



AARHUS UNIVERSITET

# Software Engineering and Architecture

Git in Practice

- A highly personalized view, so take it as my opinion
- Git is *driving a Ferrari without a safety belt!*
  - There are one zillion handles to crank!
  - There are zillions of way to use Git efficiently
  - *There are zillions of way to get lost or mess your repo up completely*
  - **Keep it simple! You ain't gonna need it!**
- And
  - Beware of the good spirit fellow student that 'helps you' by issuing a few weird git command you do not understand!
    - And makes git behave weird for the rest of the course



# If Git F... up badly?

- I have more than once done the **'reboot'**
- **Delete** the local workspace
- Clone the repo anew
  - Much better than let a fellow Git-Wizzard issue ten weird commands, give up and walk away, leaving you with:
    - **Big ball of mud**

# I am not alone 😊





# The Core Workflow

- Git clone (your repo name)
  - Get your copy of the code base (Done **once!** Or if 'rebooting' 😊)
- Git add (file)
  - Add newly made files to the staging area/index (See below)
- Git commit `–a –m` "meaningful explanation of what you made"
  - "-a" auto adds all changes to *existing files in repo* the staging area / index
  - IntelliJ will help remember to add them to the index when created within it.



# The Core Workflow

- Git push
  - Copy all your commits to the team's remote repository
- Git pull
  - Get your team mates commits into your local repo
- Git status
  - See status of your local workspace
- Git log -3
  - See last 3 versions' log messages



- Git fetch origin
  - Get the branches overview from the origin + all newly made branches
- Git branch –a
  - Show all branches *including* all on the origin that you do not have currently
- Git checkout {branchname}
  - Checkout given branch, switch to it, **and begin tracking it**

- **Do not pollute your repo!**
  - There are ‘source’ artefacts and there are ‘derived’ artefacts
    - Source = manual hard intellectual work
      - Java source files, graphics files, sound clips, etc.
    - Derived = a tool produces it in milliseconds
      - .class files, .jar, JavaDoc, test output, coverage HTML, ...
- **.gitignore**
  - Put a file ‘.gitignore’ in your root
  - State all ‘derived’ artefacts in it (folders, file wildcards)
    - /build                    ignore all that gradle produces in the build folder
    - \*.iml                    ignore IntelliJ bla bla files



# My own workflow

- Overviewing branches without graphics is hard!
- I develop on Ubuntu
  - But makes most branching/merging in SourceTree on Windows



- But can be viewed in shell:

Date	Author	Commit
25 aug 2017 9:50	henrikbaerbak <ht>	905f436
14 aug 2017 11:08	henrikbaerbak <ht>	1dfebbd
10 jul 2017 15:30	HenrikBaerbak <hl>	ab82078
6 jul 2017 11:55	henrikbaerbak <ht>	b88d4ad
3 jul 2017 16:28	Henrik Baerbak <hl>	3aeb018
3 jul 2017 16:24	henrikbaerbak <ht>	814660f
23 jun 2017 9:47	henrikbaerbak <ht>	5dc36e6
23 jun 2017 9:04	henrikbaerbak <ht>	d8e3e36
22 jun 2017 16:17	henrikbaerbak <ht>	2cc7b2e
jun 2017 15:45	henrikbaerbak <ht>	8a83c9
jun 2017 11:22	henrikbaerbak <ht>	726dbcf
jun 2017 10:02	Henrik Baerbak <hl>	a24a8a
jun 2017 9:56	henrikbaerbak <ht>	526ee4c
jun 2017 9:06	henrikbaerbak <ht>	c56b4f
jun 2017 8:42	henrikbaerbak <ht>	20052df

```
csdev@m51:~/proj/cave$ git log --graph --simplify-by-decoration --all --oneline
* 090c012 (refs/stash)
* c486118 (origin/merge-create-generic-service, merge-create-generic-service) end of line
* 335e1f5 Merge branch 'issue-create-generic-service' into merge-create-generic-service
|
| * abd384a (HEAD -> issue-create-generic-service, origin/issue-create-generic-service) Test point: A generic service c
| creator is intro in objmgr+factory. Demo that it can be used for quote service as well. A full refactoring pending, chec
| g is still a big mess.
|
| * | 0b315c6 (tag: image_cave-jar-1.29, origin/f20-solution, f20-solution) Doc update with overview.
| * | 2bc2f6d (tag: image_cave-jar-1.28) minor
| * | a9134cc (tag: image_cave-jar-1.27) MileStone: CDT now working with test containers for RealCaveServiceConnector. Cf
|
| * | e504fba Merged code that reintroduces the PlayerNameService
|
|
|
| * c8d6506 (origin/dev, dev) Milestone: PlayerNameService reintroduced. All tests pass. Manual tests looks fine.
| * | cdafc3e (tag: image_cave-jar-1.25) Snapshot. Next Action require changes to the master branch!
```



# CheatSheet

AARHUS UNIVERSITET

- Git-tower.com

## CREATE

Clone an existing repository  
`$ git clone ssh://user@domain.com/repo.git`

Create a new local repository  
`$ git init`

## LOCAL CHANGES

Changed files in your working directory  
`$ git status`

Changes to tracked files  
`$ git diff`

Add all current changes to the next commit  
`$ git add .`

Add some changes in <file> to the next commit  
`$ git add -p <file>`

Commit all local changes in tracked files  
`$ git commit -a`

Commit previously staged changes  
`$ git commit`

Change the last commit  
*Don't amend published commits!*  
`$ git commit --amend`

## COMMIT HISTORY

Show all commits, starting with newest  
`$ git log`

Show changes over time for a specific file  
`$ git log -p <file>`

Who changed what and when in <file>  
`$ git blame <file>`

## BRANCHES & TAGS

List all existing branches  
`$ git branch -av`

Switch HEAD branch  
`$ git checkout <branch>`

Create a new branch based on your current HEAD  
`$ git branch <new-branch>`

Create a new tracking branch based on a remote branch  
`$ git checkout --track <remote/branch>`

Delete a local branch  
`$ git branch -d <branch>`

Mark the current commit with a tag  
`$ git tag <tag-name>`

## UPDATE & PUBLISH

List all currently configured remotes  
`$ git remote -v`

Show information about a remote  
`$ git remote show <remote>`

Add new remote repository, named <remote>  
`$ git remote add <shortname> <url>`

Download all changes from <remote>, but don't integrate into HEAD  
`$ git fetch <remote>`

Download changes and directly merge/integrate into HEAD  
`$ git pull <remote> <branch>`

Publish local changes on a remote  
`$ git push <remote> <branch>`

Delete a branch on the remote  
`$ git branch -dr <remote/branch>`

Publish your tags  
`$ git push --tags`

## MERGE & REBASE

Merge <branch> into your current HEAD  
`$ git merge <branch>`

Rebase your current HEAD onto <branch>  
*Don't rebase published commits!*  
`$ git rebase <branch>`

Abort a rebase  
`$ git rebase --abort`

Continue a rebase after resolving conflicts  
`$ git rebase --continue`

Use your configured merge tool to solve conflicts  
`$ git mergetool`

Use your editor to manually solve conflicts and (after resolving) mark file as resolved  
`$ git add <resolved-file>`  
`$ git rm <resolved-file>`

## UNDO

Discard all local changes in your working directory  
`$ git reset --hard HEAD`

Discard local changes in a specific file  
`$ git checkout HEAD <file>`

Revert a commit (by producing a new commit with contrary changes)  
`$ git revert <commit>`

Reset your HEAD pointer to a previous commit ...and discard all changes since then  
`$ git reset --hard <commit>`

...and preserve all changes as unstaged changes  
`$ git reset <commit>`

...and preserve uncommitted local changes  
`$ git reset --keep <commit>`



- Commit related changes
  - Fixing two bugs should lead to two commits
- Commit **often**
  - ‘Take small steps’, break big into small, one step at a time
  - Safe version to retract to in case of ‘Do Over’
- Push and pull **often**
  - Do not let team efforts drift apart!

- Use the commit log to express the goal achieved/contents of the commit

The screenshot shows the Git GUI interface with a commit log table and a detailed view of a specific commit.

Graph	Description	Date
●	Updated compose file and cpf for swarm usage	14 aug 2017 11:08
●	Merge branch 'itminds-course' into itminds-rancher	3 jul 2017 16:28
●	Fixed encoding issue, prevents using docker build, sigh	3 jul 2017 16:24
●	ip updated to running rancher	23 jun 2017 9:47
●	updated rancher to reflect port change in dockerfile	23 jun 2017 9:04
●	RELEASE version: Run load balanced daemons in a docker stack	22 jun 2017 16:17
●	added docker visualizer to compose	22 jun 2017 15:45
●	snapshot, testing swarm behaviour	22 jun 2017 11:22
●	Merge branch 'itminds-course' into itminds-rancher	22 jun 2017 10:02
●	Updated HTTP IPC layer to be verbose; updated daemon /info page to print	22 jun 2017 9:56
●	Doc updates, bit of refactoring of the CPF files	22 jun 2017 9:06
●	Doc	22 jun 2017 8:42
●	Diary updates of swarm /compose testing.	16 jun 2017 14:56
●	Bughunting updates to compose and cpf for swarm ops	16 jun 2017 14:46

  

Graph	Description	Date
●	Updated compose file and cpf for swarm usage	14 aug 2017 11:08
●	Merge branch 'itminds-course' into itminds-rancher	3 jul 2017 16:28
●	Fixed encoding issue, prevents using docker build, sigh	3 jul 2017 16:24
●	ip updated to running rancher	23 jun 2017 9:47
●	updated rancher to reflect port change in dockerfile	23 jun 2017 9:04
●	RELEASE version: Run load balanced daemons in a docker stack	22 jun 2017 16:17
●	added docker visualizer to compose	22 jun 2017 15:45
●	snapshot, testing swarm behaviour	22 jun 2017 11:22
●	Merge branch 'itminds-course' into itminds-rancher	22 jun 2017 10:02
●	Updated HTTP IPC layer to be verbose; updated daemon /info page to print	22 jun 2017 9:56
●	Doc updates, bit of refactoring of the CPF files	22 jun 2017 9:06
●	Doc	22 jun 2017 8:42
●	Diary updates of swarm /compose testing.	16 jun 2017 14:56
●	Bughunting updates to compose and cpf for swarm ops	16 jun 2017 14:46

**Commit:** 905f436a58bd00e20a1c750b7ae10ae76a13397 [905f436]  
**Parents:** 1df6bb9446  
**Author:** henrikbaerbak <hbc@cs.au.dk>  
**Date:** 25. august 2017 09:50:10  
**Committer:** henrikbaerbak

After course cleanup, some live-demo stuff added

- mcd-mq.cpf
- swarm.cpf
- droplet.cpf



# Best Practices

AARHUS UNIVERSITET

- I have developed a practice of a 'tag line'

Graph	Description
	Milestone: Updated all (most?) CPF files to reflect the new format for CPF keys (CONNECTOR_IMPLEMENTATION and SERVER_ADDRESS)
	Release/Milestone. CaveService introduced as standalone service. updateRoom pending.
	Release/Milestone: CaveService as standalone service introduced. Major update of factory
	Fixed /info paht on cave service
	Updated CaveService spark server to have GET on /exits/(x.v.z) supported by tests
	Augmented CaveService interface with 'getC
	Augmented CaveService interface with 'getC
	Snapshot: Before adding getExits() to caveSe
	break.
	Broken test: merged dev stuff. now test case
	Moved FakeStorage datastructure initializati
	diary update
	Snapshot: Failed tests but most code is in pla
	MileStone: Added getter in StdObjMgr for ge
	Broken snapshot. In proces of intro getGene
	Testcases pass; bu
	Added diary notes from the abandoned expe
	Release o
	MileStone: weather service also in new cfg
	Inspector now following the new factory conventions.

**Release:** All features are working now  
**Milestone:** Major (part) feature working now  
**Snapshot:** Safe ground to retract to, all tests pass, typically before starting new feature TDD.  
**Broken:** Failing test case present, show 'I got to this point before taking a break'



- ***Commits may break but pushed ones may not***
  - I sometimes commit broken builds if I must change work task
    - They highlight what I am working on to myself the next day!
  - But never push them
    - Pushed commits must reflect a finished step/feature/bugfix/**all tests pass**
  - But – best practice is of course that also commits have **all tests passing**



# Mandatory Note

- **Use AU GitLab.**
- Use your own login name on Git repo when you are in the 'driver seat' = programmer role in TDD
- TAs are instructed to review you logs for
  - Clarity and sensible commit logs
  - 'small steps and commit often'
  - **Equal workload of each group participant**