Git and GitLab Best Practices

Do you really use Git and Gitlab efficiently?

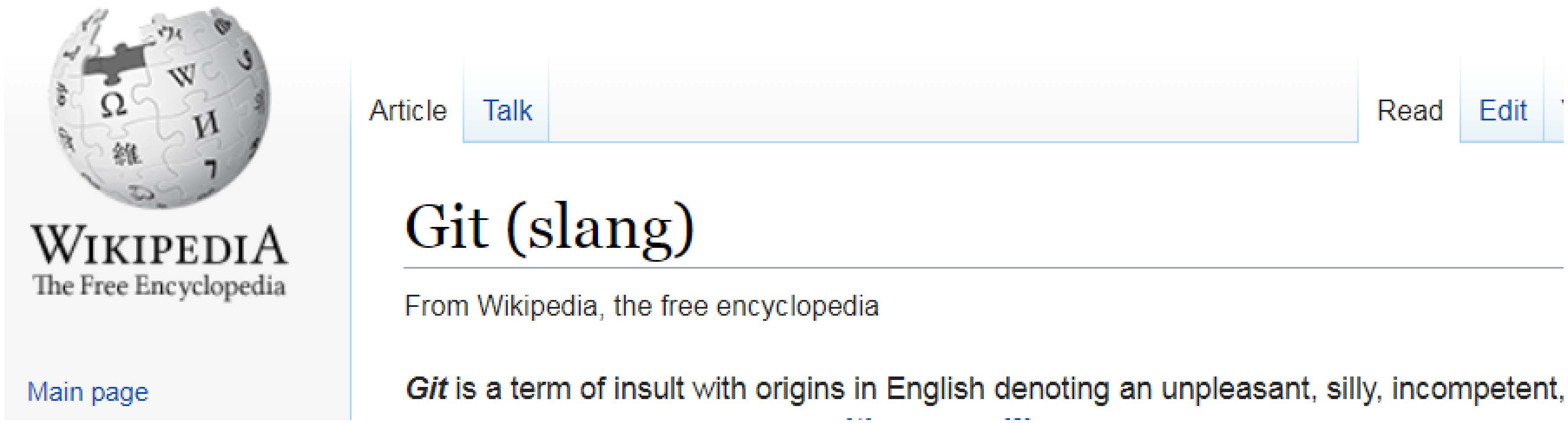
Ilya Rokhkin

2020

Git – meaning?

What does the word Git means?

GIT Overview

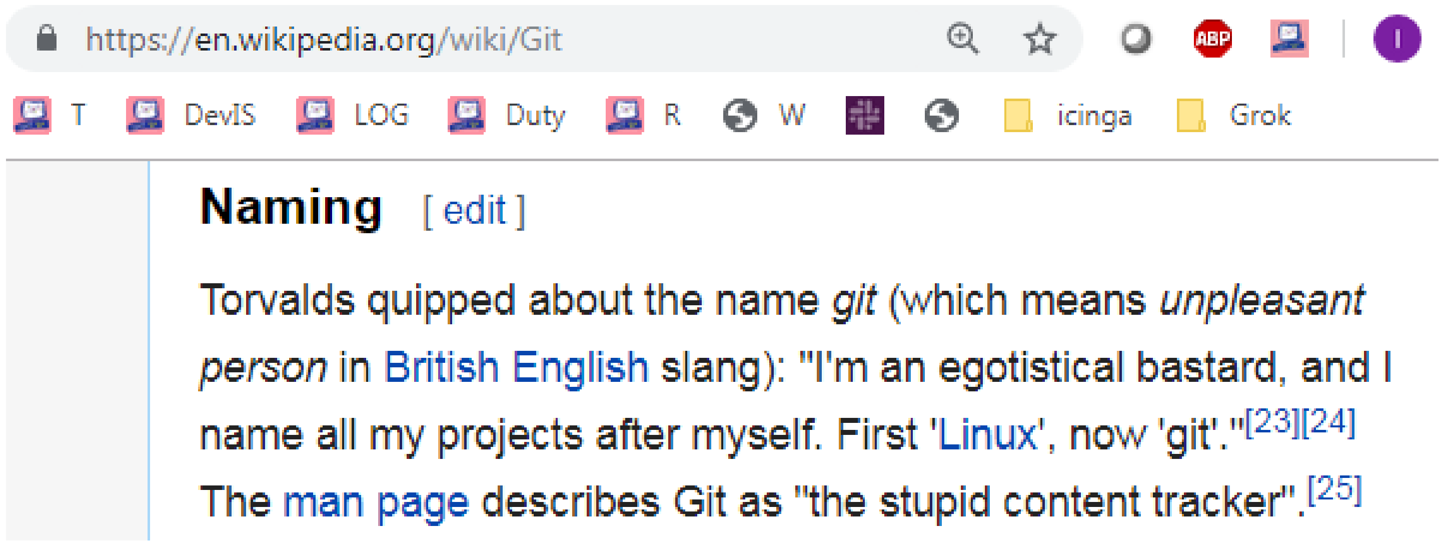


Article Talk Read Edit

Git (slang)

From Wikipedia, the free encyclopedia

Git is a term of insult with origins in English denoting an unpleasant, silly, incompetent,

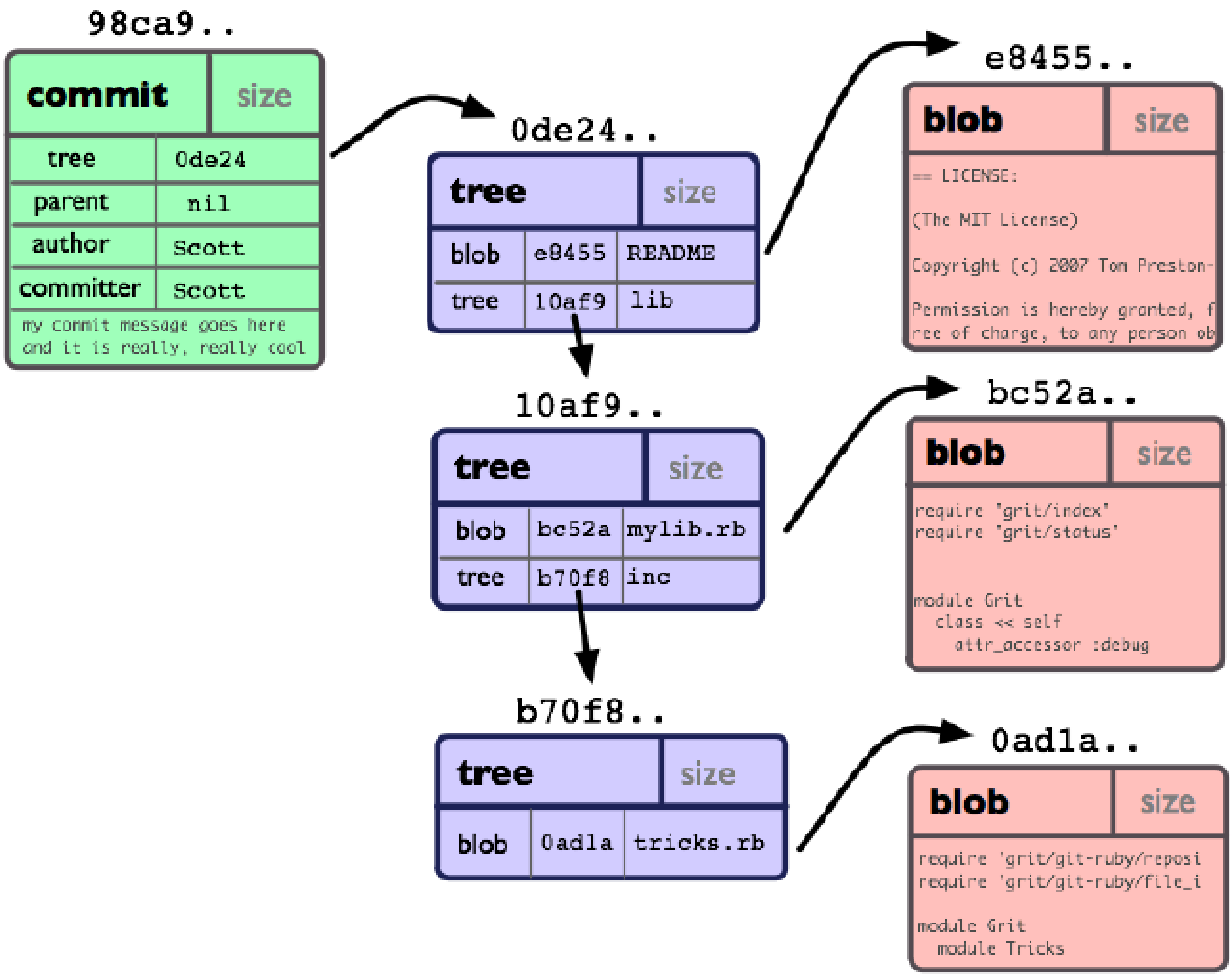


https://en.wikipedia.org/wiki/Git

Naming [edit]

Torvalds quipped about the name *git* (which means *unpleasant person* in British English slang): "I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'git'."[23][24] The man page describes Git as "the stupid content tracker".[25]

Commit Object



About myself: Ilya Rokhkin

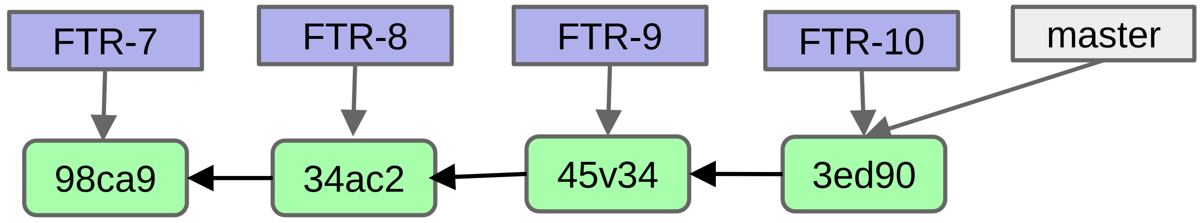
- 20+ years experience in Version Control Systems
- Official Git trainer in CKP, Intel, Marvell.
- Freelance Git trainer.
- Volunteer as Hebrew teacher in “Ulpan” 10+ years

Agenda:

1. Git + GitLab best practices in a workflow
2. Git + GitLab best practices hands on
3. GitLab Best practices
4. GitLab CI overview

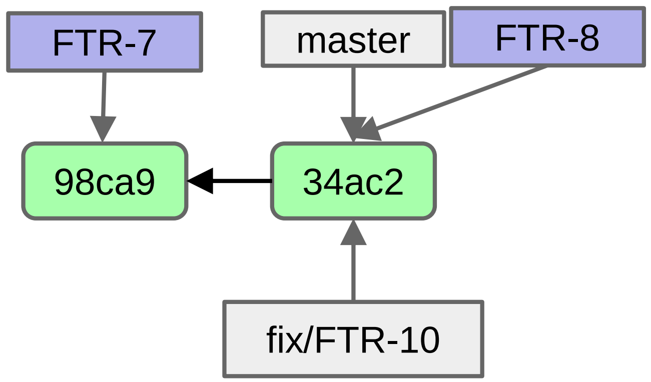
1. Git + GitLab best practices in work flow. Best practice: 1 Commit per Jira issue in main branches

Remote Repository



Create new feature branch FTR-10 (Jira)

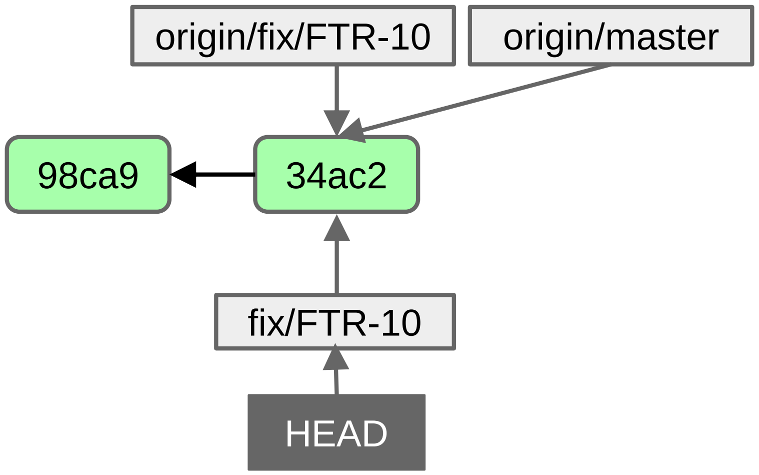
Remote Repository



Tracking Branch (Local)

Remote Tracking Branch

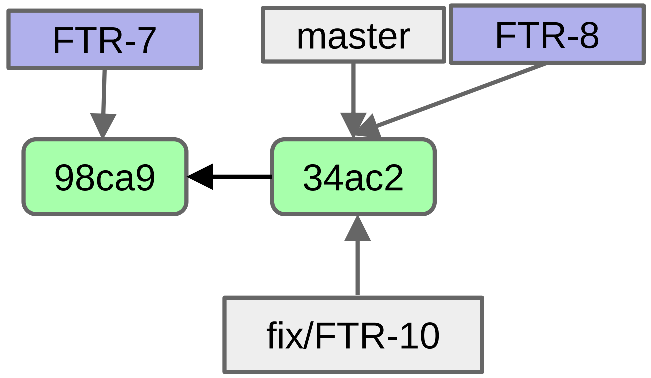
Local Repository



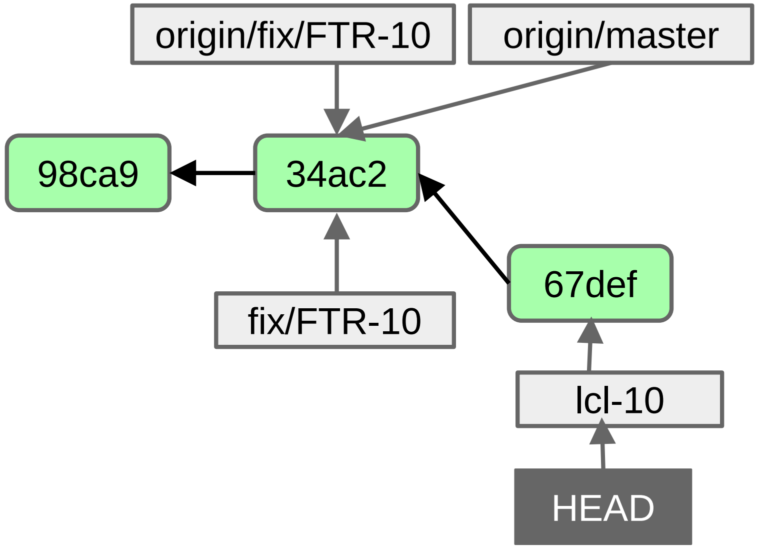
Clone + checkout feature branch FTR-10

Checkout local branch lcl-10, save->auto commit

Remote Repository

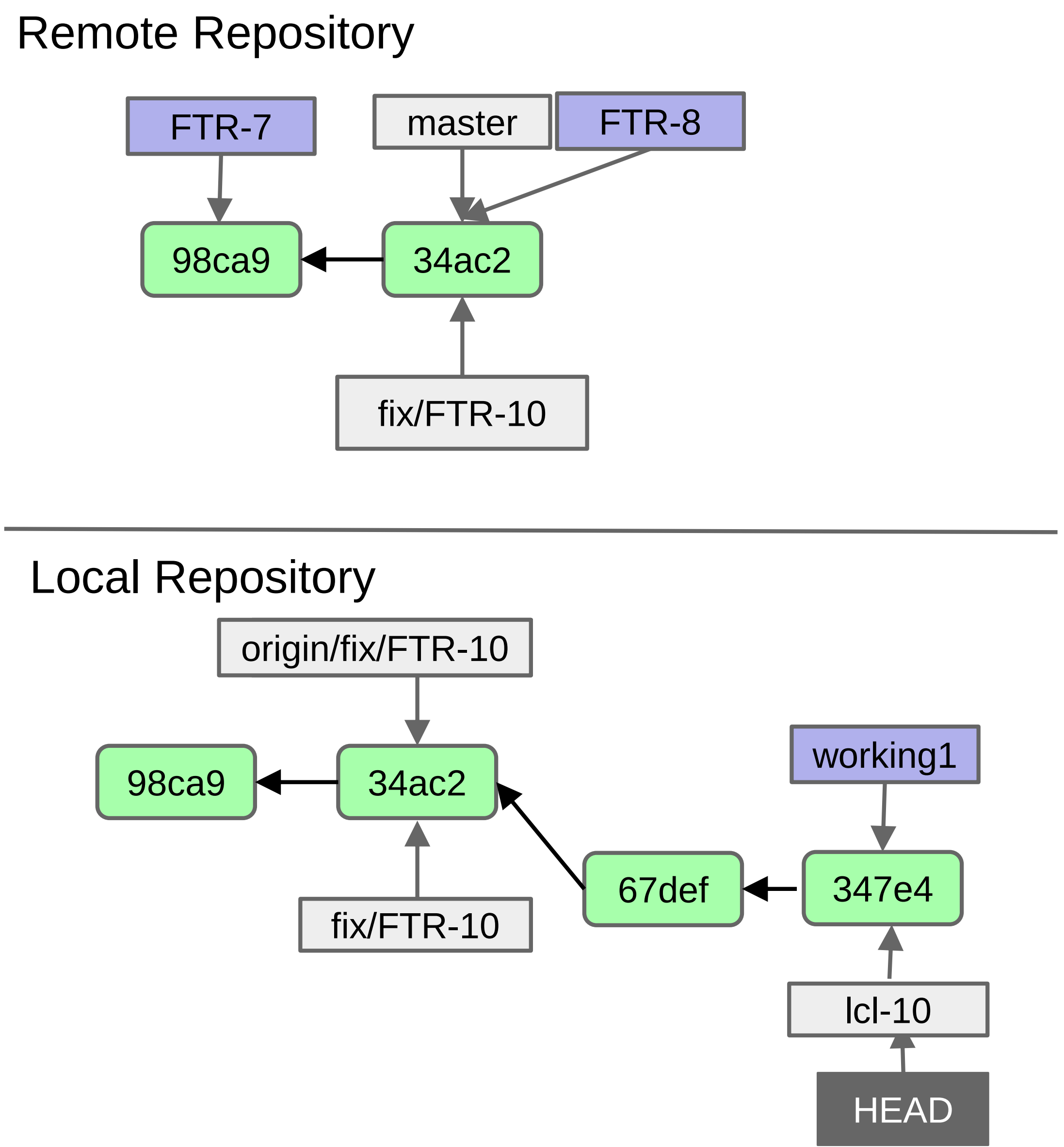


Local Repository



Checkout local branch LCL-10
Changing files + auto commit

Save -> auto commit, tag important changes: working1

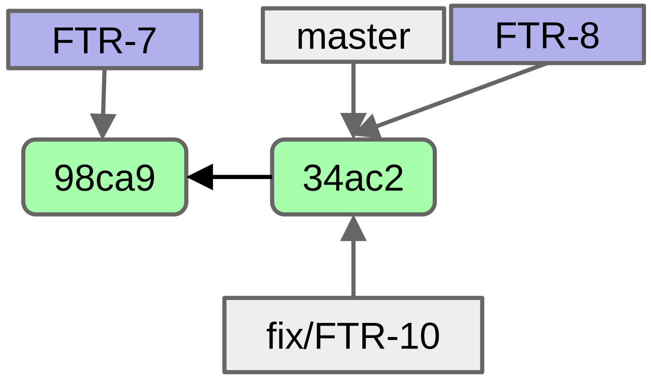


Each save -> auto commit

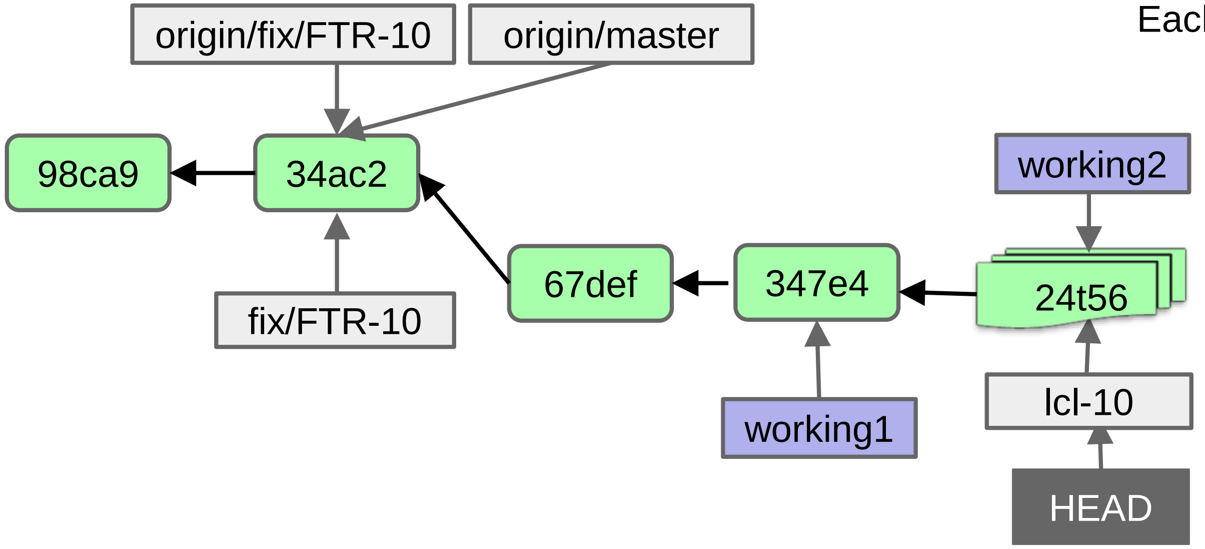
Tag working commits: workingN

Making how many auto commits as needed, tagging as needed working2

Remote Repository



Local Repository

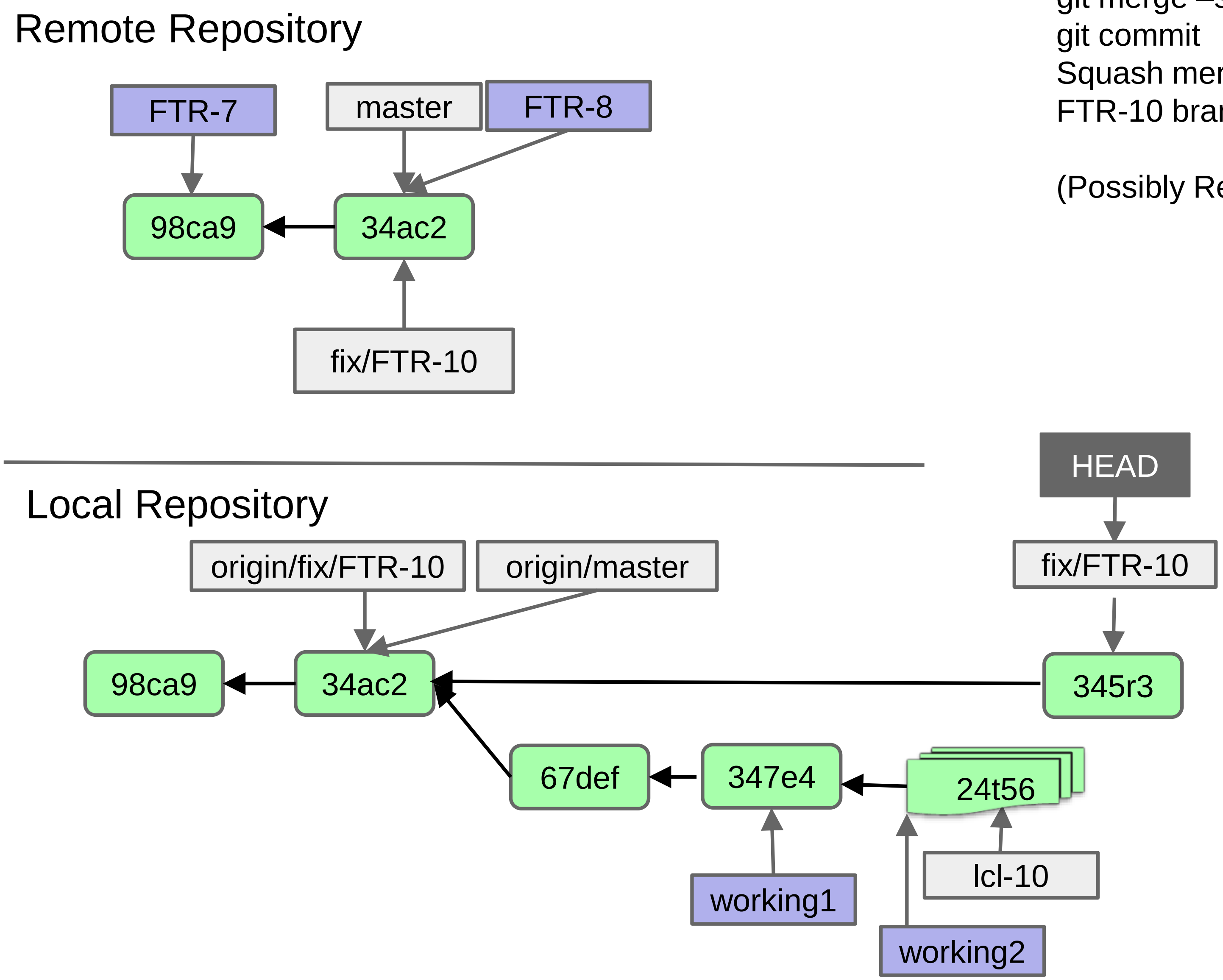


Changing files + auto commit
Each save -> auto commit

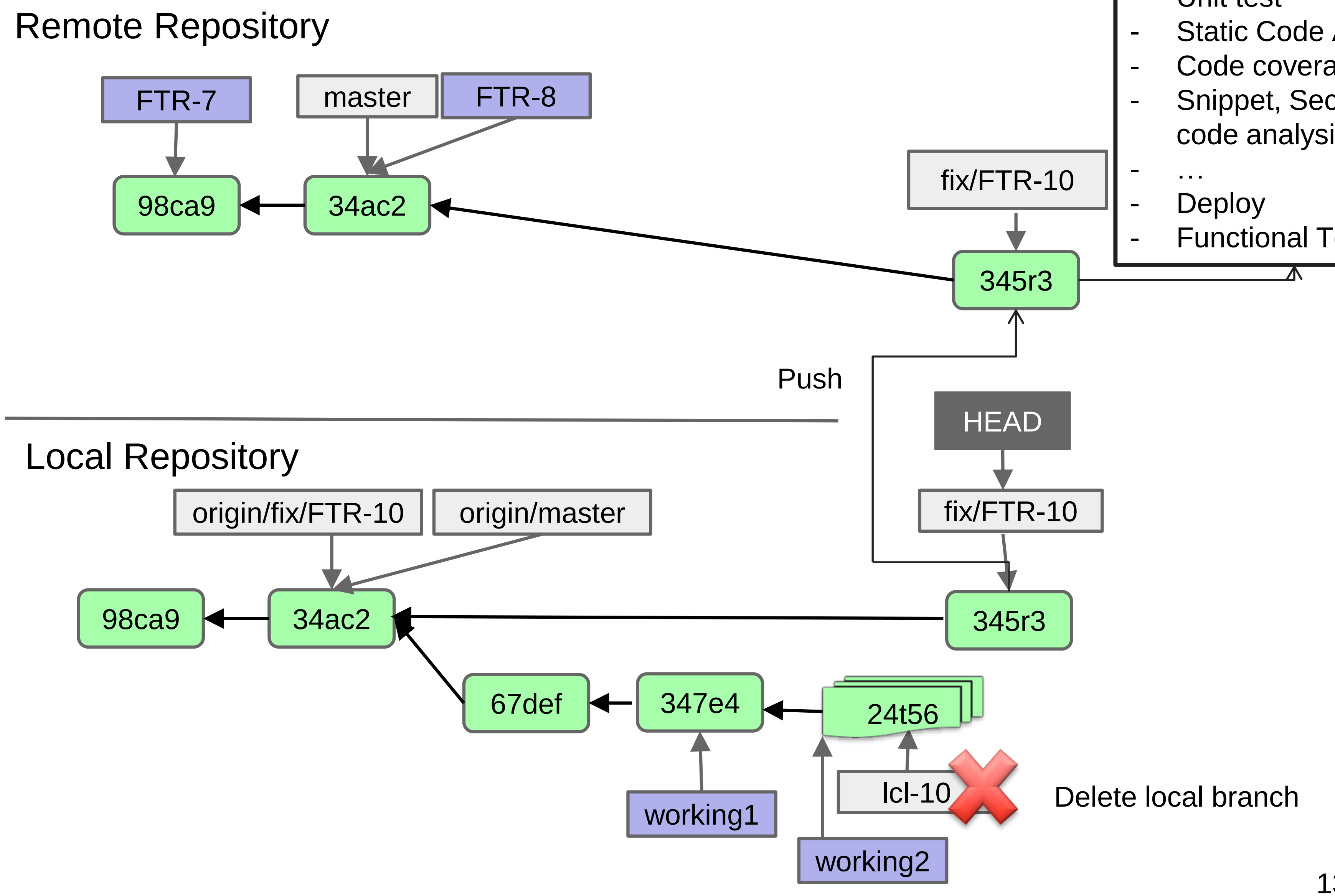
Merge --squash lcl-10 to fix/FTR-10

git checkout fix/FTR-10
git merge –squash lcl-10
git commit
Squash merge to 1 commit in FTR-10 branch

(Possibly Rebase master)

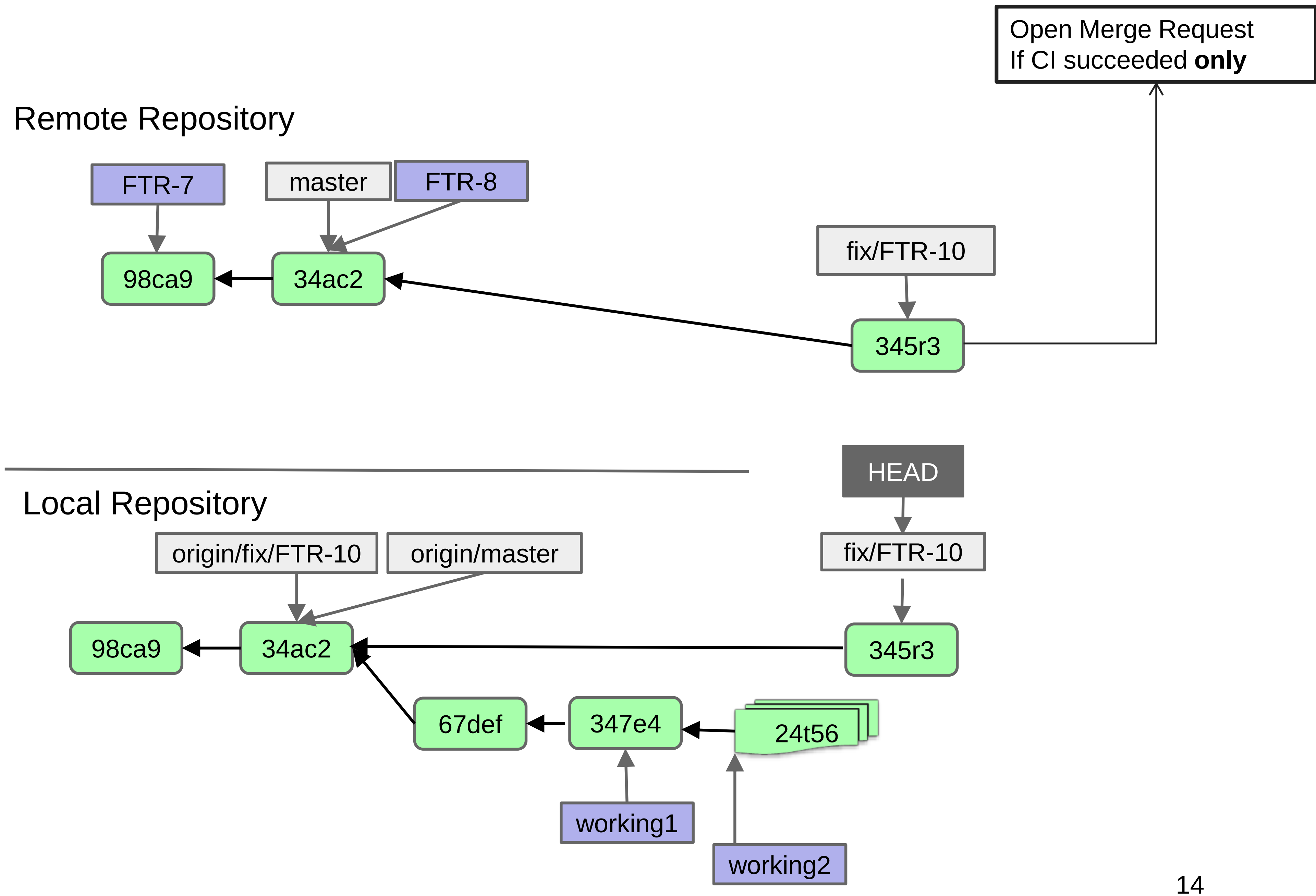


Push -> trigger CI, remove branch lcl-10

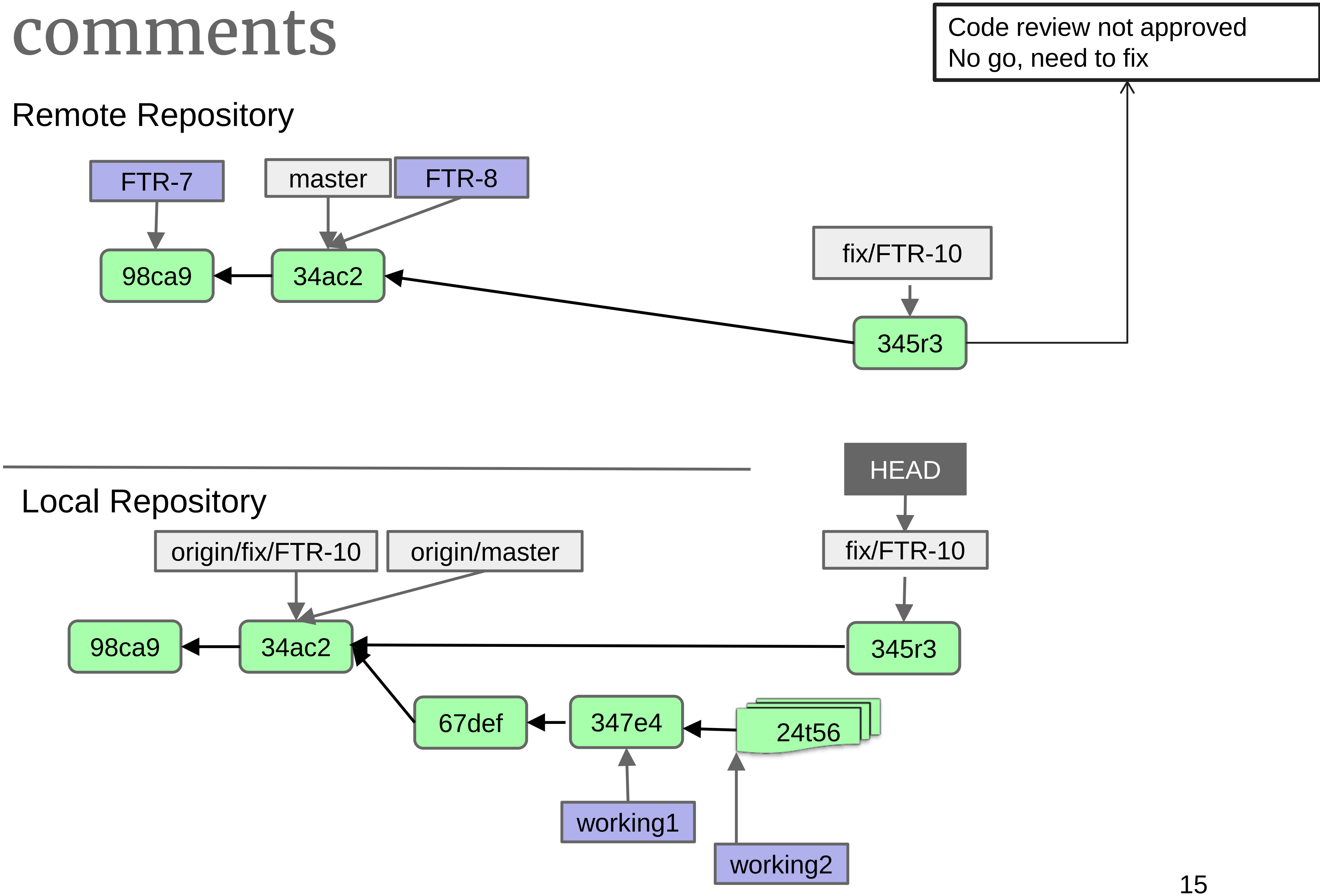


Push triggers CI pipeline:
- Build
- Unit test
- Static Code Analysis
- Code coverage test
- Snippet, Security code analysis
- …
- Deploy
- Functional Test

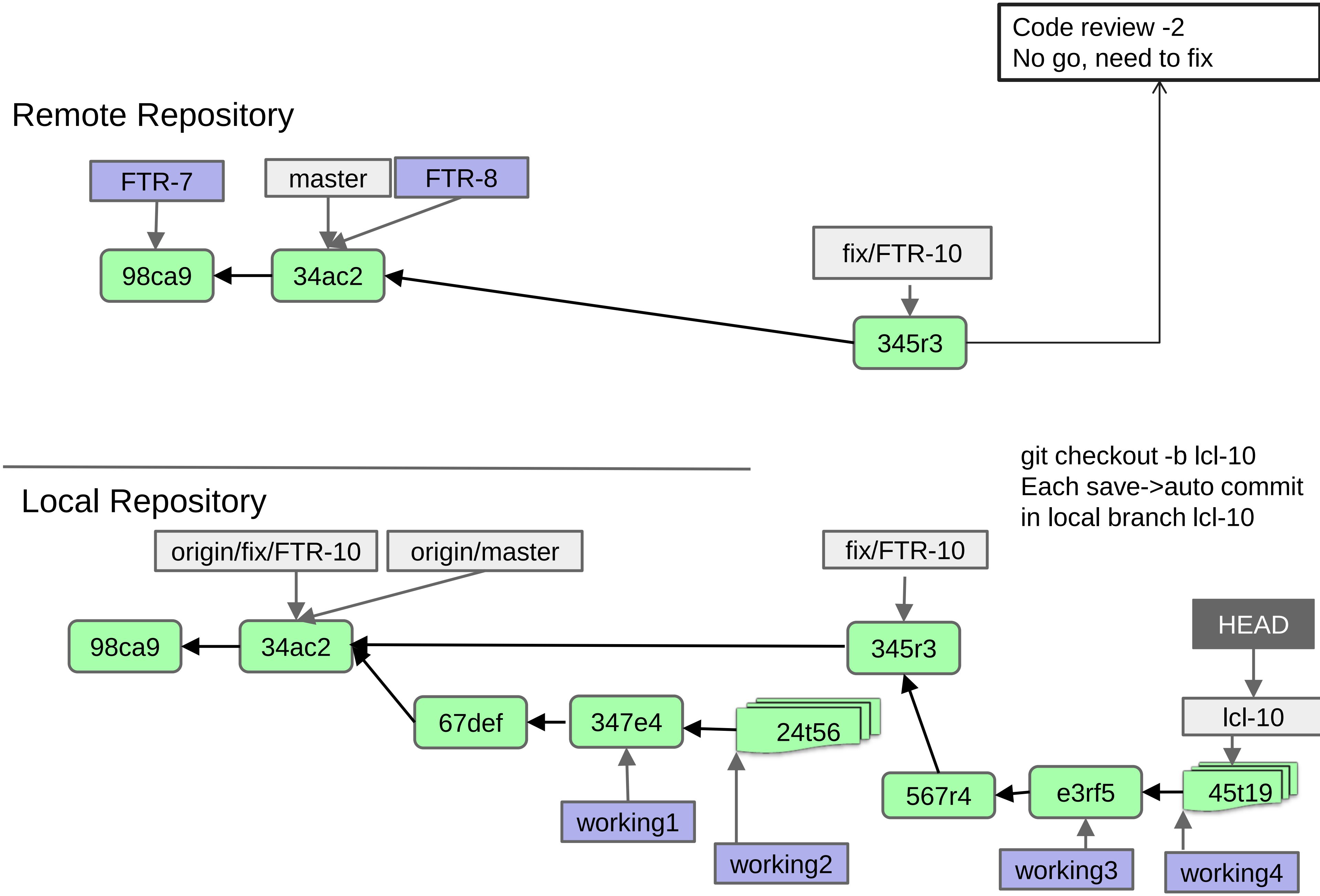
Open Gitlab MR



Change not approved, need to fix comments



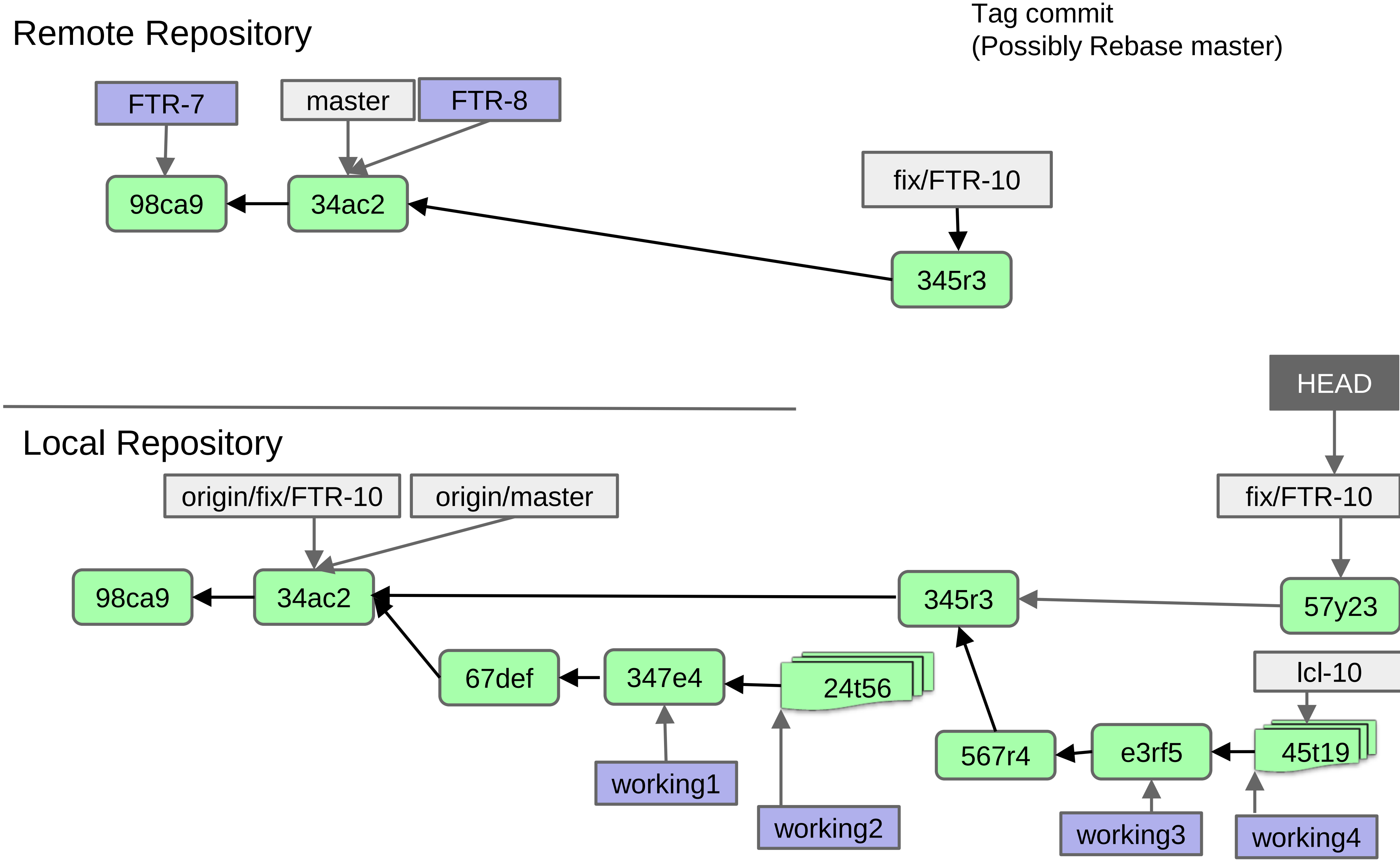
Again checkout lcl-10,save->autocommit



git checkout -b lcl-10
Each save->auto commit
in local branch lcl-10

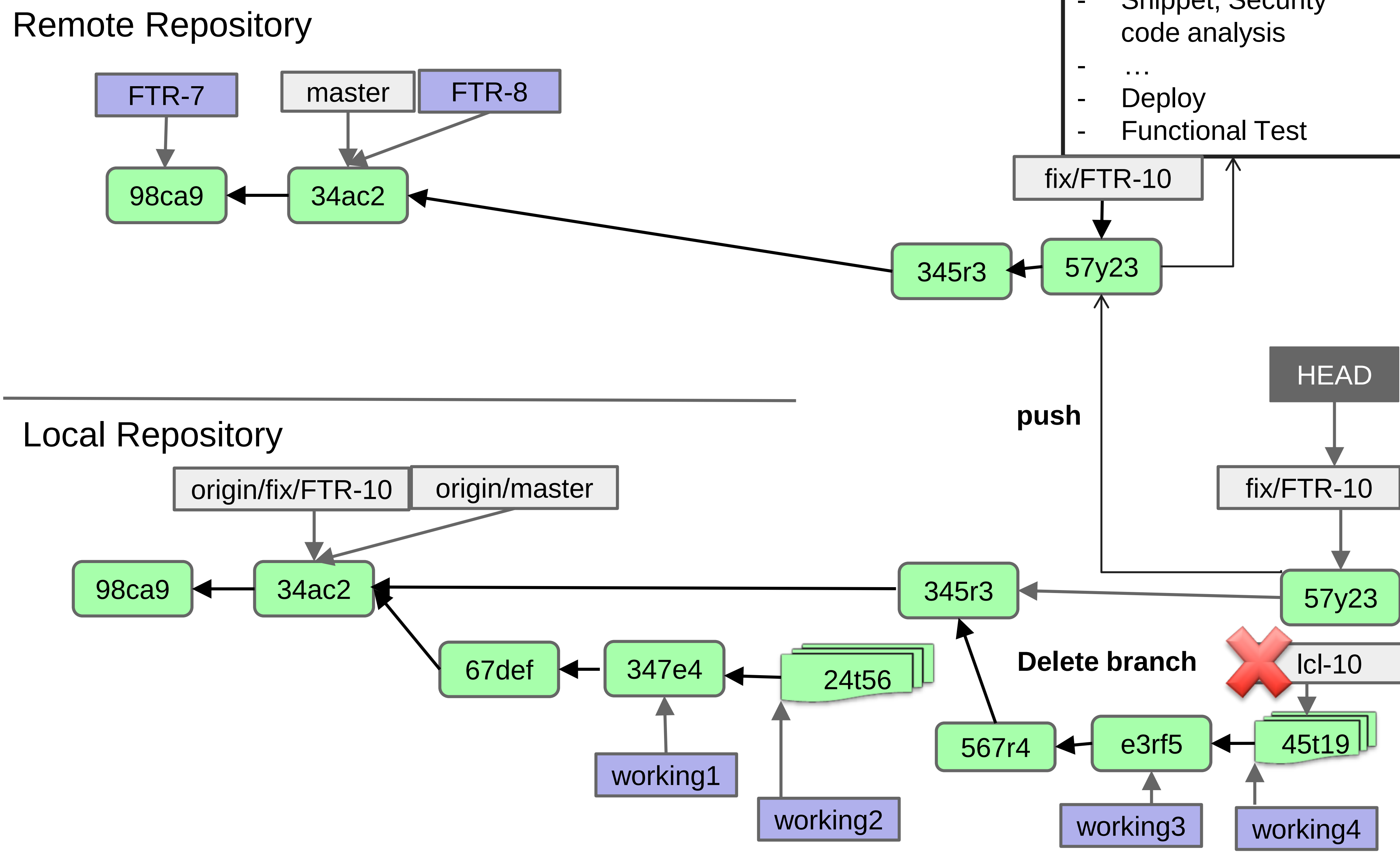
Squash merge to FTR-10

Squash merge to 1 commit in FTR-10 branch
Tag commit
(Possibly Rebase master)

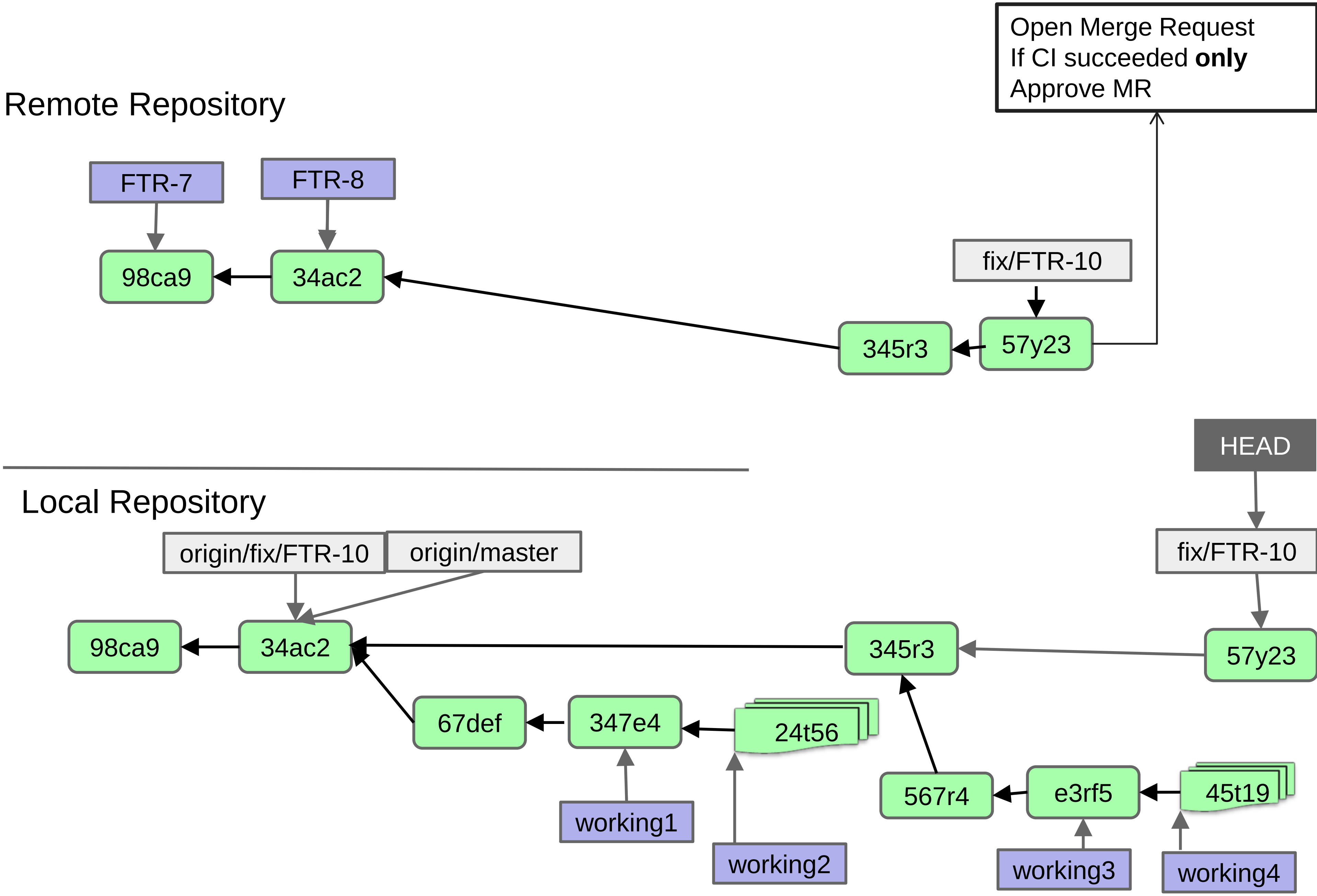


Push fix of review, CI, delete local branch lcl-10

Push triggers CI pipeline:
- Build
- Unit test
- Static Code Analysis
- Code coverage test
- Snippet, Security code analysis
- …
- Deploy
- Functional Test

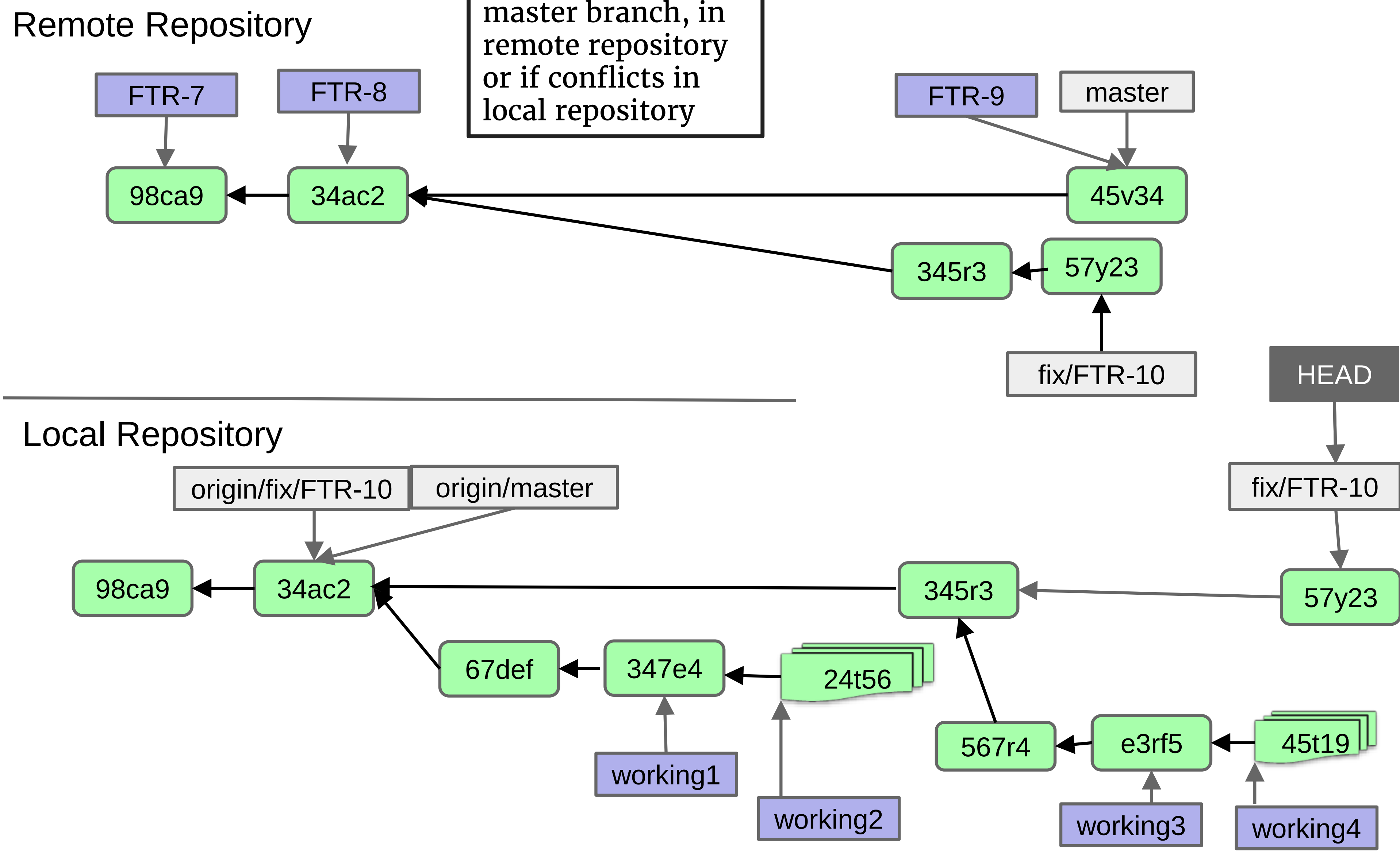


Open Merge Request, Approve MR

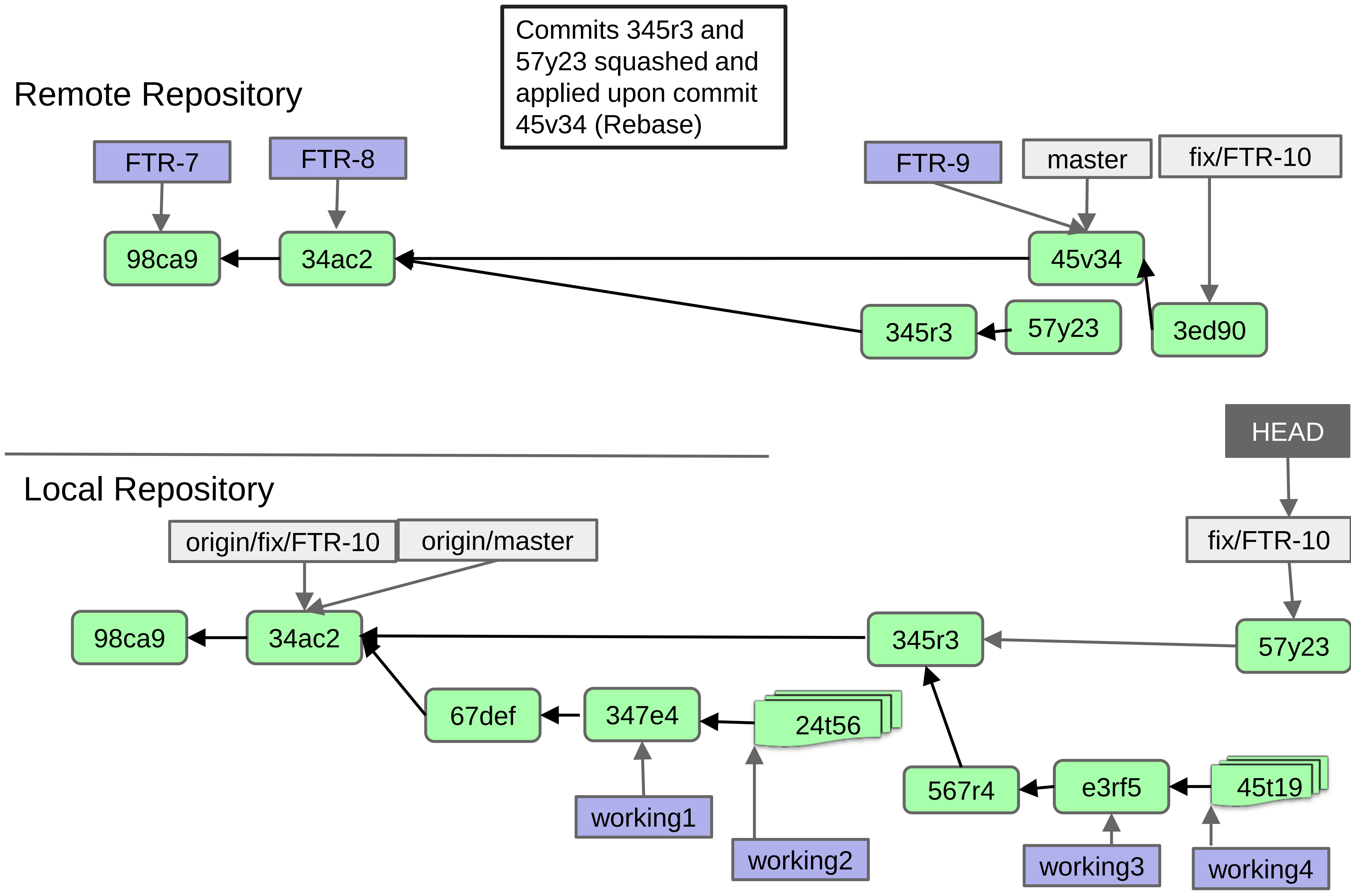


Other commit was merged, FF merge no possible now

Need to rebase feature branch on master branch, in remote repository or if conflicts in local repository

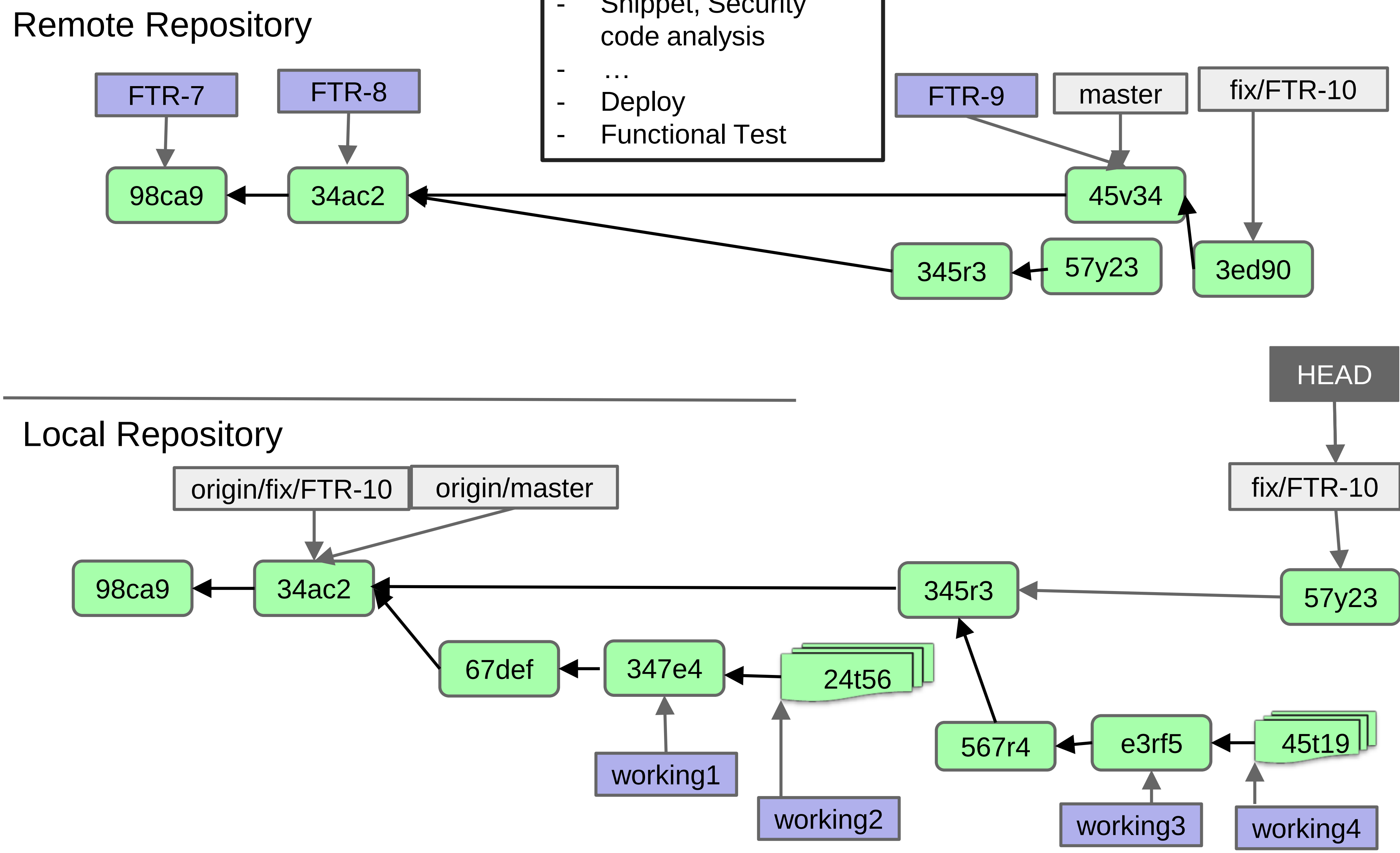


Rebase with squash **only**

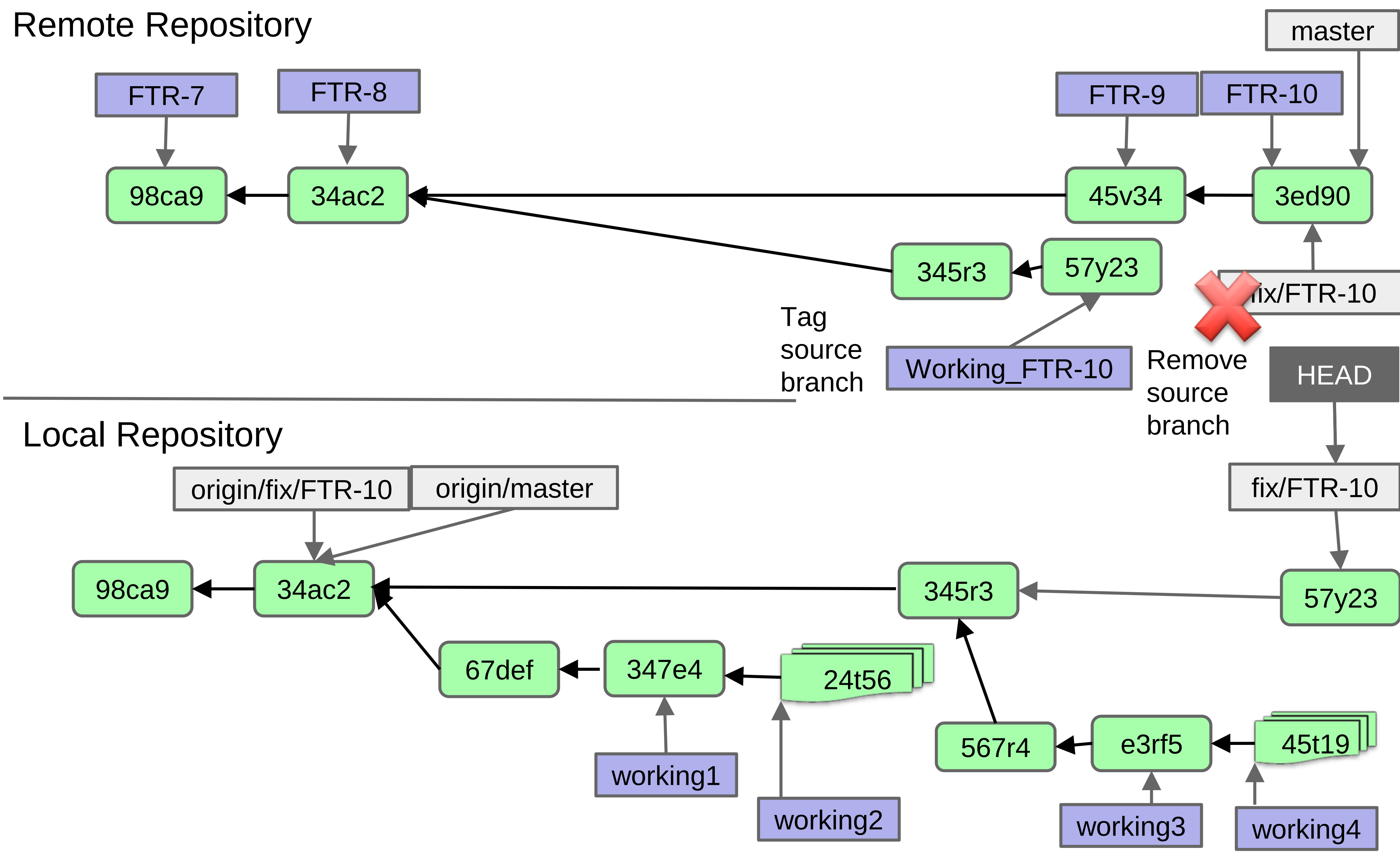


Trigger CI on new commit

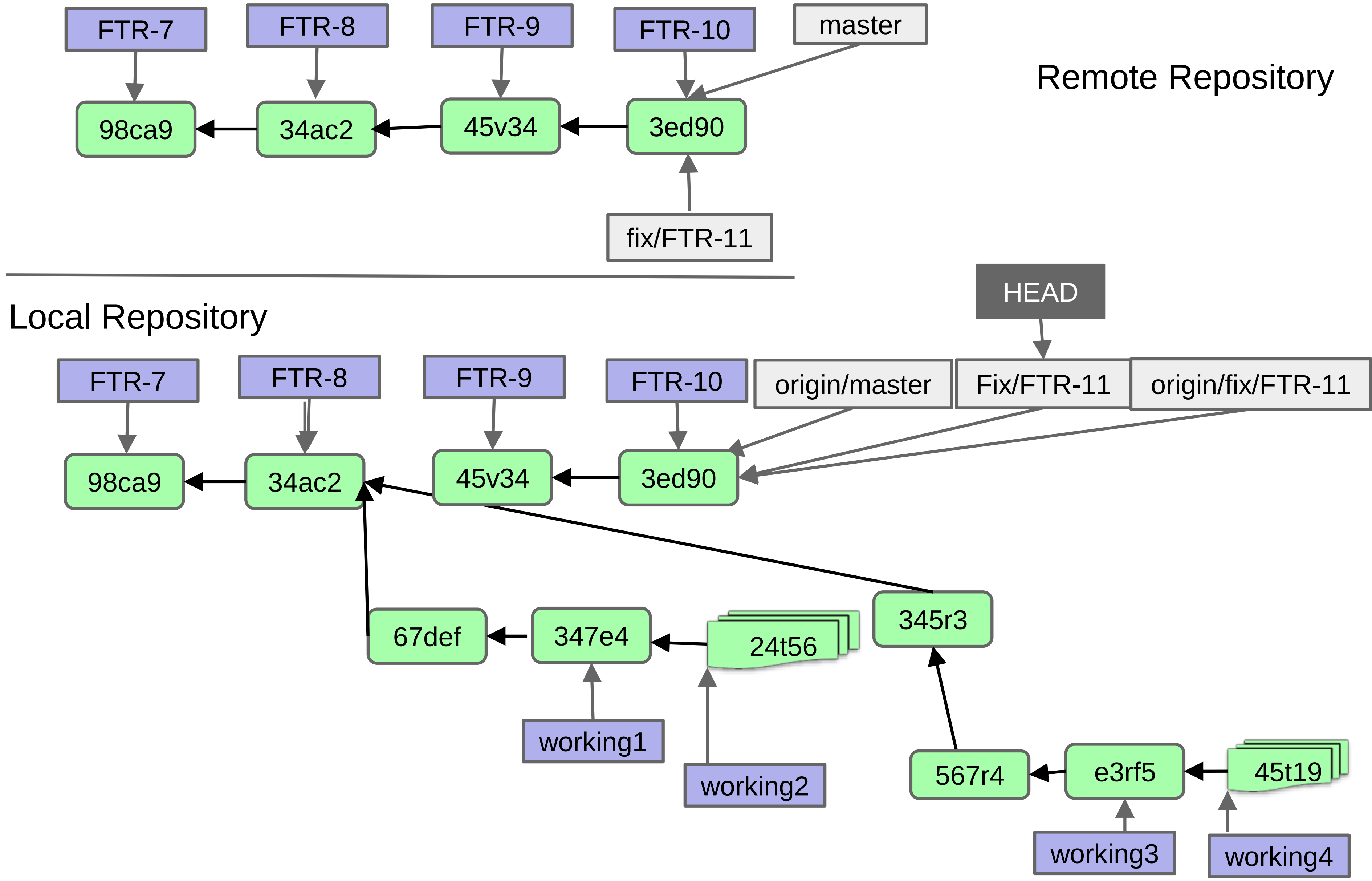
New commit triggers CI
- Build
- Unit test
- Static Code Analysis
- Code Coverage Test
- Snippet, Security code analysis
- …
- Deploy
- Functional Test



FF Merge, Tag and Remove Source Branch, Fast Forward Merge must



We got desired result: 1 Commit per Jira issue in main branch. Start new Feature FTR-11



2. Git + GitLab best practices hands on

https://gitlab.com/ilyaro/git_gitlab_best_practices

https://gitlab.com/ilyaro/git_gitlab_best_practices/-/blob/master/Commands.md

1.Work with local feature branches (topics)

Create your own feature branch from master: fix/<Jira-task> here: https://gitlab.com/ilyaro/demo-be

git clone -b fix/<Jira-task> https://gitlab.com/ilyaro/demo-be

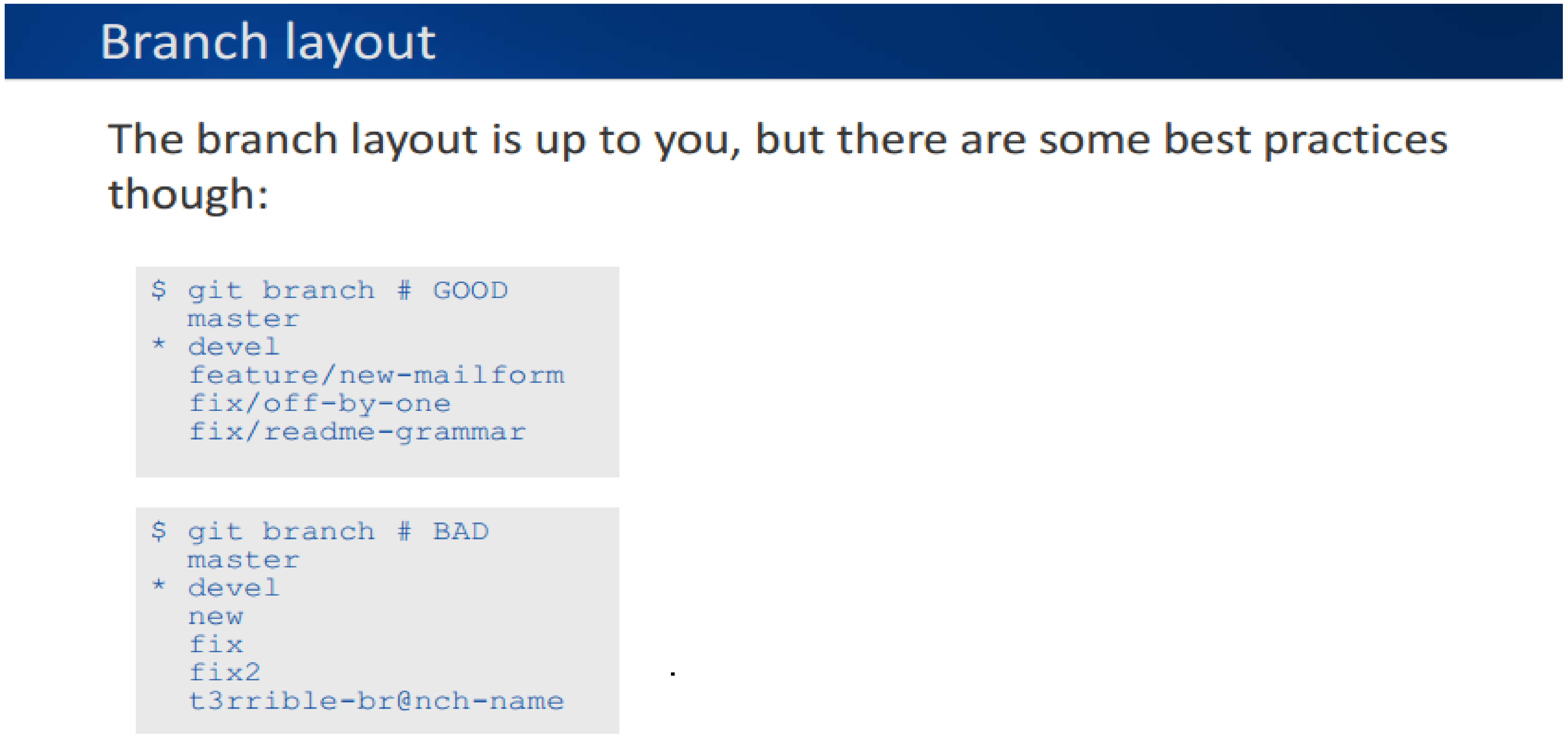
git checkout lcl-<task>

Configure IDE/Editor to auto commit: https://gitlab.com/ilyaro/git_best_practices_ppt/README.md

git log - to show initial status

2. Edit .\demo-be\src\main\java\com\example\dep\Dep.java (Use IntelliJ, VS Code, vim or any other IDE,

Best practices – Branch Layout



Branch layout

The branch layout is up to you, but there are some best practices though:

```
$ git branch # GOOD
  master
* devel
  feature/new-mailform
  fix/off-by-one
  fix/readme-grammar
```

```
$ git branch # BAD
  master
* devel
  new
  fix
  fix2
  t3rrible-br@nch-name
```

Git + GitLab best practices hands on cont.

after configure on save trigger)

Change 1 line – save

Add 1 line – save

Delete 1 line – save

It will create 3 commits

3 Tag important commit (working commits)

```
git tag working1
```

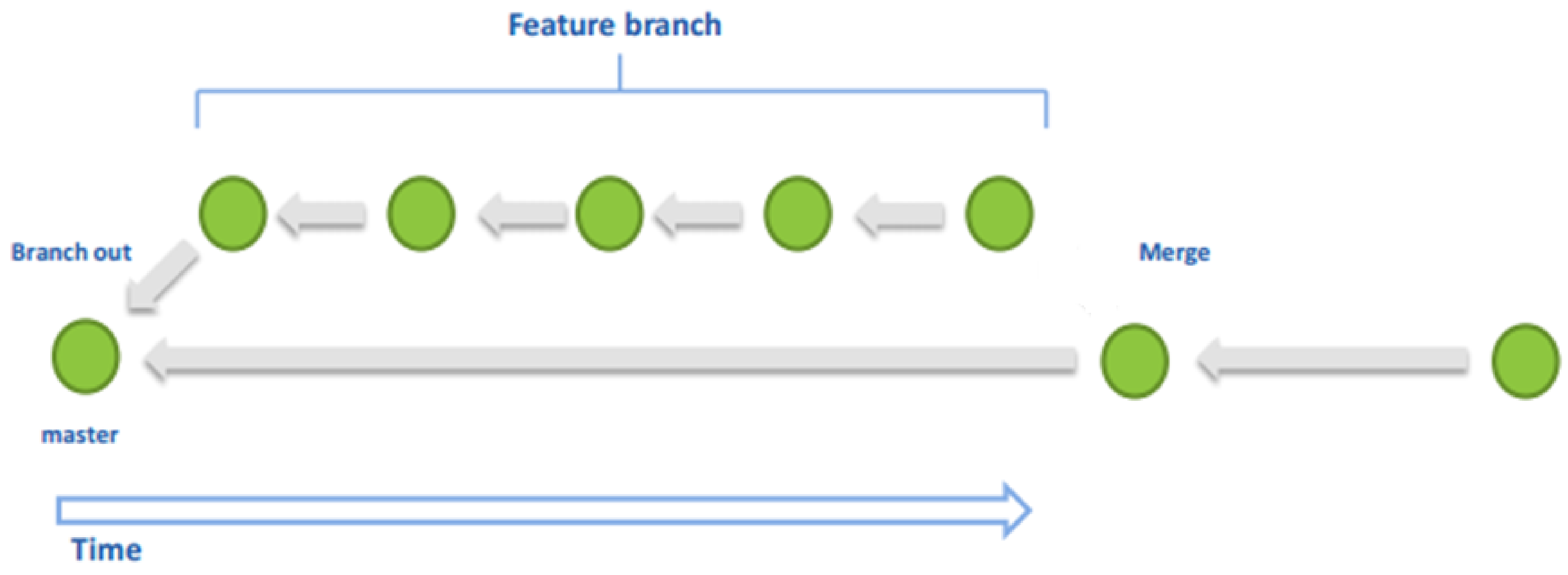
4 Log with diff with only change, hash and refs

```
git log -c -U0 --pretty="format:%H%d“
```

Or in GUI

Best practices – local feature branches (topics)

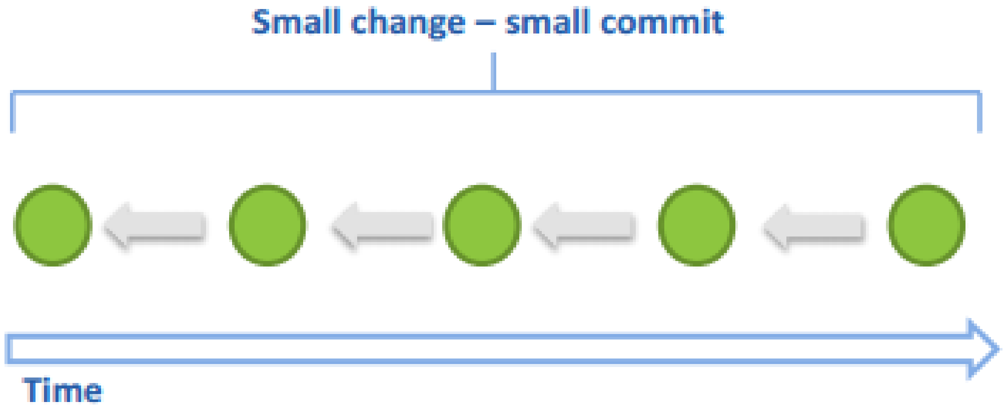
Work on feature branches locally



Demo: create feature branch, checkout

Best practices – commit

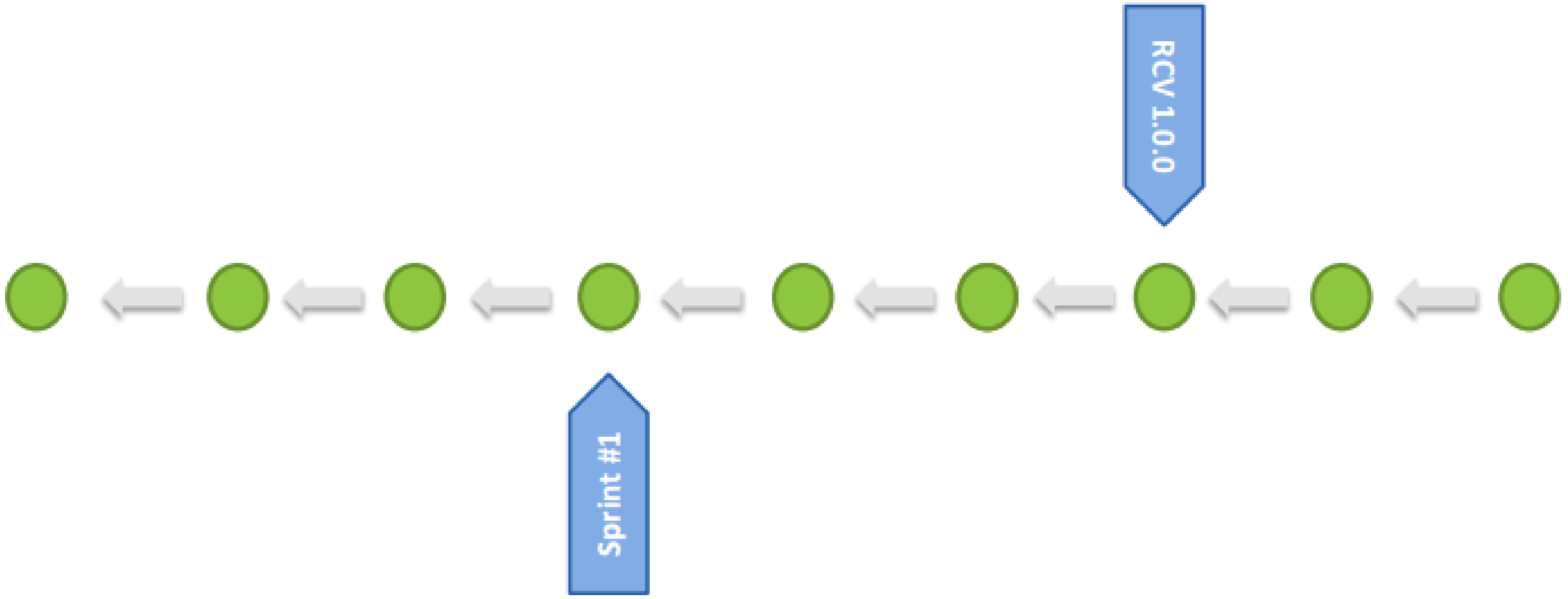
Keep changes small wherever possible and commit frequently



Demo: vim, change, save -> auto-commit, git log –c, git reset

Best practices – tag milestones (even local)

Tag important milestones (for history and for accessibility)



Git + GitLab best practices hands on cont.

5. Squash in merge from feature branch to master

See what to merge with git log --graph or IDE GUI

git log --oneline --graph –all

checkout feature branch

git checkout fix/<Jira-task>

merge --squash from topic/demo_gh

git merge --squash lcl-<task>

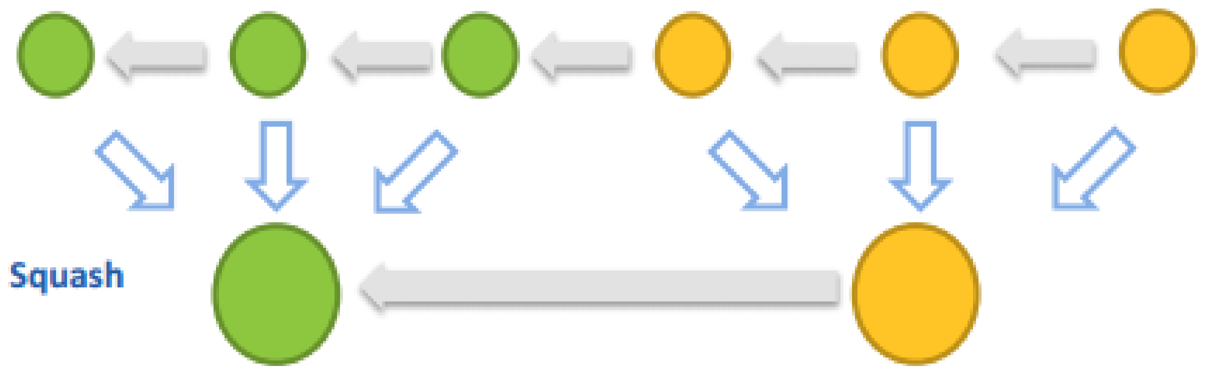
now commit 1 squashed commit to the feature branch: Give meaningfull and concise commit message:

Example: "Testing auto commit on save and squash merge "Specify Jira-task in commit message body

git commit

Best practices – squashing before push

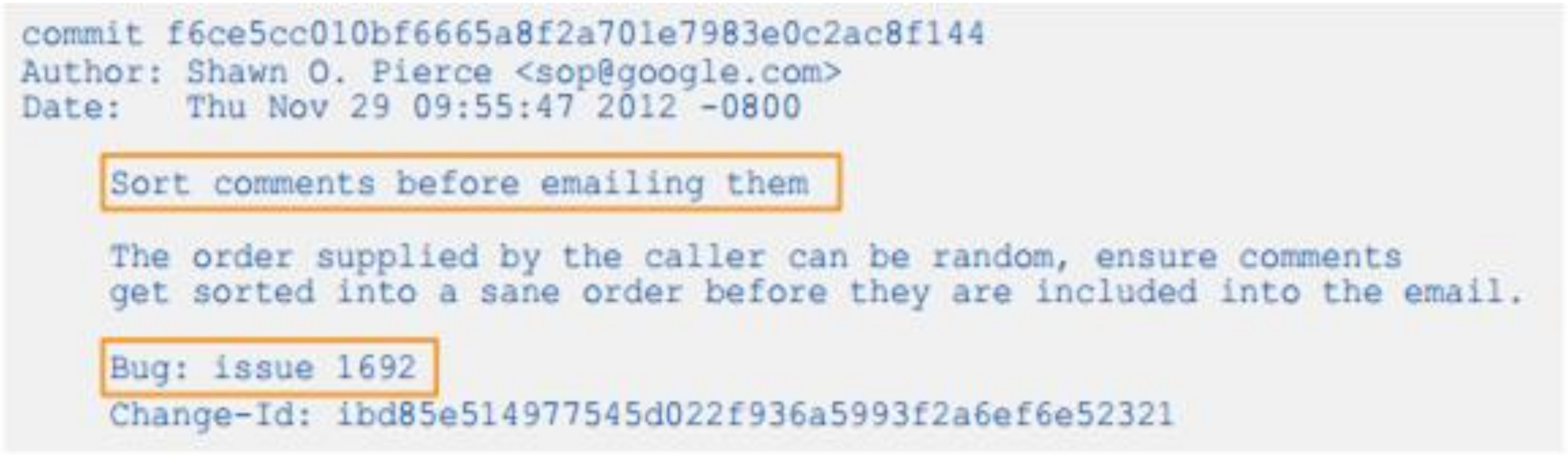
Before pushing, squash related changes together to make for better understanding by others



Demo: git merge --squash, git commit,
git log --oneline --graph --all, git push

Best practices - concise commit messages

Limit commit message Subject to 70 chars and it must be simple, clear, self explained – this is the line all will see!
Meaningful details put in the body of the commit message



```
commit f6ce5cc010bf6665a8f2a701e7983e0c2ac8f144
Author: Shawn O. Pierce <sop@google.com>
Date:   Thu Nov 29 09:55:47 2012 -0800

    Sort comments before emailing them

    The order supplied by the caller can be random, ensure comments
    get sorted into a sane order before they are included into the email.

    Bug: issue 1692
    Change-Id: ibd85e514977545d022f936a5993f2a6ef6e52321
```

Demo: Example:

https://github.com/datreeio/node-datreeio/commits/master

Git + GitLab best practices hands on cont.

6. Delete local branch

git branch -D lcl-<Jira-task>

See log again

git log --oneline --graph --all

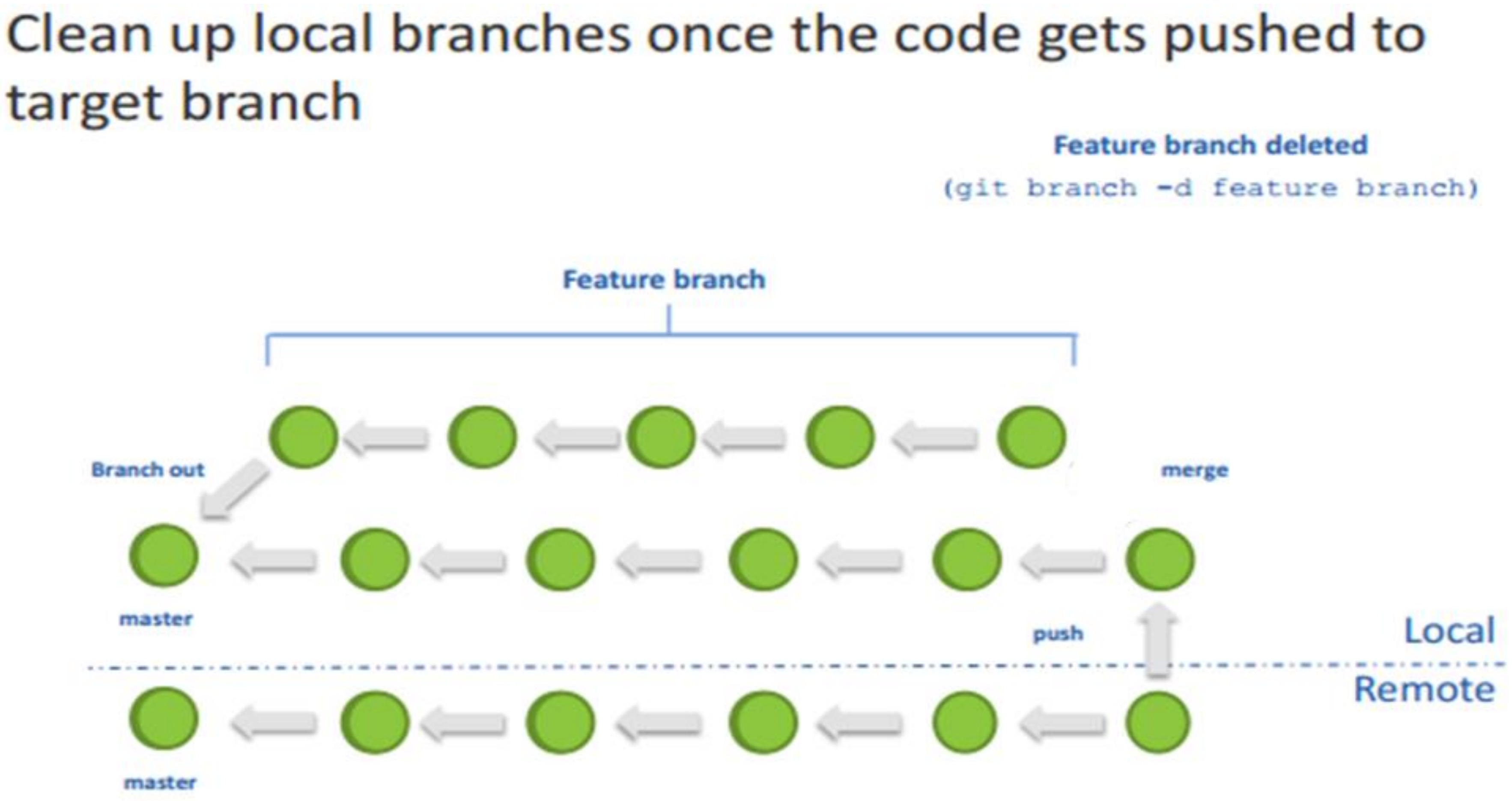
7. Push to Gitlab Feature branch

git push

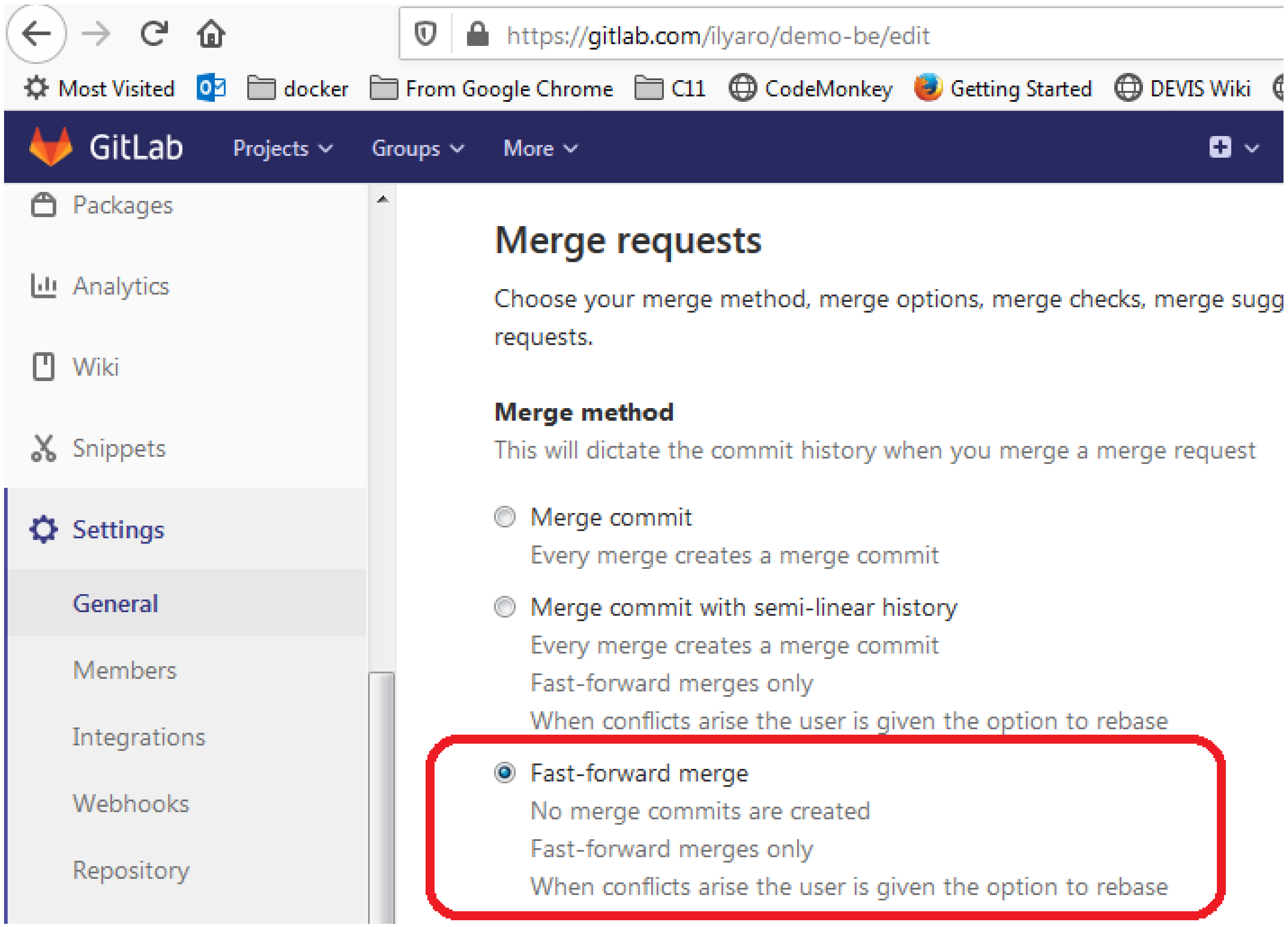
See Gitlab CI piopeline running:

https://gitlab.com/ilyaro/demo-be/pipelines

Best practices – clean up local branches


Clean up local branches once the code gets pushed to target branch

3. GitLab Best practices
Fast-forward Merge in Gitlab



Merge requests

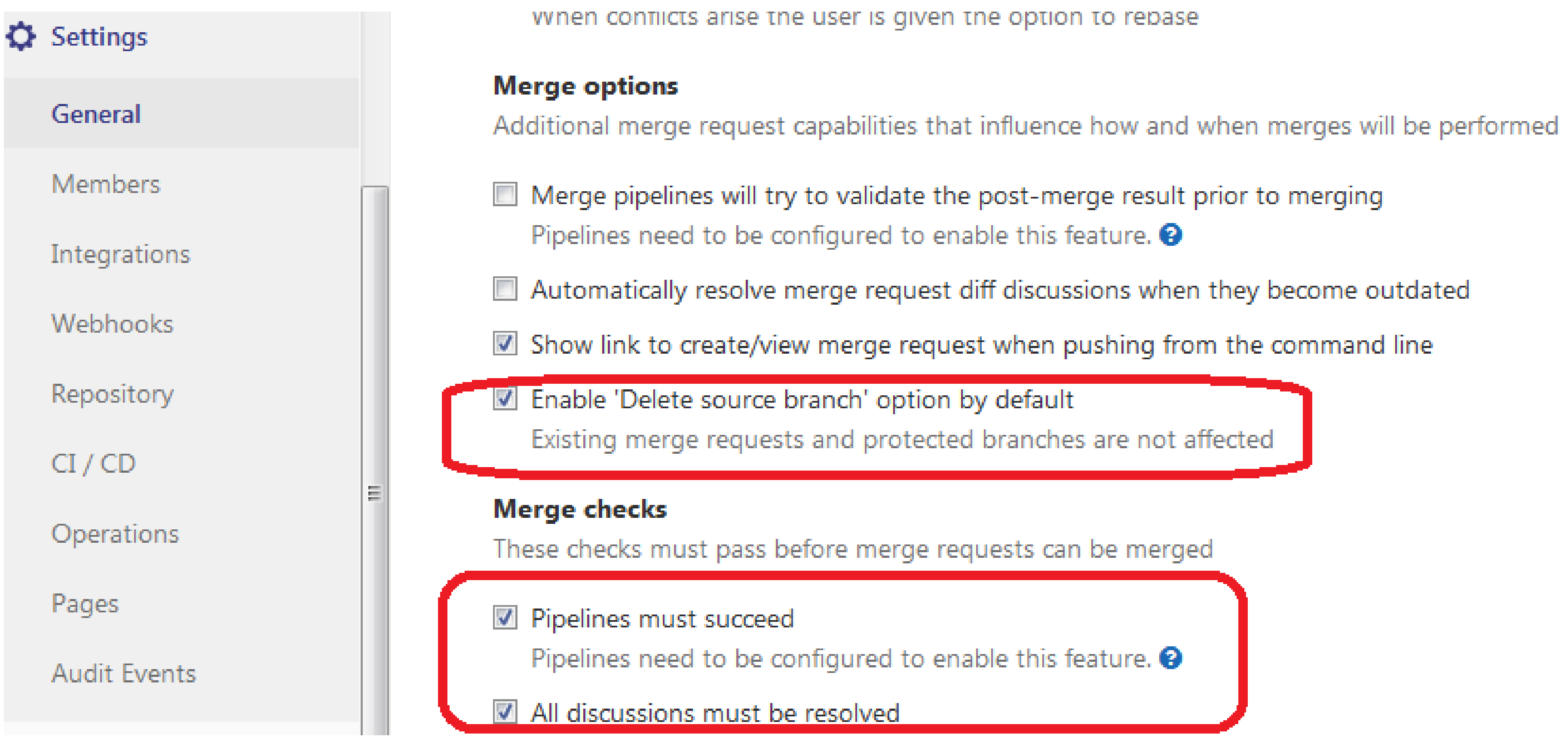
Choose your merge method, merge options, merge checks, merge sugg
requests.

Merge method

This will dictate the commit history when you merge a merge request

- ○ Merge commit
  Every merge creates a merge commit
- ○ Merge commit with semi-linear history
  Every merge creates a merge commit
  Fast-forward merges only
  When conflicts arise the user is given the option to rebase
- [✓ Fast-forward merge]
  No merge commits are created
  Fast-forward merges only
  When conflicts arise the user is given the option to rebase

3. GitLab Best practices cont.
Delete source branch, pipeline, discussions



When conflicts arise the user is given the option to rebase

Merge options

Additional merge request capabilities that influence how and when merges will be performed

- [ ] Merge pipelines will try to validate the post-merge result prior to merging
  Pipelines need to be configured to enable this feature.
- [ ] Automatically resolve merge request diff discussions when they become outdated
- [x] Show link to create/view merge request when pushing from the command line
- [x] Enable 'Delete source branch' option by default
  Existing merge requests and protected branches are not affected

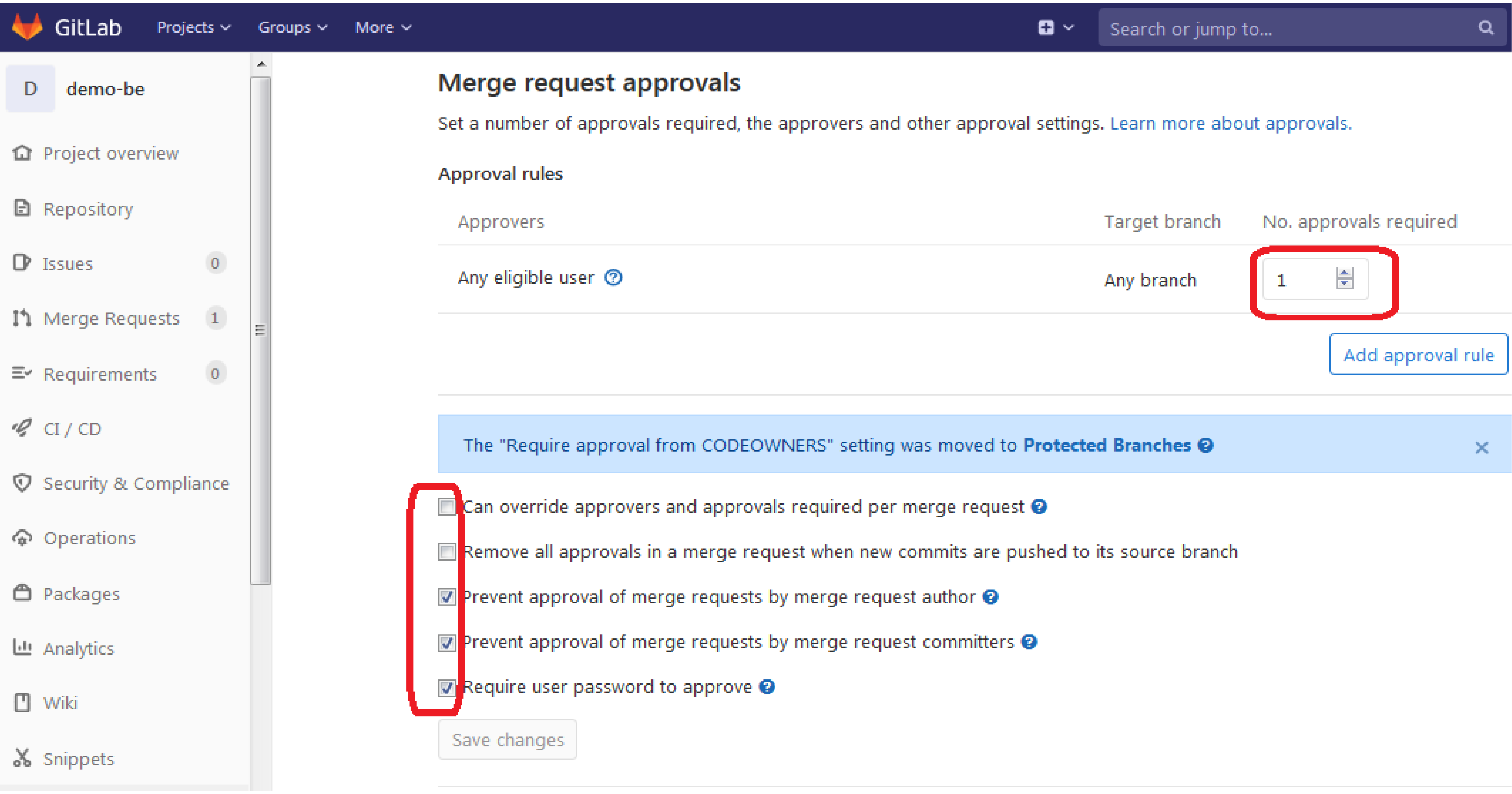
Merge checks

These checks must pass before merge requests can be merged

- [x] Pipelines must succeed
  Pipelines need to be configured to enable this feature.
- [x] All discussions must be resolved

3. GitLab Best practices

Must at least 1 approver, not author and not committer.



Merge request approvals

Set a number of approvals required, the approvers and other approval settings. Learn more about approvals.

Approval rules

Approvers	Target branch	No. approvals required
Any eligible user	Any branch	1

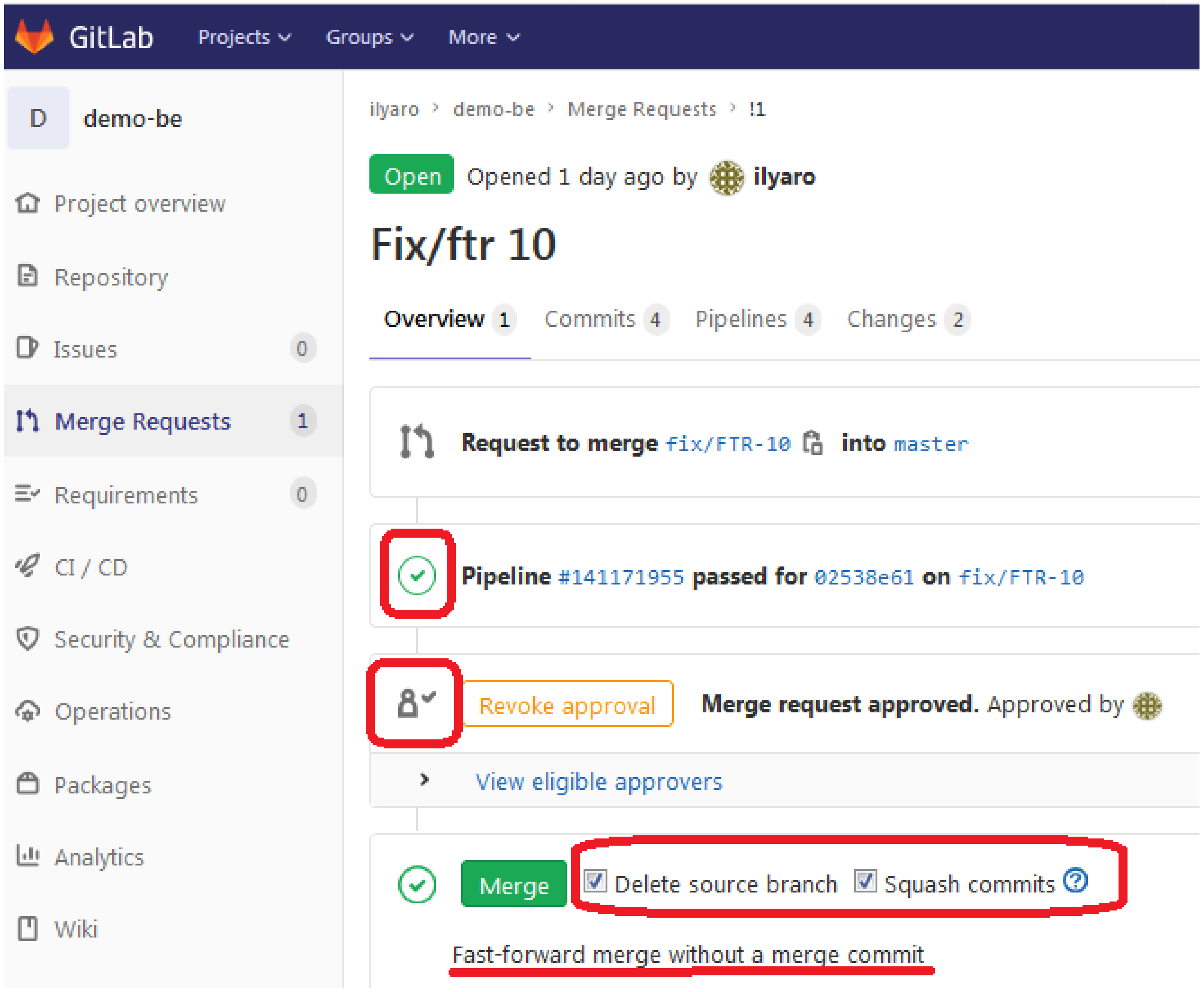
Add approval rule

The "Require approval from CODEOWNERS" setting was moved to **Protected Branches**

- [ ] Can override approvers and approvals required per merge request
- [ ] Remove all approvals in a merge request when new commits are pushed to its source branch
- [x] Prevent approval of merge requests by merge request author
- [x] Prevent approval of merge requests by merge request committers
- [x] Require user password to approve

Save changes

FF Merge only after: CI Pipeline succeed and Code Review + Approval. Squash + Delete Source Branch



Opened 1 day ago by ilyaro

Fix/ftr 10

Request to merge fix/FTR-10 into master

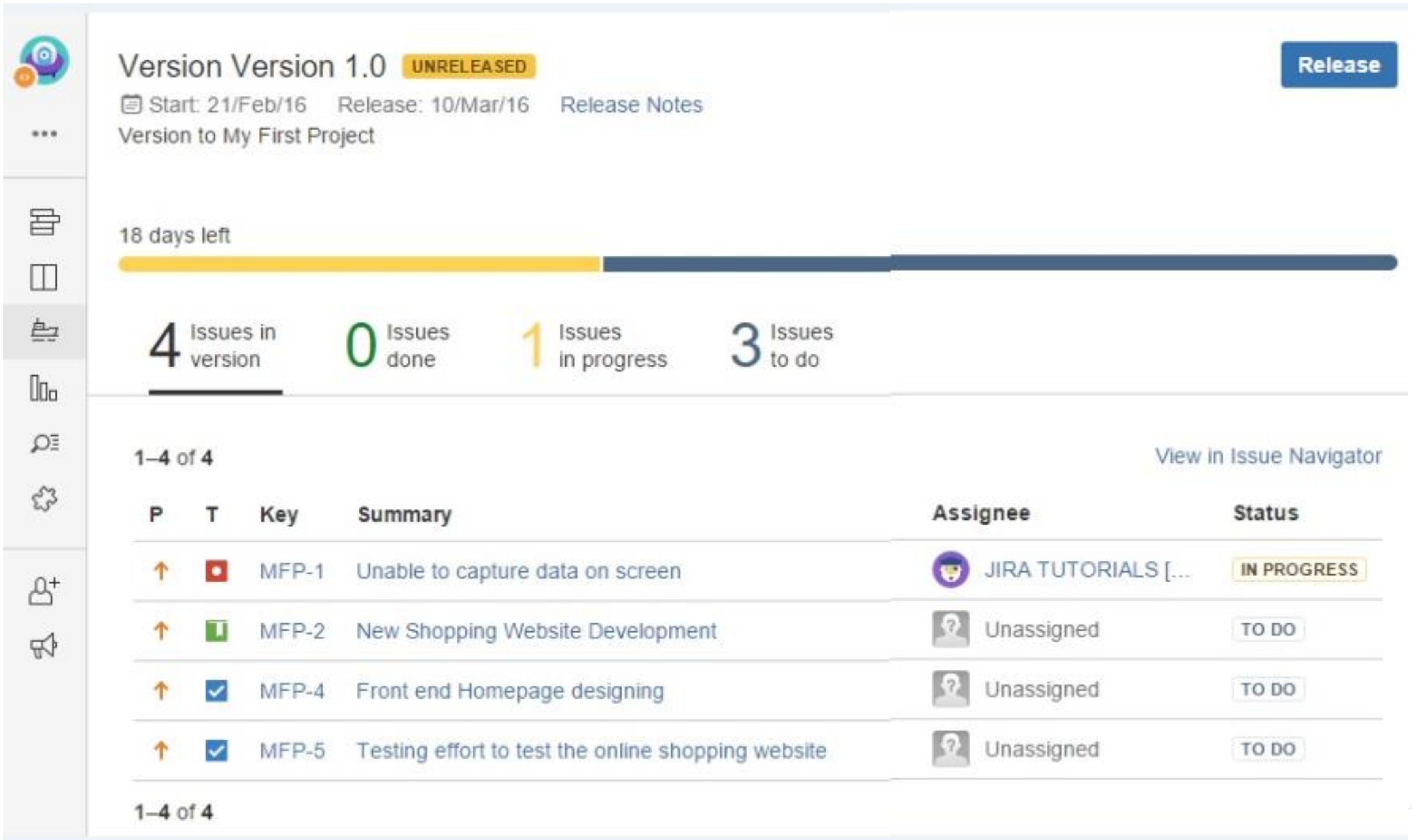
Pipeline #141171955 passed for 02538e61 on fix/FTR-10

Merge request approved. Approved by

☑ Delete source branch ☑ Squash commits

Fast-forward merge without a merge commit

Gitlab FF Merge must be related to Jira Version Release. Only tasks in Version must be merged.



Version Version 1.0 UNRELEASED

Start: 21/Feb/16 Release: 10/Mar/16 Release Notes

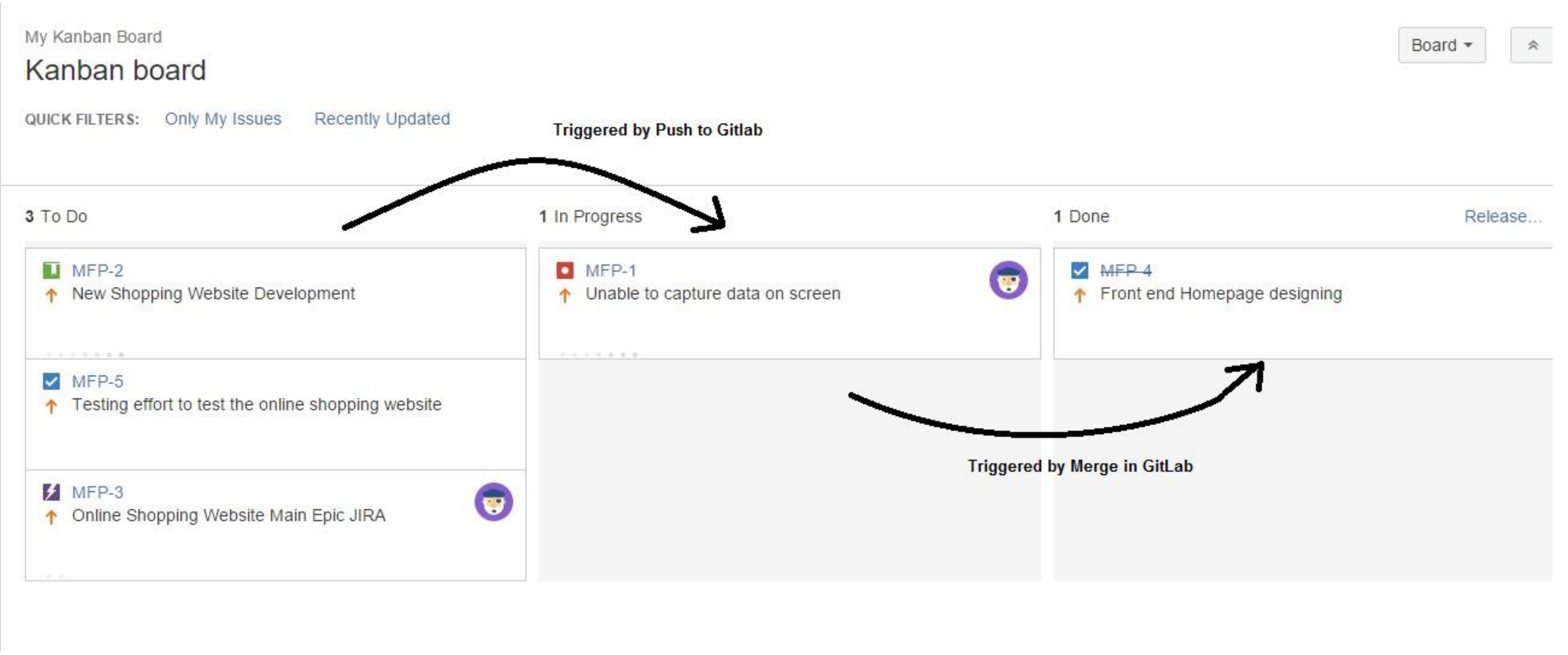
Version to My First Project

1–4 of 4

P	T	Key	Summary	Assignee	Status
↑	■	MFP-1	Unable to capture data on screen	JIRA TUTORIALS [...	IN PROGRESS
↑	■	MFP-2	New Shopping Website Development	Unassigned	TO DO
↑	✓	MFP-4	Front end Homepage designing	Unassigned	TO DO
↑	✓	MFP-5	Testing effort to test the online shopping website	Unassigned	TO DO

1–4 of 4

Merge need to change status to “Done”, Approve to status “Code Review”, Push – to status “In Progress”



My Kanban Board

Kanban board

QUICK FILTERS: Only My Issues Recently Updated

3 To Do	1 In Progress	1 Done
MFP-2 New Shopping Website Development	MFP-1 Unable to capture data on screen	~~MFP-4~~ Front end Homepage designing
MFP-5 Testing effort to test the online shopping website		
MFP-3 Online Shopping Website Main Epic JIRA		

3. GitLab Best practices cont.
GitLab CI best practices. DevOps templates

Push triggers CI/CD pipeline (**topic/feature branch**):

- Build
- Unit test
- Static Code Analysis
- Code coverage test
- Snippet, Security code analysis
- …
- STG Deploy
- Functional Tests
- Regression Tests

Fast Forward Merge to **master branch**, triggers CI pipeline:

- PRD Deploy (Automatic/Manual)
- Functional Test
- Regression Tests

4. Git Lab CI overview

- CI reference Guide
  https://docs.gitlab.com/ee/ci/yaml/
- CI definition file
  https://gitlab.com/ilyaro/demo-be/-/blob/master/.gitlab-ci.yml
- Pipelines
  https://gitlab.com/ilyaro/demo-be/pipelines

Questions?

Summary

Ilya Rokhkin (Git Trainings)

https://github.com/ilyaro/git_best_practices_ppt

rokhkin_ilya@yahoo.com

054-5224805

Git GitLab Best Practices

Thank You!