



Faculty of Computer Science, Institute of Systems Architecture, Chair of Systems Engineering

# RoboLab Autumn Course

## Git – Crash Course

by M.Sc. Samuel Knobloch

# Gliederung

- Git – A (very) brief introduction
- Basics
- Advanced
- Alternative: IDE

# Git – A (very) brief introduction

# Git – A (very) brief introduction

- Distributed Version Control System (DVCS)
  - Local and remote repositories are possible – can be independent from each other
- Can be used for individual work or team synchronisation
  - Parallel development of several versions of the same software
- Features: Stash, History, Rollback to old states
- Automated merging of different working states

# Basics

# Basics – git clone

- Clone repository from a server

→ <dir> is optional

```
$ git clone git@remote.url:repo/path <dir>
```

```
$ git clone https://remote.url/repo/path <dir>
```

- Expandable with flags

→ Important flag: --recursive

```
$ git clone --recursive git@remote.url:repo/path <dir>
```

## Basics (Example) – git clone

```
$ git clone git@se-gitlab.inf.tu-dresden.de:robolab/robolab-template.git robolab-  
template  
Cloning into 'robolab-template'...  
[...]  
Receiving objects: 100% (156/156), 30.89 KiB | 2.21 MiB/s, done.  
Resolving deltas: 100% (89/89), done.
```

```
$ git clone --recursive git@se-gitlab.inf.tu-dresden.de:robolab/robolab-template.git  
robolab-template  
Cloning into 'robolab-template'...  
remote: Enumerating objects: 26, done.  
[...]  
Receiving objects: 100% (156/156), 30.89 KiB | 930.00 KiB/s, done.  
Resolving deltas: 100% (89/89), done.  
Submodule 'robolab-deploy' (git@se-gitlab.inf.tu-dresden.de:robolab/robolab-  
deploy.git) registered for path 'robolab-deploy'  
Cloning into '/home/head sick/tmp/robolab-template/robolab-deploy'...  
[...]  
Receiving objects: 100% (66/66), 21.39 KiB | 2.38 MiB/s, done.  
Resolving deltas: 100% (21/21), done.  
Submodule path 'robolab-deploy': checked out  
'02e78bbf97591780493dad1b731a3085ec71c258'
```

# Basics - git init

- Create a new local repository
  - --initial-branch is optional

```
$ git init --initial-branch=master
```



## Basics (Example) - git init

```
$ git init --initial-branch=master
Initialized empty Git repository in robolab-intro/.git/
```

```
$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:     git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:     git branch -m <name>
Initialized empty Git repository in robolab-intro/.git/
```

# Basics – git add

- Add new files
  - There is no command line output
  - Result can be checked using git status

```
$ git add <file>  
$ git add <dir>/  
$ git add *.<type>  
$ git add .
```

# Basics – git status

- Check the state of your local files
  - What is already under version control
  - What is not under version control
  - What has been changed or deleted

```
$ git status
```

## Basics (Example) - git status

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  NEWFILE1.md
  NEWFILE2.md
  NEWFILE3.md
  NEWFILE4.md
  OTHER.txt
  subdir/

nothing added to commit but untracked files present (use "git add" to track)
```

## Basics (Example) – git add, git status

```
$ git add *.md

$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   NEWFILE1.md
    new file:   NEWFILE2.md
    new file:   NEWFILE3.md
    new file:   NEWFILE4.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    OTHER.txt
    subdir/
```

## Basics (Example) – git add, git status

```
$ git add .  
  
$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   NEWFILE1.md  
    new file:   NEWFILE2.md  
    new file:   NEWFILE3.md  
    new file:   NEWFILE4.md  
    new file:   OTHER.txt  
    new file:   subdir/SUBFILE1.md
```

# Basics - git commit

- Create a commit after adding files
  - Commit message is added to command

```
$ git commit <file> -m "<message">  
$ git commit <dir> -m "<message">  
  
$ git commit -a -m "<message">
```


- Commit message is added interactively in an editor

```
$ git commit -a
```

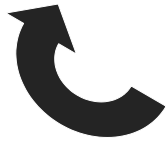
## Basics (Example) - git commit

```
$ git commit subdir/ -m "Added subdir"
[master (root-commit) 564828a] Added subdir
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 subdir/SUBFILE1.md
```

```
$ git commit -a
[master 9e2ec0f] Added all new Markdown and Text files
5 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 NEWFILE1.md
create mode 100644 NEWFILE2.md
create mode 100644 NEWFILE3.md
create mode 100644 NEWFILE4.md
create mode 100644 OTHER.txt
```



```
1 Added all new Markdown and Text files
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch master
6 # Changes to be committed:
7 #   new file:   NEWFILE1.md
8 #   new file:   NEWFILE2.md
9 #   new file:   NEWFILE3.md
10 #  new file:   NEWFILE4.md
11 #   new file:   OTHER.txt
12 #
```





# Basics – git remote

- Connect a local repository with a remote/server
  - Command does not create output

```
$ git remote add origin <url>
```

- Update remote url

```
$ git remote set-url origin <new_url>
```

- Show remote information

```
$ git remote -v
```

## Basics (Example) - git remote

```
$ git remote -v  
origin    git@gitlab.com:knobsam/robofab-intro.git (fetch)  
origin    git@gitlab.com:knobsam/robofab-intro.git (push)
```

# Basics – git push

- Synchronise remote with local state
  - Default branch is called master
  - When pushing the first time you have to give the remote branch to connect to

```
$ git push
```

```
$ git push -u origin <branch>
```

## Basics (Example) – git push

```
$ git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master
```

```
$ git push -u origin master
client_global_hostkeys_private_confirm: server gave bad signature for RSA key 0
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 553 bytes | 553.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To gitlab.com:knobsam/robolab-intro.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

# Basics – git pull

- Synchronise local with remote state
  - May create a merge under some circumstances

```
$ git pull
```

## Basics (Example) - git pull

```
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
From gitlab.com:knobsam/robofab-intro
 0b9b685..8331578 master    -> origin/master
Updating 0b9b685..8331578
Fast-forward
 OTHER.txt          | 0
 subdir/SUBFILE2.md | 0
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 OTHER.txt
create mode 100644 subdir/SUBFILE2.md
```

# Basics – git fetch

- Get informations from the remote
  - The flag -p synchronises the branches and removes links to already deleted server branches
  - No files will be downloaded

```
$ git fetch
```

```
$ git fetch -p
```

## Basics (Example) – git fetch

```
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
From gitlab.com:knobsam/robolab-intro
    0b9b685..8331578  master    -> origin/master
```

```
$ git fetch -p
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
From gitlab.com:knobsam/robolab-intro
    0b9b685..8331578  master    -> origin/master
- [deleted]         staging    -> origin/staging
```



# Basics – git branch

- Branching
  - Initially, the new branch is only created locally
  - The flag -a lists all local and remote branches that exists

```
$ git branch <new_branch>
```

```
$ git branch -a
```

- Deleting is also possible

```
$ git branch -D <branch>
```

## Basics (Example) – git branch

```
$ git branch testing
$ git branch -a
* master
testing
remotes/origin/master
```

```
$ git branch -D testing
Deleted branch testing (was 0b9b685).
$ git branch -a
* master
remotes/origin/master
```

# Basics – git checkout

- Checkout
  - Does only work on existing branches
  - The flag -b creates a new branch and switches to it

```
$ git checkout <existing_branch>
```

```
$ git checkout -b <new_branch>
```

## Basics (Example) – git checkout

```
$ git checkout staging  
error: pathspec 'staging' did not match any file(s) known to git
```

```
$ git checkout -b staging  
Switched to a new branch 'staging'
```

```
$ git branch -a  
master  
* staging  
testing  
remotes/origin/master
```

# Basics – git switch

- A (new) and fast way for creating and switching between branches
  - The flag -c creates a new branch and switches to it

```
$ git checkout <existing_branch>
```

```
$ git checkout -b <new_branch>
```

# Advanced

# Advanced – git log

- Let's have a look at the history
  - Command without any parameters is quite easy
  - Command with parameters is more clear and better understandable
  - Longer histories are opened inside an e.g. less environment (on Linux)

```
$ git log
```

```
$ git log --graph --decorate --all
```

# Advanced (Example) – git log

```
$ git log
commit bfafaca32c702cad4b4c07c830773b182ec61fc3 (HEAD -> staging, origin/staging)
Author: Samuel Knobloch <samuelknobloch@gmail.com>
Date: Thu Aug 19 10:40:21 2021 +0200

    Added Staging files

commit 0b9b685b160c45b7da3616ca001854c810235d9d (master)
Author: Samuel Knobloch <samuelknobloch@gmail.com>
Date: Thu Aug 19 10:13:43 2021 +0200

    Removed Text file

commit 9e2ec0fc81408a848c17dc18984aa790e3868139
Author: Samuel Knobloch <samuelknobloch@gmail.com>
Date: Thu Aug 19 09:58:22 2021 +0200

    Added all new Markdown and Text files

commit 564828a465fbdec078e873a93434b7f639e73ad4
Author: Samuel Knobloch <samuelknobloch@gmail.com>
Date: Thu Aug 19 09:57:40 2021 +0200

    Added subdir
```



# Advanced (Example) – git log

```
$ git log --graph --decorate --all
* commit bfafacea32c702cad4b4c07c830773b182ec61fc3 (HEAD -> staging, origin/staging)
| Author: Samuel Knobloch <samuelknobloch@gmail.com>
| Date: Thu Aug 19 10:40:21 2021 +0200
|
| Added Staging files
|
| * commit 8331578d2cfa1b95279993e7cdc2b6f7a91c156e (origin/master)
| / Author: Samuel Knobloch <samuelknobloch@gmail.com>
| Date: Thu Aug 19 10:15:54 2021 +0200
|
| Added some new files
|
| * commit 0b9b685b160c45b7da3616ca001854c810235d9d (master)
| Author: Samuel Knobloch <samuelknobloch@gmail.com>
| Date: Thu Aug 19 10:13:43 2021 +0200
|
| Removed Text file
|
| * commit 9e2ec0fc81408a848c17dc18984aa790e3868139
| Author: Samuel Knobloch <samuelknobloch@gmail.com>
| Date: Thu Aug 19 09:58:22 2021 +0200
|
| Added all new Markdown and Text files
|
| * commit 564828a465fbdec078e873a93434b7f639e73ad4
| Author: Samuel Knobloch <samuelknobloch@gmail.com>
| Date: Thu Aug 19 09:57:40 2021 +0200
|
| Added subdir
```

# Advanced – git merge

- Combine (merge) two branches
  - Target branch has to have the latest state (execute `git pull` if necessary)
  - Commit message will be created automatically

```
$ git checkout <first_branch>
$ git merge <other_branch>

$ git commit -a -m "<message>"

$ git push
```

## Advanced (Example) – git merge

```
$ git checkout master
Already on 'master'
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)
$ git pull
Updating 0b9b685..8331578
Fast-forward
 OTHER.txt          | 0
 subdir/SUBFILE2.md | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 OTHER.txt
 create mode 100644 subdir/SUBFILE2.md
$ git merge origin/staging
Merge made by the 'recursive' strategy.
 STAGING1.md | 0
 STAGING2.md | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 STAGING1.md
 create mode 100644 STAGING2.md
```

## Advanced (Example) – git merge

```
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 305 bytes | 305.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
To gitlab.com:knobsam/roboLab-intro.git
 8331578..cedeefc  master -> master
```

## Advanced (Example) – git merge

```
$ git push
To gitlab.com:knobsam/robolab-intro.git
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'gitlab.com:knobsam/robolab-intro.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

# Advanced - git merge, fast-forward

- Merging without creating commit messages: Fast Forward

```
$ git checkout <first_branch>  
$ git merge <other_branch> --ff  
  
$ git push
```

## Advanced (Example) – git merge, fast-forward

```
$ git checkout master
$ git merge origin/fast-forward --ff
Updating cedeefc..0b50aeb
Fast-forward
 STAGING1.md | 0
 STAGING2.md | 0
 2 files changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 STAGING1.md
 delete mode 100644 STAGING2.md
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To gitlab.com:knobsam/robofab-intro.git
 cedeefc..0b50aeb master -> master
```

# Advanced – git rebase

- Combine multiple commits into a single on
  - A reset is executed on the given base branch
  - When pushing, the flag --force is required in order to overwrite the branch history

```
$ git checkout <branch>
$ git rebase -i <origin_base_branch>

$ git push --force
```



## Advanced (Example) – git rebase

```
$ git log --graph --decorate --all
* commit 3e3fd66f4fca12ee09e093faae998c7a04f387a4 (HEAD -> rebase, origin/rebase)
 | Author: Samuel Knobloch <samuelknobloch@gmail.com>
 | Date: Thu Aug 19 11:15:32 2021 +0200
 |
 |     Added third file
 |
 * commit 607419e73bd7e5049999ac90ae9775f546c859c5
 | Author: Samuel Knobloch <samuelknobloch@gmail.com>
 | Date: Thu Aug 19 11:15:22 2021 +0200
 |
 |     Added second file
 |
 * commit 4c787615b03792f84463b3a2e491e589ab5c9d55
 | Author: Samuel Knobloch <samuelknobloch@gmail.com>
 | Date: Thu Aug 19 11:15:15 2021 +0200
 |
 |     Added first file
 |
 ~~~~~
```

## Advanced (Example) – git rebase

```
$ git checkout rebase
Branch 'rebase' set up to track remote branch 'rebase' from 'origin'.
Switched to a new branch 'rebase'
$ git rebase -i origin/master
[detached HEAD 1fcd1b3] Added first file Added second file Added third file
Date: Thu Aug 19 11:15:15 2021 +0200
3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 REBASE1.md
 create mode 100644 REBASE2.md
 create mode 100644 REBASE3.md
Successfully rebased and updated refs/heads/rebase.
$ git push --force
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 276 bytes | 276.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for rebase, visit:
remote:   https://gitlab.com/knobsam/robofab-intro/-/merge_requests/new?merge_request%5Bsource_branch%5D=rebase
remote:
To gitlab.com:knobsam/robofab-intro.git
+ 3e3fd66...1fcd1b3 rebase -> rebase (forced update)
```

## Advanced (Example) – git rebase

```
$ git log --graph --decorate --all
* commit 1fcd1b3a4b4b27cdef07a190a8df9f28f053d361 (HEAD -> rebase, origin/rebase)
 | Author: Samuel Knobloch <samuelknobloch@gmail.com>
 | Date: Thu Aug 19 11:15:15 2021 +0200
 |
 |     Added first file
 |     Added second file
 |     Added third file
 |
 |-----
```

# Advanced – git pull, rebase

- The better Pull command
  - No merge will be created when synchronising the local with the remote repository

```
$ git pull --rebase
```

## Advanced (Example) – git pull, rebase

```
$ git pull --rebase
From gitlab.com:knobsam/robolab-intro
   0b50aeb..1fcd1b3  master    -> origin/master
Already up to date.
```

# Alternatives: IDE

# Alternatives: IDE

- Modern IDEs do have a good to very good integration of versioning
  - Git, Mercurial, Subversion, CVS
- A lot of things are encapsulated, Workflows are not visible
  - Not always good for understanding what's going on
- Harder to deal with errors if you are a Git beginner (and you will have errors)