AARHUS UNIVERSITET

Software Engineering and Architecture

Git in Practice

Git

- A highly personalized view, so take it as my opinion
- Git is *driving a Ferari 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

AARHUS UNIVERSITET

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**

AARHUS UNIVERSITET

I am not alone ☺

AARHUS UNIVERSITET

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.

AARHUS UNIVERSITET

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

Branching

- 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**

AARHUS UNIVERSITET

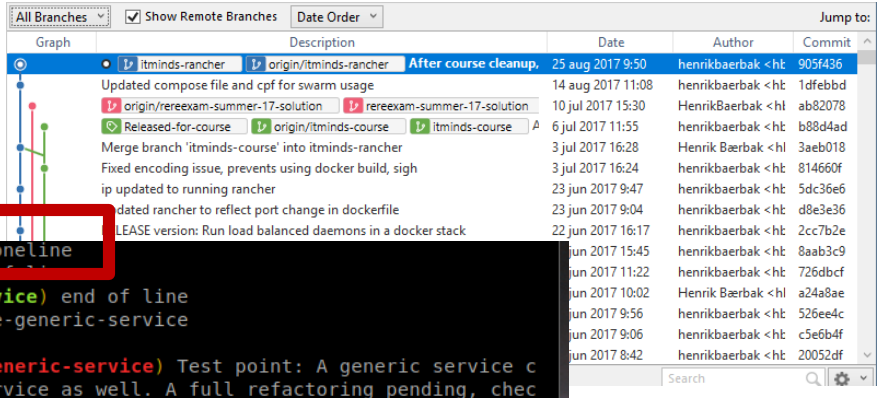
.gitignore

- **Do not polute 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

AARHUS UNIVERSITET

My own workflow

- Overviewing branches without graphics is hard!
- I develop on Lubuntu
  - But makes most branching/merging in SourceTree on Windows ☺
- But can be viewed in shell:




```
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
reator is intro in objmgr+factory. Demo that it can be used for quote service as well. A full refactoring pending, chec
king to see if it is usable in the mandatory 2.1 solution
* | 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
g is still a big mess.
* |   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!
```

AARHUS UNIVERSITET

CheatSheet

- 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>
```

AARHUS UNIVERSITET

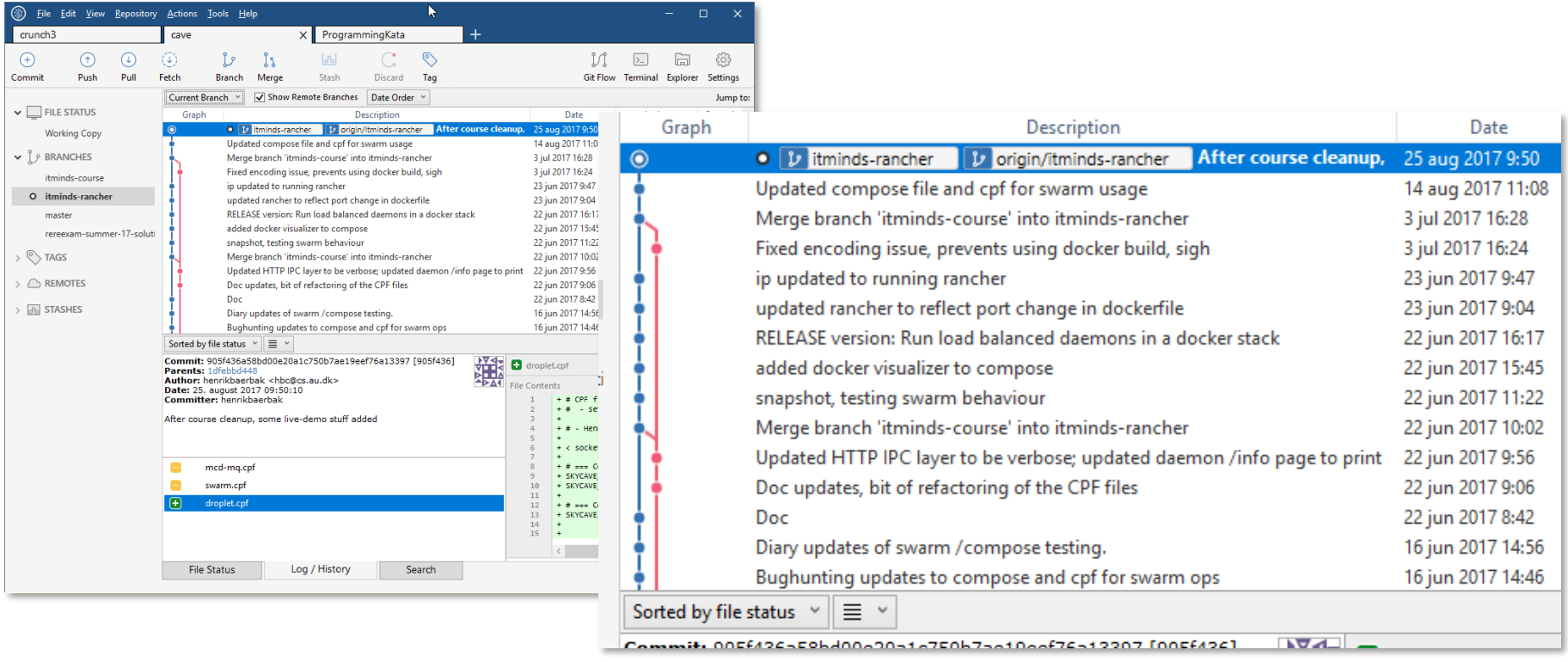
Best Practices

- 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!

AARHUS UNIVERSITET

Best Practices

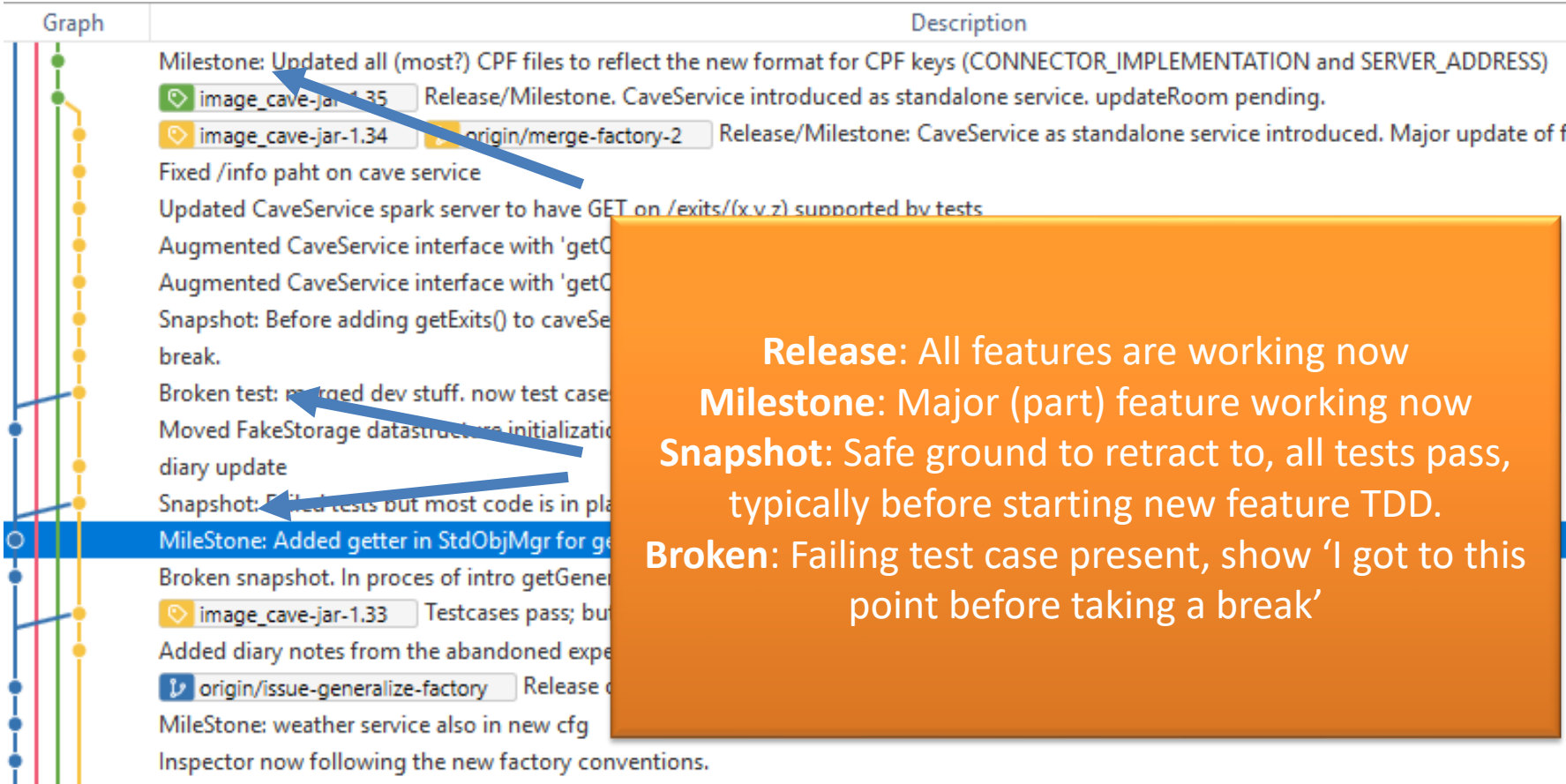
- **Use the commit log to express the goal achieved/contents of the commit**



AARHUS UNIVERSITET

Best Practices

- I have developed a practice of a ‘tag line’



AARHUS UNIVERSITET

Best Practices

- ***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**

AARHUS UNIVERSITET

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**