AARHUS UNIVERSITET

Software Engineering and Architecture

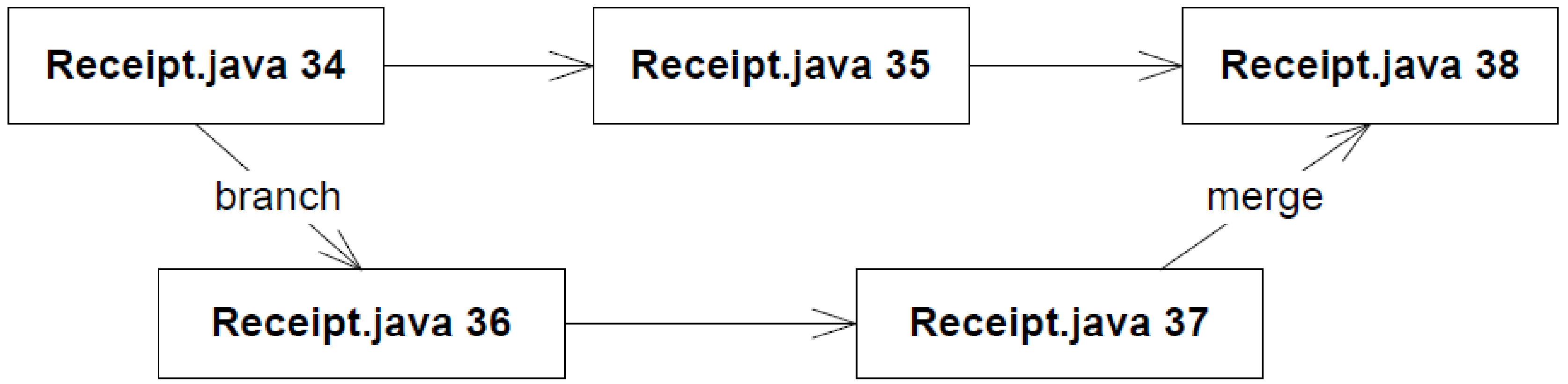
Release Management & Branching Models

AARHUS UNIVERSITET

Branching

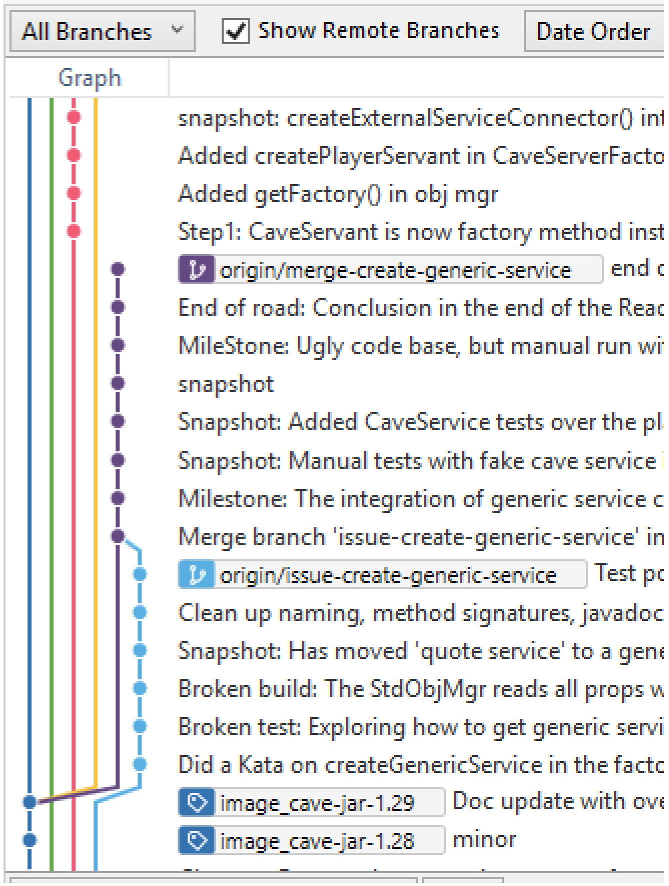
Definition: Branch

A branch is a point in the version graph where a version is ancestor to two or more descendant versions.



Git branches

- Git is *really strong in branching support !*
  - Why? Because it is a powerfull development tool…
- Example: *feature branch*
  - Arne makes branch ‘add-german’ and change code without interfering with Bente
  - Bente makes branch ‘fix-bug-21’ and fixes – well – bug #21
- Merge back when done
  - Or orphan branch if really bad idea…

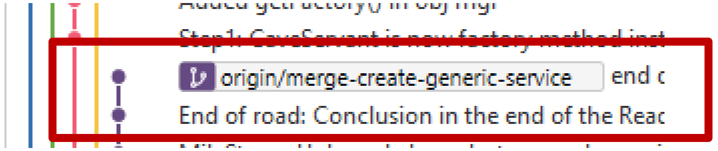


```
snapshot: createExternalServiceConnector() int
Added createPlayerServant in CaveServerFacto
Added getFactory() in obj mgr
Step1: CaveServant is now factory method inst
origin/merge-create-generic-service   end c
End of road: Conclusion in the end of the Reac
MileStone: Ugly code base, but manual run wi
snapshot
Snapshot: Added CaveService tests over the pl
Snapshot: Manual tests with fake cave service
Milestone: The integration of generic service c
Merge branch 'issue-create-generic-service' in
origin/issue-create-generic-service   Test pc
Clean up naming, method signatures, javadoc
Snapshot: Has moved 'quote service' to a gene
Broken build: The StdObjMgr reads all props w
Broken test: Exploring how to get generic servi
Did a Kata on createGenericService in the factc
image_cave-jar-1.29   Doc update with ove
image_cave-jar-1.28   minor
```

Software is a Lab!

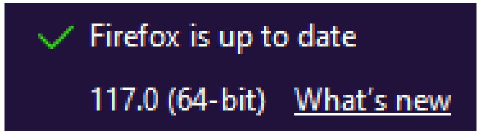
- I do a lot of *experiments* on my code!
  - I was originally trained as a physicist ☺
- *Experiment = I think this is a good idea, but do not know?*
- *How do I get 'to know!'* *By doing it!*

- Make an experimental branch in git
  - It was a good idea! Merge that branch into main
  - It was a bad idea!!! Orphan the branch


Step1: CaveServant is now factory method inst
origin/merge-create-generic-service end c
End of road: Conclusion in the end of the Reac

Release Management

- *You need to know what you release!*
  - Users report bugs and you need to fix them fast on the right code



Firefox is up to date

117.0 (64-bit) What's new

- Example
  - Release to AlphaTown
    - Rewrite part of the AlphaTown code to support BetaTown
      - (Major refactorings in core AlphaTown code)
  - AlphaTown phones us ”Hurry, fix major bug *now*”
    - But the code base is in a ‘state of flux’ (read: messy, broken, …) and also includes new features that AlphaTown has not paid for
  - What to do???

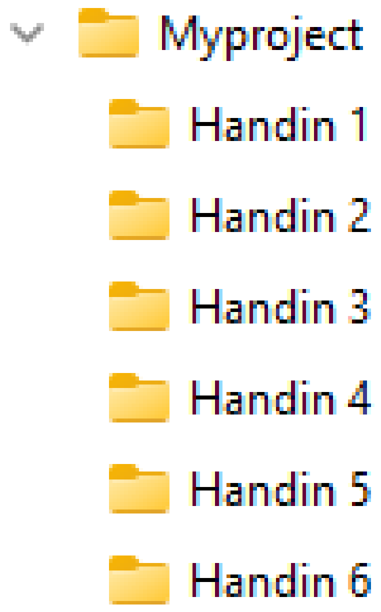
AARHUS UNIVERSITET

Not all versions are equal

- Some versions attain a special meaning: **Release**
- How to manage?
  - Write down the version identity. Git: 4ef678a…
  - ‘Tag’ a version on the graph.
    - Essentially put a human readable *label* on specific version
  - Make a ‘release branch’ (**single release branch model)**
    - Branch and name the new branch ‘Release-AlphaTown-V1.7.4’
  - Merge into a ‘release branch’ and tag it (**major release branches model)**
    - Merge current version into global release branch and tag it
  - **Main** branch is the ‘release branch’, no dev on this branch
    - “GitHub Flow” model

(SideNote)

- Some of you have in a previous course handed in mandatories using Git but made *one folder pr hand-in*?
  - I.e. ‘releases’ of the mandatory project
- This is **not the software dev way!**
  - This is a 1990’es manual hack in the absence of a version control system
- SWEA: We use Git to do release management



Two Release Management Models

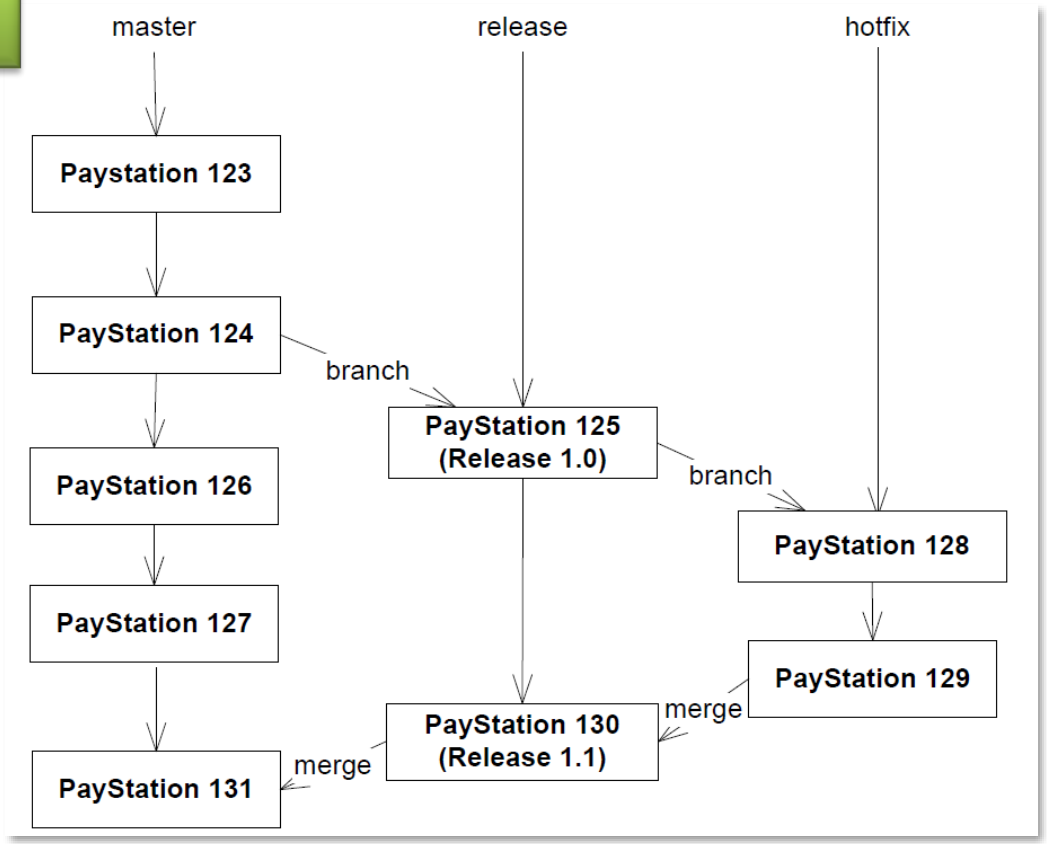
- Single Release Branch — Next slide
  - Daily development on ‘master’
  - New release => Merge into ‘release’ branch
  - Pro: Always find release as tip on release branch

- Major Release Branch*es* — Next+1 slide
  - New release => Create *new* branch
  - Pro: Naming the releases by the branch name

- *Used in SWEA up until E2020…*

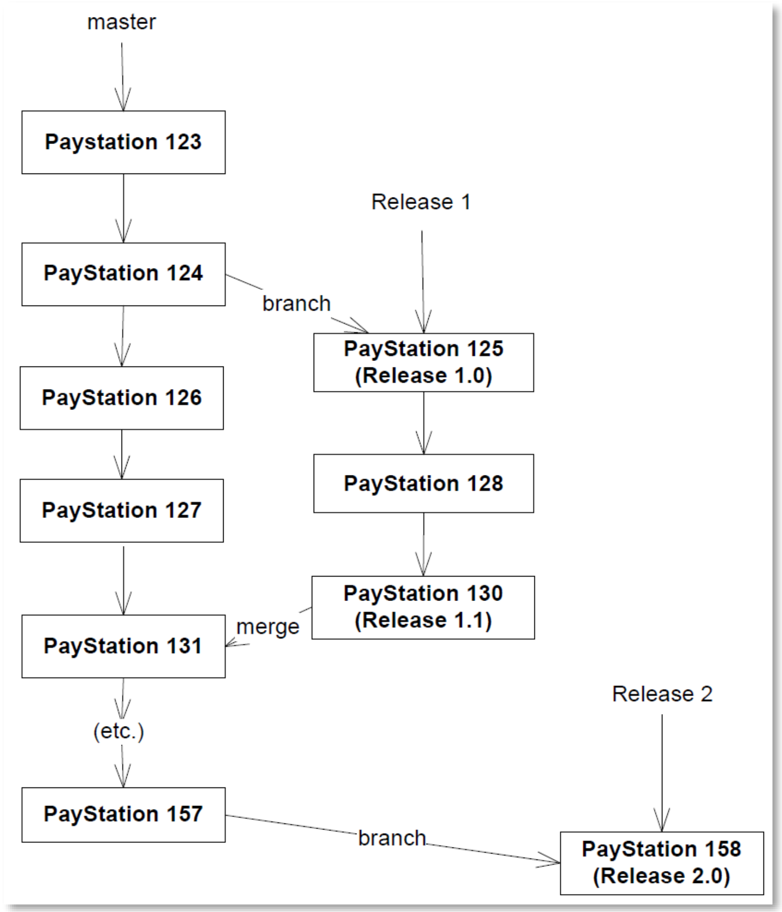
Simple Release Model A

- Single Release Branch
  - Hotfixing must be done on separate branch
  - And merged back



Simple Release Model B

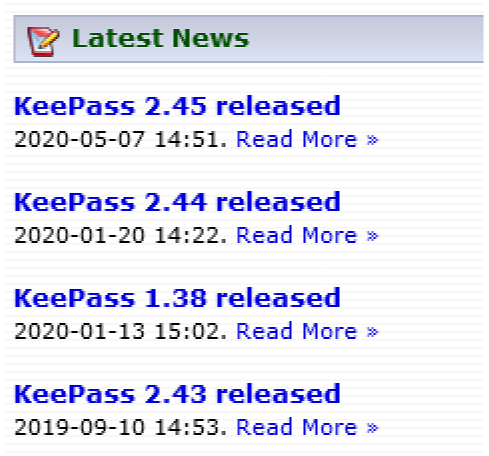
- Major Release Branches
  - Each major release give rise to new branch



AARHUS UNIVERSITET

Continuous Deployment

- Release Management is important but…
  - There is a distinct *release process involved*
  - *I download the latest release and install*
- Lots of modern software does *not follow that paradigm*
  - You do not download & install facebook
  - Web systems are *continuously* updated…


Latest News

KeePass 2.45 released
2020-05-07 14:51. Read More »

KeePass 2.44 released
2020-01-20 14:22. Read More »

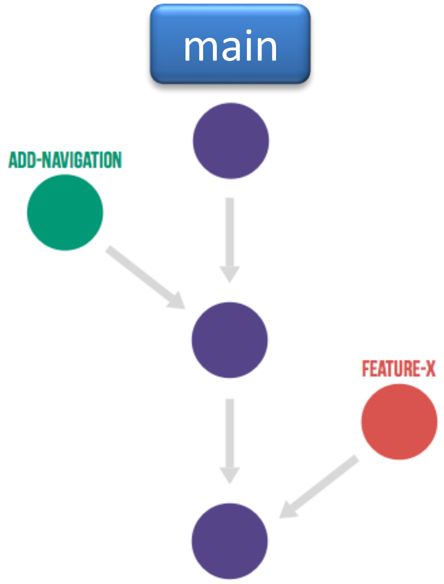
KeePass 1.38 released
2020-01-13 15:02. Read More »

KeePass 2.43 released
2019-09-10 14:53. Read More »

- CD = You *continuously* get the latest release
  - Releasing every couple of hours! Done by machines…

CD Release Management

- CD streamlines release management!
  - 'main' is the release branch!
- Daily work done on *feature branches*
  - When feature/iteration is 'working'…
    - Tests pass, requirements complete
  - … you merge back into master
- **GitHub Flow**
  - [https://docs.github.com/en/get-started/quickstart/github-flow]
  - Note: This release management model is **not tied to GitHub!**



AARHUS UNIVERSITET

SWEA Relation

- In the SWEA mandatory project…
- You should create an **'iteration branch'** that holds the development in the given iteration / delivery
- Like branch 'iteration3' =
  - Work on the requirements for mandatory 'iteration 3'
    - Contains 'work in progress' code, not suitable for customers
  - **But 'main' branch can always be released**
    - **Because it is correct, working, without bugs, stable, latest…**

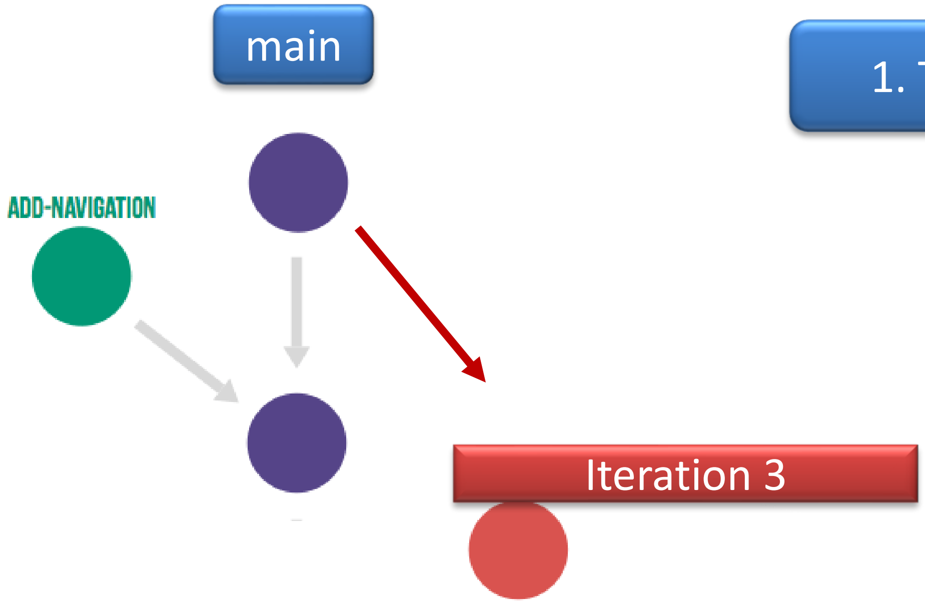
AARHUS UNIVERSITET

GitHub Flow

In Practice

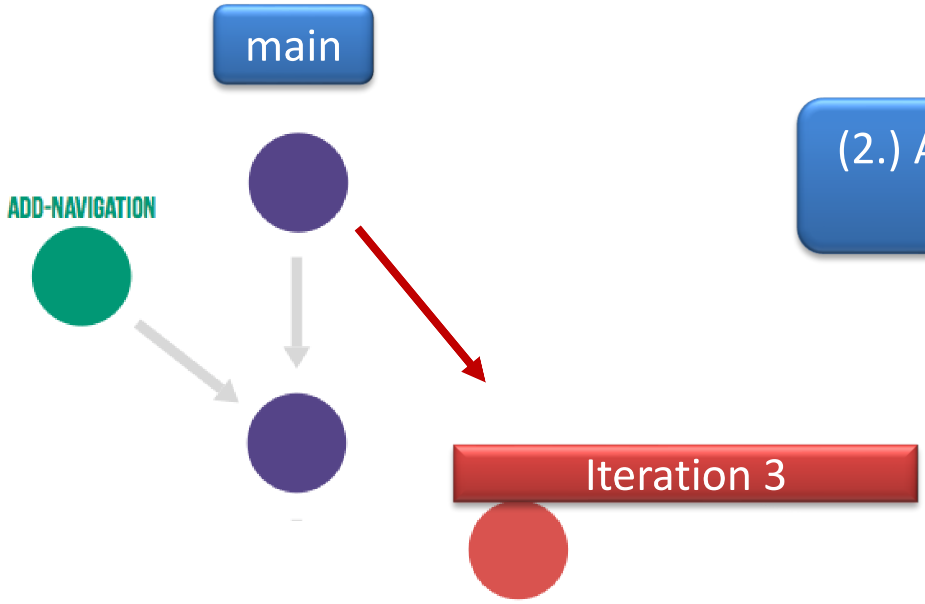
AARHUS UNIVERSITET

Overview



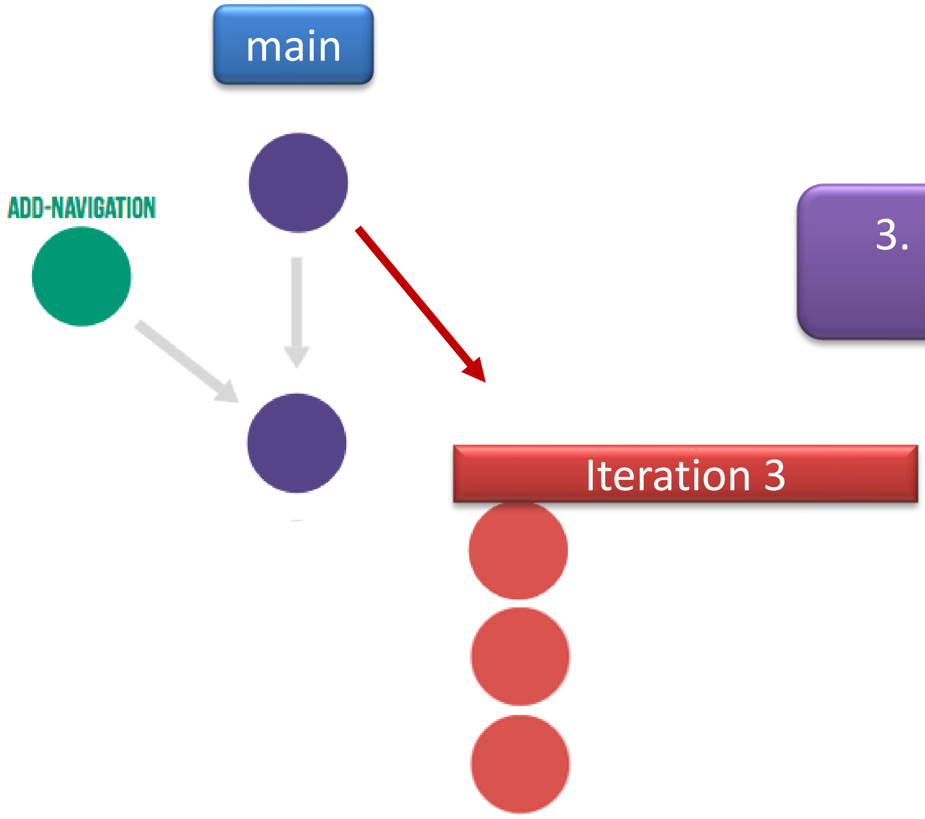
1. Tell Git to create branch

Overview



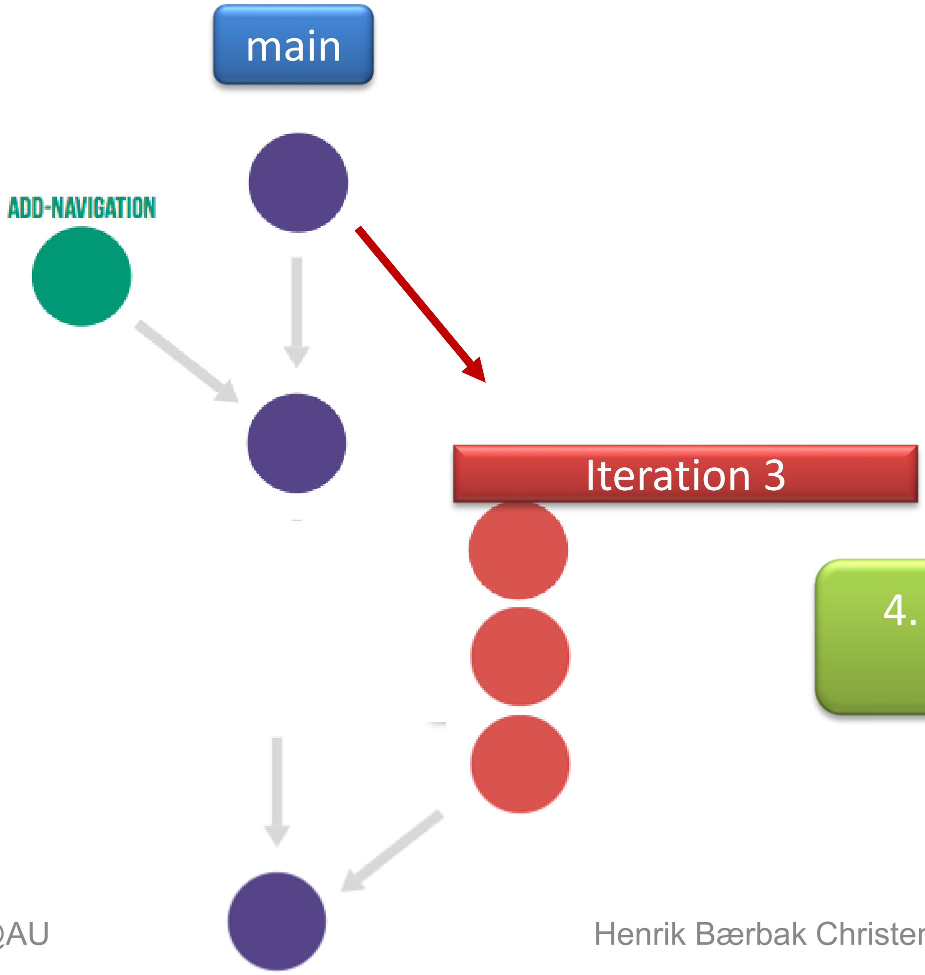
(2.) Associate 'Merge Request' with the branch

Overview



3. Develop on that branch, commit often ☺

Overview

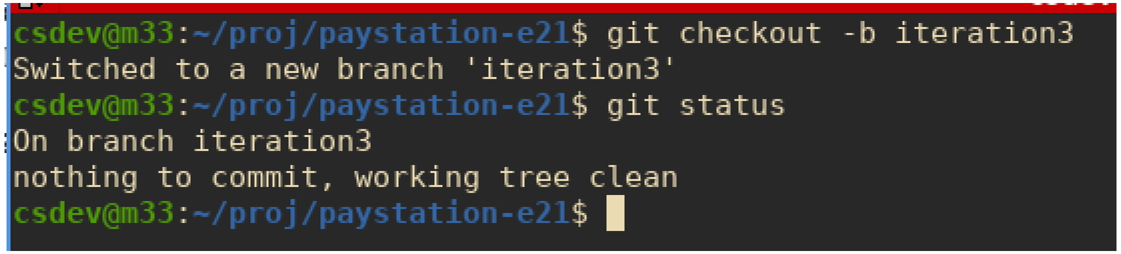


4. Release: Merge back into main branch

AARHUS UNIVERSITET

Starting Iteration Work

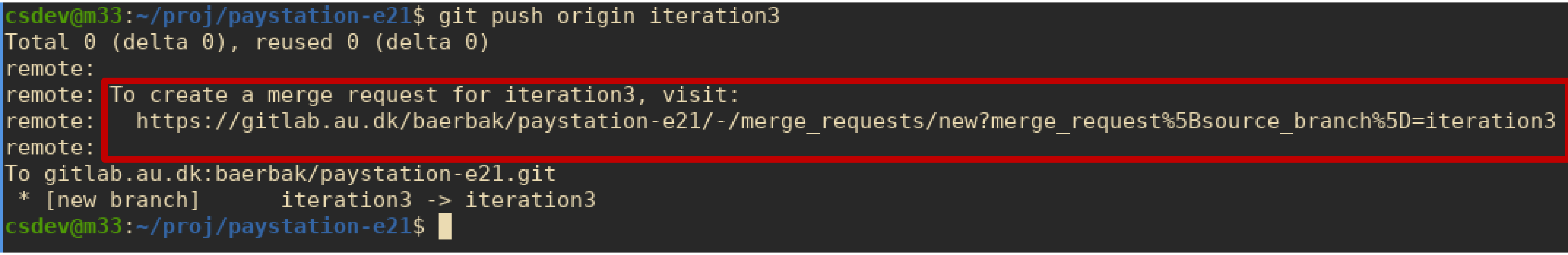
- *Let us start on the exciting mandatory 3 – hurrah!*



```
csdev@m33:~/proj/paystation-e21$ git checkout -b iteration3
Switched to a new branch 'iteration3'
csdev@m33:~/proj/paystation-e21$ git status
On branch iteration3
nothing to commit, working tree clean
csdev@m33:~/proj/paystation-e21$ 
```

To see same procedure in IntelliJ's git, see screencasts on week plan…

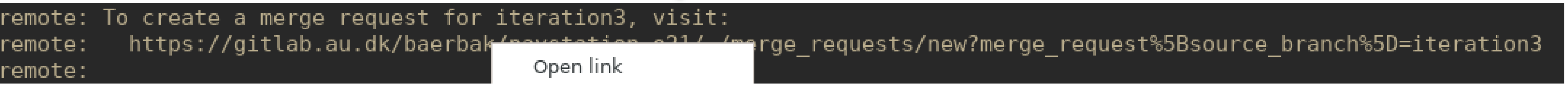
- Tell GitLab about the branch
  - Link will be provided if you want to create a 'merge request'



```
csdev@m33:~/proj/paystation-e21$ git push origin iteration3
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for iteration3, visit:
remote:   https://gitlab.au.dk/baerbak/paystation-e21/-/merge_requests/new?merge_request%5Bsource_branch%5D=iteration3
remote:
To gitlab.au.dk:baerbak/paystation-e21.git
 * [new branch]      iteration3 -> iteration3
csdev@m33:~/proj/paystation-e21$ 
```

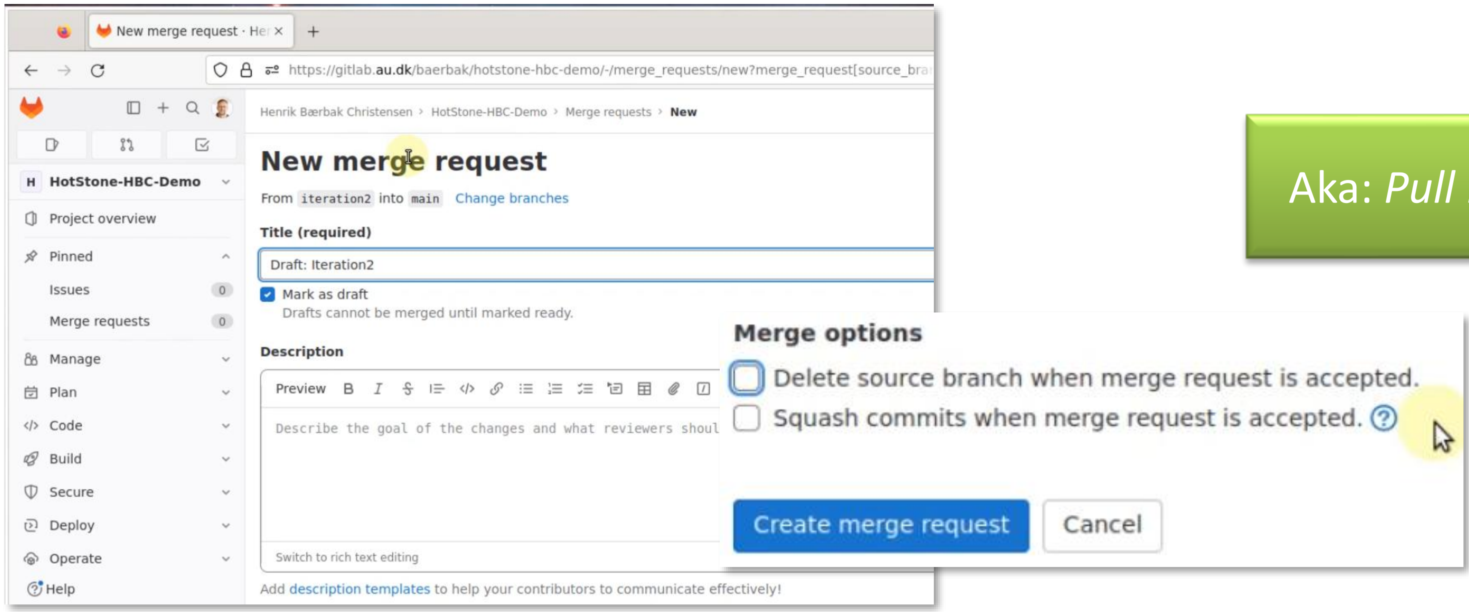
(Merge Request)

- Follow the link that Git provides



```
remote: To create a merge request for iteration3, visit:
remote:   https://gitlab.au.dk/baerbak/              /merge_requests/new?merge_request%5Bsource_branch%5D=iteration3
remote:
```

- And fill in the details about *Description*, and 'Create…'



New merge request · He
https://gitlab.au.dk/baerbak/hotstone-hbc-demo/-/merge_requests/new?merge_request[source_bra
HotStone-HBC-Demo
Project overview
Pinned
Issues 0
Merge requests 0
Manage
Plan
Code
Build
Secure
Deploy
Operate
Help
Henrik Bærbak Christensen › HotStone-HBC-Demo › Merge requests › New

New merge request

From `iteration2` into `main` Change branches

Title (required)

Draft: Iteration2

[✓] Mark as draft
Drafts cannot be merged until marked ready.

Description

Describe the goal of the changes and what reviewers shoul

Switch to rich text editing

Add description templates to help your contributors to communicate effectively!

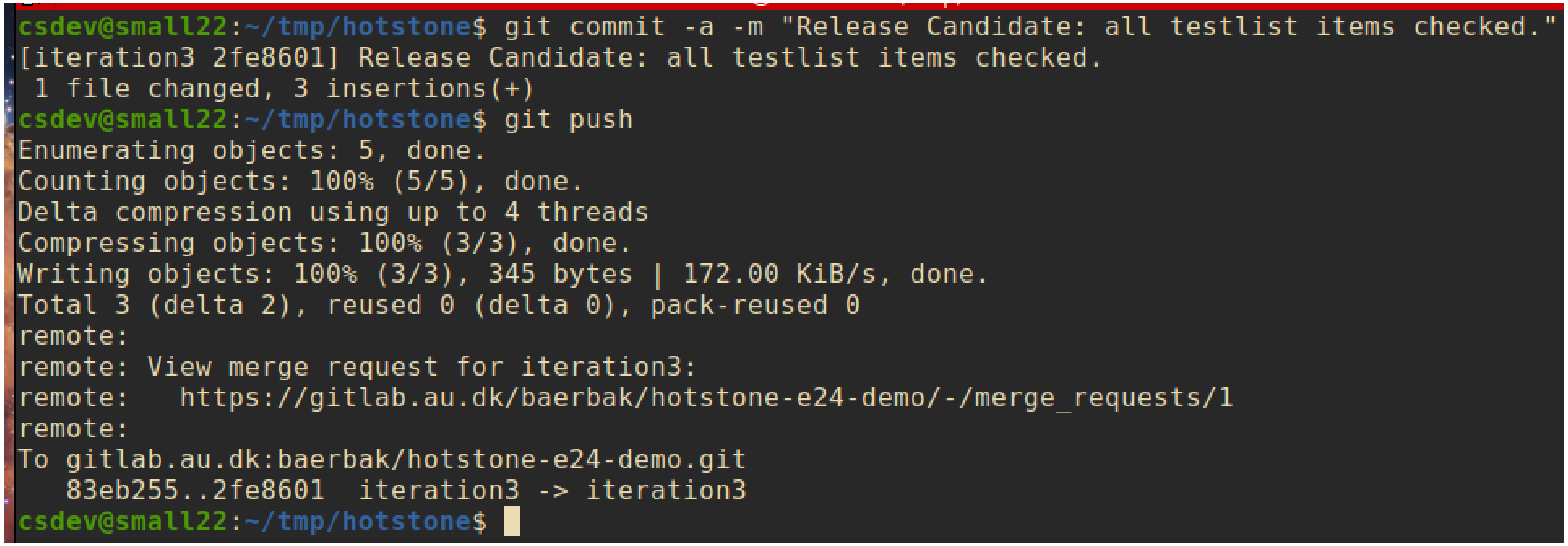
Merge options

[ ] Delete source branch when merge request is accepted.
[ ] Squash commits when merge request is accepted.

Aka: *Pull Request*

Now: Work ☺

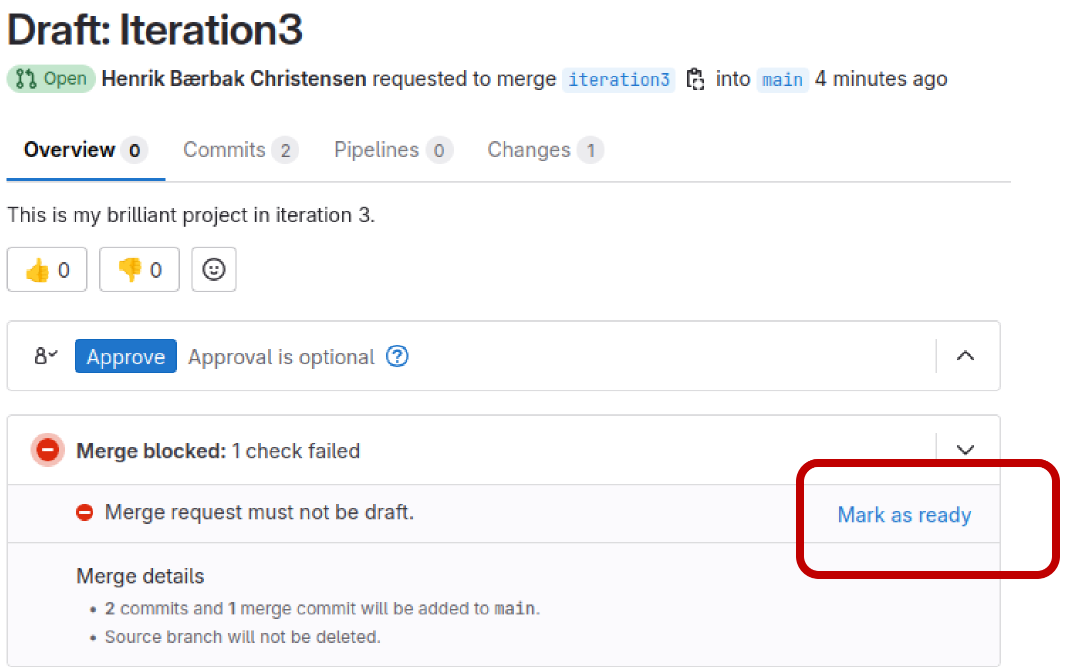
- Do the TDD
  - Do a ‘commit and push’ after each finished TDD iteration



```
csdev@small22:~/tmp/hotstone$ git commit -a -m "Release Candidate: all testlist items checked."
[iteration3 2fe8601] Release Candidate: all testlist items checked.
 1 file changed, 3 insertions(+)
csdev@small22:~/tmp/hotstone$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 345 bytes | 172.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote:
remote: View merge request for iteration3:
remote:   https://gitlab.au.dk/baerbak/hotstone-e24-demo/-/merge_requests/1
remote:
To gitlab.au.dk:baerbak/hotstone-e24-demo.git
   83eb255..2fe8601  iteration3 -> iteration3
csdev@small22:~/tmp/hotstone$
```

Release Time

- We find it is time to release
  - That is: this is the best shot at a mandatory hand-in
- (Mark the iteration branch as 'ready' in AU GitLab)



Draft: Iteration3

Open Henrik Bærbak Christensen requested to merge `iteration3` into `main` 4 minutes ago

This is my brilliant project in iteration 3.

Approval is optional

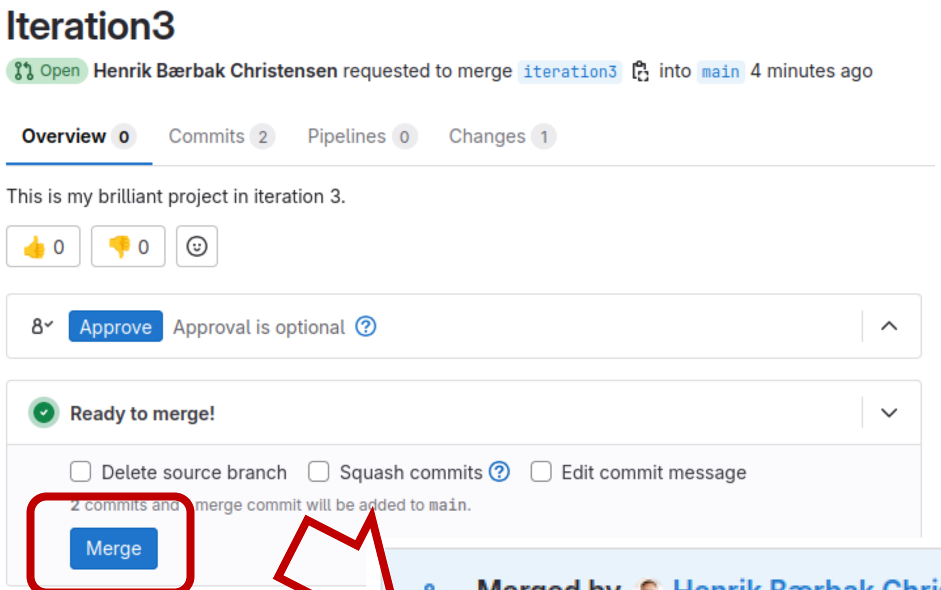
Merge blocked: 1 check failed

Merge request must not be draft.

Merge details

- 2 commits and 1 merge commit will be added to `main`.
- Source branch will not be deleted.

And merge back to Main

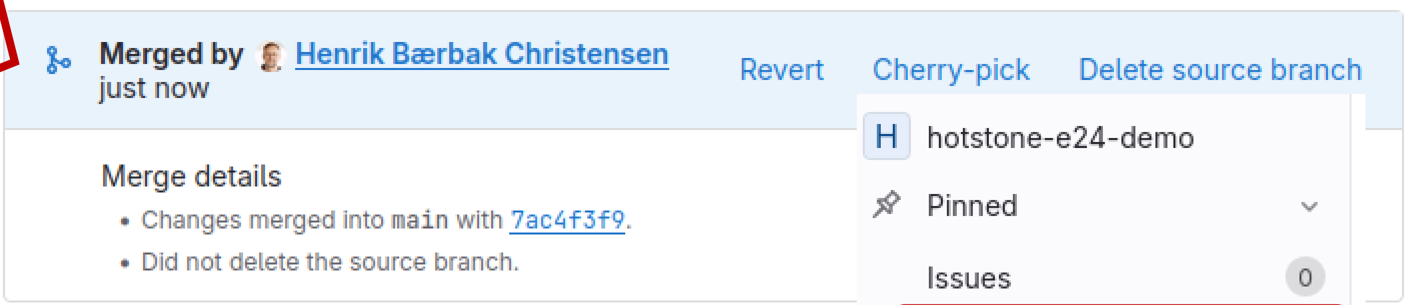


Iteration3

Open Henrik Bærbak Christensen requested to merge `iteration3` into `main` 4 minutes ago

This is my brilliant project in iteration 3.

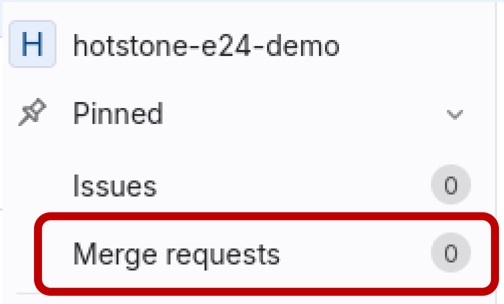
2 commits and 1 merge commit will be added to main.



Merged by Henrik Bærbak Christensen just now

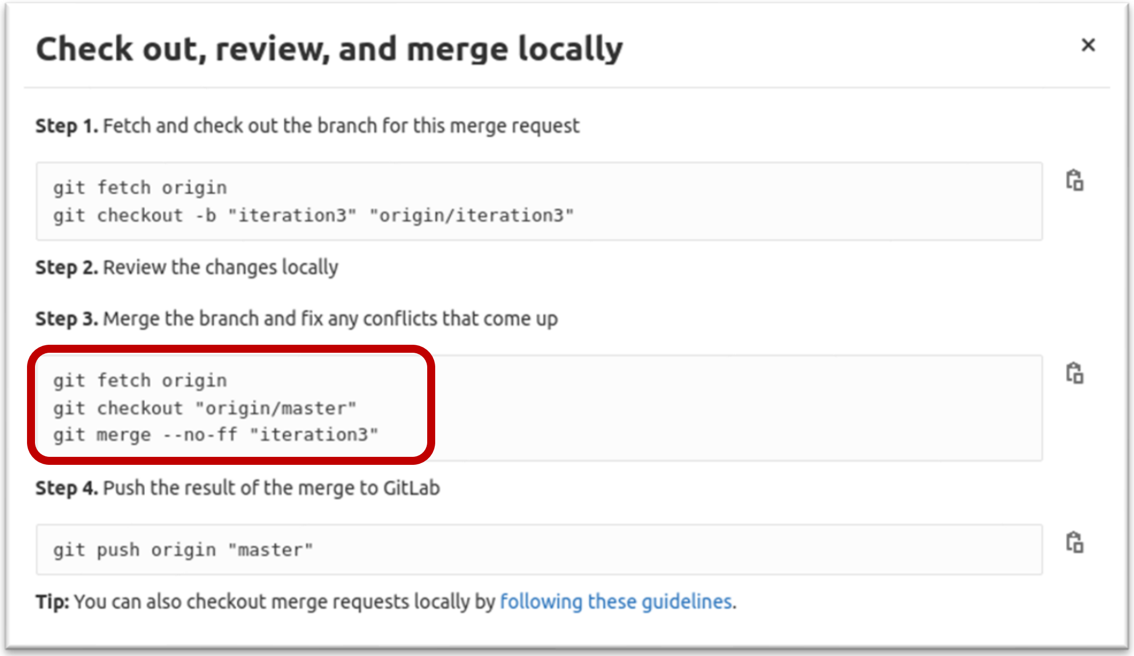
Merge details

- Changes merged into `main` with 7ac4f3f9.
- Did not delete the source branch.



Release Time

- The ‘**commandline**’ way



Check out, review, and merge locally

Step 1. Fetch and check out the branch for this merge request

```
git fetch origin
git checkout -b "iteration3" "origin/iteration3"
```

Step 2. Review the changes locally

Step 3. Merge the branch and fix any conflicts that come up

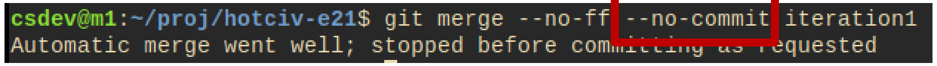
```
git fetch origin
git checkout "origin/master"
git merge --no-ff "iteration3"
```

Step 4. Push the result of the merge to GitLab

```
git push origin "master"
```

Tip: You can also checkout merge requests locally by following these guidelines.

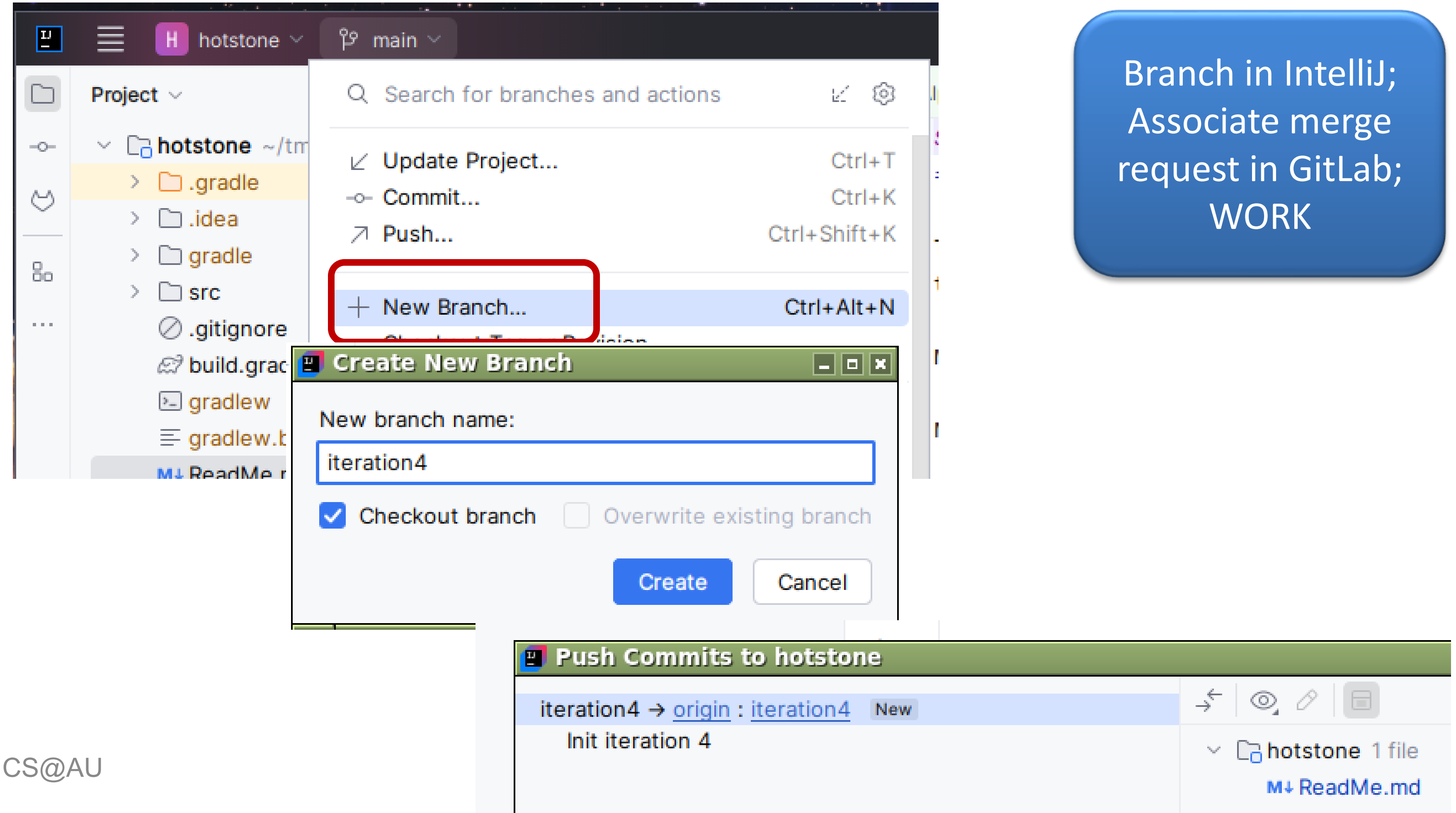
Or use the - -no-commit, to ‘dryrun’



```
csdev@m1:~/proj/hotciv-e21$ git merge --no-ff --no-commit iteration1
Automatic merge went well; stopped before committing as requested
```

AARHUS UNIVERSITET

Equivalent in IntelliJ



New branch name: iteration4

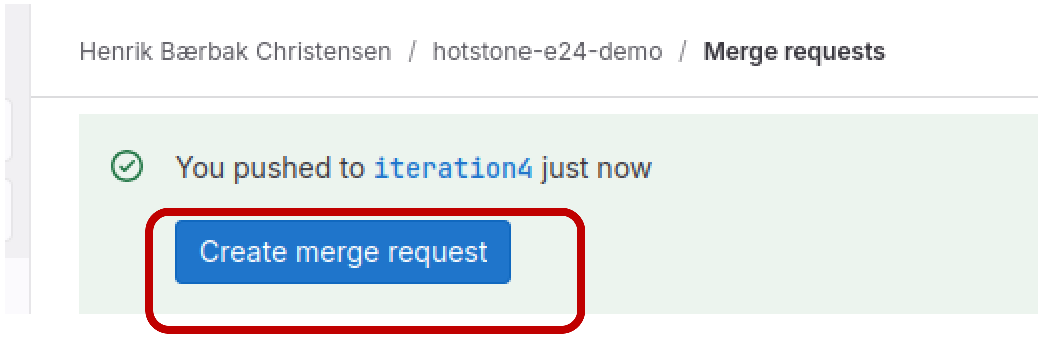
[✓] Checkout branch
[ ] Overwrite existing branch

iteration4 → origin : iteration4 New
Init iteration 4

Branch in IntelliJ;
Associate merge request in GitLab;
WORK

AARHUS UNIVERSITET

Over to GitLab



Henrik Bærbak Christensen / hotstone-e24-demo / Merge requests

You pushed to iteration4 just now

- And fill in the details as outlined earlier

AARHUS UNIVERSITET

Merge Request/Branch

- I have shown it here where a merge request is associated with the ‘iteration 3’ branch.
  - It is a bit overkill in SWEA context to create merge requests, so
  - It is *optional to do that in the mandatory…*

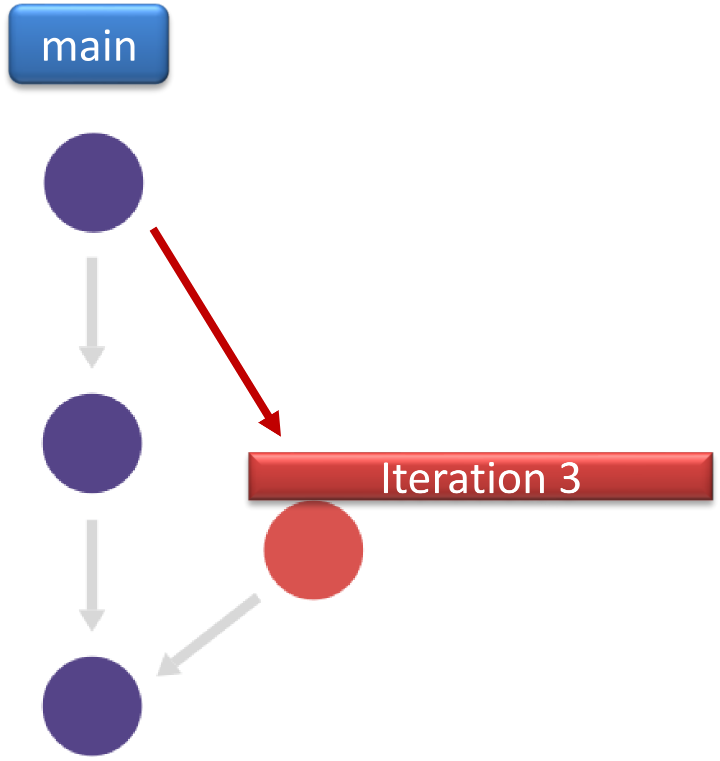
- **The branch is important, required in mandatory**
  - Working on an iteration branch is important

• git checkout –b iteration4	Create branch
• git commit & git push	Work on branch
• git checkout main; git merge iteration4	Merge back to main

AARHUS UNIVERSITET

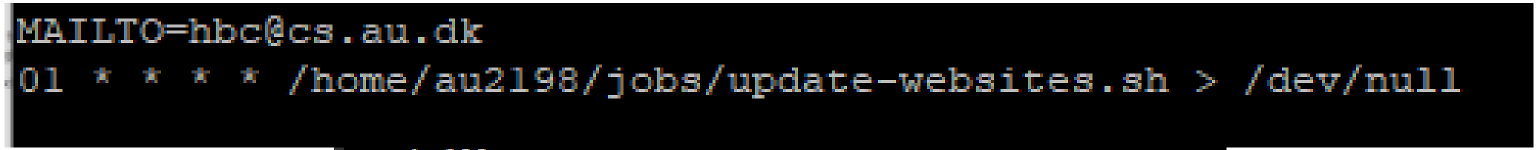
In the Branching Model

- Release is now present on the main branch.
- **The key point:**
  - **You can always release the software on the main branch!**
- CD = Cont. Delivery
  - Every 1 hour, a computer simply copies SW from main branch onto production machines

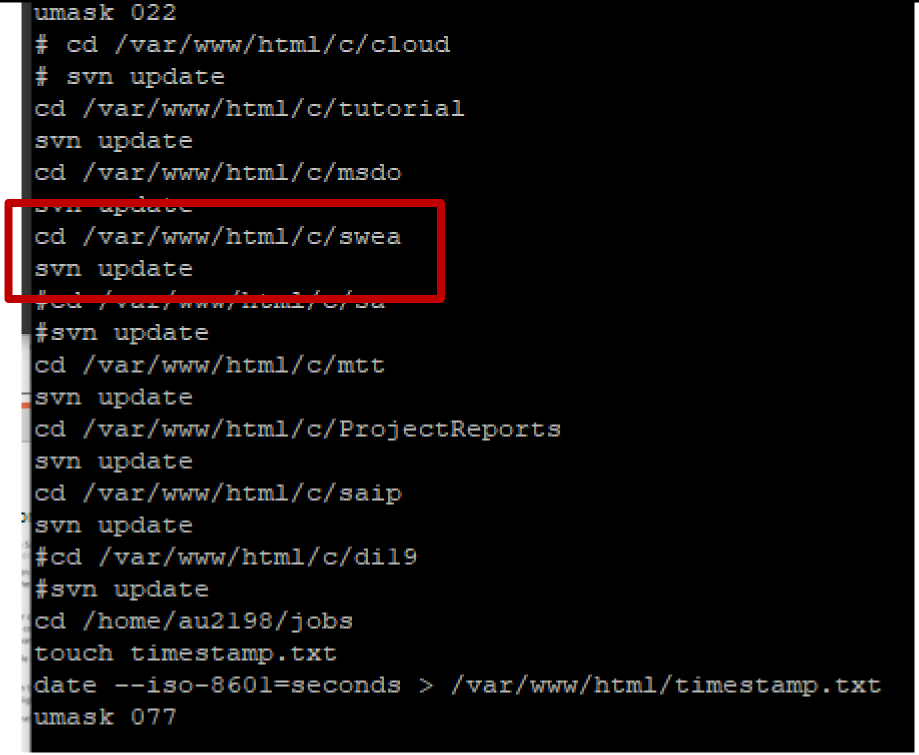


Simple Example

- Crontab on ‘baerbak.cs.au.dk’



```
MAILTO=hbc@cs.au.dk
01 * * * * /home/au2198/jobs/update-websites.sh > /dev/null
```



```
umask 022
# cd /var/www/html/c/cloud
# svn update
cd /var/www/html/c/tutorial
svn update
cd /var/www/html/c/msdo
svn update
cd /var/www/html/c/swea
svn update
#cd /var/www/html/c/sa
#svn update
cd /var/www/html/c/mtt
svn update
cd /var/www/html/c/ProjectReports
svn update
cd /var/www/html/c/saip
svn update
#cd /var/www/html/c/dil9
#svn update
cd /home/au2198/jobs
touch timestamp.txt
date --iso-8601=seconds > /var/www/html/timestamp.txt
umask 077
```

Summary

- Branching supports the release and development process
  - Releasing, bugfixing, subteams, feature branches, …
- Many different models can be made
  - **Keep it simple! Emphasize ease in daily work!**

- In SWEA we adopt a simple CD model – GitHub Flow
  - Latest working release on ‘main’
  - Do development on an ‘iteration’ branch,
    - Optional use ‘merge requests’