

```
git checkout feature/4711
```

### **Checkout and create branch**

```
## Only possible once  
git checkout -b feature/4712
```

### **Recover deleted file**

```
rm todo.txt  
## get from last from last commit  
git checkout HEAD -- todo.txt
```

## **git merge**

### **Merge without conflict with fast-forward**

```
## Disadvantage: No proper history, because only one branch visible in log
## after fast-forward - merge

## Important that no changes are in master right before merging
git checkout master
git merge feature/4711
```

### **Merge (3-way) also on none-conflict (no conflicts present)**

```
git merge --no-ff feature/4711
```

## git tag

```
## set tag on current commit -> HEAD of branch
git tag -a v1.0 -m "my message for tag"
## publish
git push --tags

## set on specific commit
git tag -a v0.1 -m "Initial Release" a23c

## checkout files of a specific tag
git checkout v0.1
## or
git checkout tags/v0.1
```

## Advanced Commands

### git reflog

#### command

- show everything you (last 30 days), also stuff that is not visible in branch anymore

#### Example

```
git reflog
```

**when many entries a pager like less (aka man less) will be used**

```
## you can get out of the page with pressing the key 'q'
```

## **git reset - Back in Time**

### **Why ?**

- Back in time -> reset
- e.g. `git reset --hard e2d5`
- attention: only use it, when changes are not published (remotely) yet.
- → It is your command, IN CASE your are telling yourself, omg, what's that, what did i do here, let me undo that

### **Example**

```
git reset --hard 2343
```



# github

## GitHub Actions

### What are actions ?

Actions are individual tasks that you can combine to create jobs and customize your workflow.

You can create your own actions, or use and customize actions shared by the GitHub community.

Ref: <https://docs.github.com/en/actions/creating-actions/about-actions>

### Walkthrough to create a simple script based on Docker

```
## Step 1: Create private repo on github
```

```
## Step 2: Clone repo from github
```

```
## Step 3: Create Docker file
```

```
## Dockerfile
```

### Reference:

- <https://docs.github.com/en/actions/creating-actions/creating-a-docker-container-action>

## Github Pages

### Types of Pages

- Personal Page: <http://jmetzger.github.io>
- Project Page <http://>

### Personal Site

```
## Step 1: create personal repo
e.g.
https://github.com/gittrainereu/gittrainereu.github.io

git clone https://github.com/gittrainereu/gittrainereu.github.io
cd gittrainereu.github.io
echo "Hello World" > index.html
git add -A
git commit -m "Initial commit"
git push -u origin master

https://gittrainereu.github.io
```

### Project Page

# Nix kaputtmachen - so gehts

## Die 5 goldenenen Regeln

1. Kein `git commit --amend` auf bereits veröffentlicht (gepushed) commit.
2. Kein `git reset` vor bereits veröffentlichte (gepushed) commits  
(1234 (HEAD -letzter Commit) < 5412 (vö - HEAD~1 - vorletzte Commit ) -> kein reset auf 1234)
3. Mach niemals ein `git push --force` (JM sagt)
4. Kein Rebase auf bereits veröffentlichte commits (nach vö von Feature branchen)  
- ausser Feature-Branch kann online gelöscht und nochmal erstellt werden

## Tips & tricks

### Beautified log

### Walkthrough

```
git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset \
-%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset'"
```

### PRETTY FORMATS

- all documented in git help log (section PRETTY FORMAT)
- <https://git-scm.com/docs/git-log>

## Change already committed files and message

```
## Walkthrough
touch newfile.txt
git add .
git commit -am "new file added"

## Ups forgotten README
touch README
git add .
git commit --amend # README will be in same commit as newfile.txt
## + you can also changed the commit message
```

## Best practice - Delete origin,tracking and local branch after pull request/merge request

```
## After a succesful merge or pull request und gitlab / github
## Follow these steps for a succesful cleanup

## 1. Delete feature branch in web interface (e.g. gitlab / github)
## e.g. feature/4811

## 2. Locally on your system prune the remote tracking branch
git fetch --prune

## 3. Switch to master or main (depending on what you master branch is)
git checkout master

## 4. Delete local branch
git branch -d feature/4811
```

## Change language to german - Linux

```
sudo update-locale LANG=en_US.UTF-8
su - kurs

## back to german

sudo update-locale LANG=de_DE.UTF-8
su - kurs

## Reference:
https://www.thomas-krenn.com/de/wiki/Locales\_unter\_Ubuntu\_konfigurieren

## update-locale does a change in
$ cat /etc/default/locale
LANG=en_US.UTF-8
```

## Reference tree without sha-1

### Exercises

#### merge feature/4712 - conflict

##### Exercise

1. You are in master-branch
2. Checkout new branch feature/4712
3. Change line1 in todo.txt
4. `git add -A; git commit -am "feature-4712 done"`
5. Change to master
6. Change line1 in todo.txt
7. `git add -A; git commit -am "change line1 in todo.txt in master"`
8. `git merge feature/4712`



## merge request with bitbucket

```
## Local
git checkout -b feature/4822
ls -la
touch f1.txt
git add .
git commit -am "f1.txt"
touch f2.txt
git add .
git commit -am "f2.txt"
git push origin feature/4822
```

## Online bitbucket

```
## create merge request
## and merge
```

## Delete branch online after merge

### Cleanup locally

```
git fetch --prune
git checkout master
git branch -D feature/4822
git pull --rebase
```

## Snippets

### publish lokal repo to server - bitbucket

```
# Step 1: Create repo on server without README and .gitignore /set both to NO when
creating

# Step 2: on commandline locally
cd /path/to/repo
git remote add origin https://erding2017@bitbucket.org/erding2017/git-remote-
jochen.git
git push -u origin master

# Step 3: for further commits
echo "test" > testdatei
git add .
git commit -am "added testdatei"
git push
```

## failure-on-push-fix

```
## Step 1: push produces error
## you have done git push -u origin master the last to setup remote tracking branch by
option -u
git push
Password for 'https://erding2017@bitbucket.org':
To https://bitbucket.org/erding2017/git-remote-jochen.git
 ! [rejected] (fetch first)
error: failed to push some refs to 'https://erding2017@bitbucket.org/erding2017/git-
remote-jochen.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
## Step 2: Integrate changes from online
git pull
## Step 2a: Editor opens and you need to save and ext (without changing anything)

## Step 3: re-push
git push
```

## failure-on-push-with-conflict

### Failure push

```
## Step 1: push produces error
## you have done git push -u origin master the last to setup remote tracking branch by
option -u
git push
Password for 'https://erding2017@bitbucket.org':
To https://bitbucket.org/erding2017/git-remote-jochen.git
! [rejected] (fetch first)
....
## Step 2: Integrate changes from online
git pull

## Step 3: Solve conflict
Auto-merging agenda.txt
CONFLICT (content): Merge conflict in agenda.txt
Automatic merge failed; fix conflicts and then commit the result.
kurs@ubuntu-tr01:~/training$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
    (use "git pull" to merge the remote branch into yours)

## Step 3a: Open file agenda.txt
## Decide for which version
## - remove all <<<<< and ===== and >>>>>>>>> - lines

## Step 3b: then: save + exit from editor

## Step 3c: mark resolution
git status
git add todo.txt

## Step 3d:
git status
## as written there
git commit

## Step 4: re-push
git push
```

### recipe

```
git push # failure
git pull
git add todo.txt
```

```
git commit  
git push
```

## Extras

### Best practices

- Delete branches, not needed anymore
- `git merge --no-ff` -> for merging local branches (to get a good history from local)
- from online: `git pull --rebase` // clean history from online, not to many branches
- nur auf einem Arbeiten mit max. 2 Teilnehmern, wenn mehr feature-branch

### Teil 2:

- Be careful with git commands that change history.
  - never change commits, that have already been pushed
- Choose workflow wisely
- Avoid `git push -f` in any case // should not be possible
- Disable possibility to push -f for branch or event repo

## Using a mergetool to solve conflicts

### Meld (Windows)

- <https://meldmerge.org/>

### Configuration in Git for Windwos (git bash)

```
## you have to be in a git project
git config --global merge.tool meld
git config --global diff.tool meld
## Should be on Windows 10
git config --global mergetool.meld.path
"/c/Users/Admin/AppData/Local/Programs/Meld/Meld.exe"
## do not create an .orig - file before merge
git config --global mergetool.keepBackup false
```

### How to use it

```
## when you have conflict you can open the mergetool (graphical tool with )
git mergetool
```

# Help

## Help from commandline

### On Windows

```
## on git bash enter
git help <command>
## e.g.
git help log

## --> a webpage will open with content
```



## Documentation

### GIT Pdf

- <http://schulung.t3isp.de/documents/pdfs/git/git-training.pdf>

### GIT Book EN

- <https://git-scm.com/book/en/v2>

### GIT Book DE

- <https://git-scm.com/book/de/v2>

## Third Party Tools

### Continuous Integration / Continuous Deployment (CI/CD)

```
## Test often / Test automated (CI)

* Jenkins
* Github Actions
* Git Webhooks

## Publish new versions frequently (CD)

* Jenkins
* Github Action
* Git Webhooks
```