... in the Internet of Things

Bachelor Project (PO)
Git version control
Hamburg 04.04.2022

José Álamos jose.alamos@haw-hamburg.de
Peter Kietzmann peter.kietzmann@haw-hamburg.de
Leandro Lanzieri leandro.lanzieri@haw-hamburg.de

Prof. Dr. Thomas C. Schmidt INET AG, Dept. Informatik HAW Hamburg t.schmidt@haw-hamburg.de

Git Version Control

What is Git?

- Distributed version control system
- It allows
  - Storing content (source code, documentation)
  - Snapshots: keeping track of changes
  - Sharing code between developers
- Each developer stores its own full copy of the code (repository).

Git basics: commit

- Incremental set of changes
- Consists of:
  - A commit hash
  - Metadata (commit name, author, email)
  - List of changes
    - “Add FOO in file bar.c, after line 93”
    - “Remove BUZ from file foo,c, line 3”
  - A parent commit

Git basics: commit

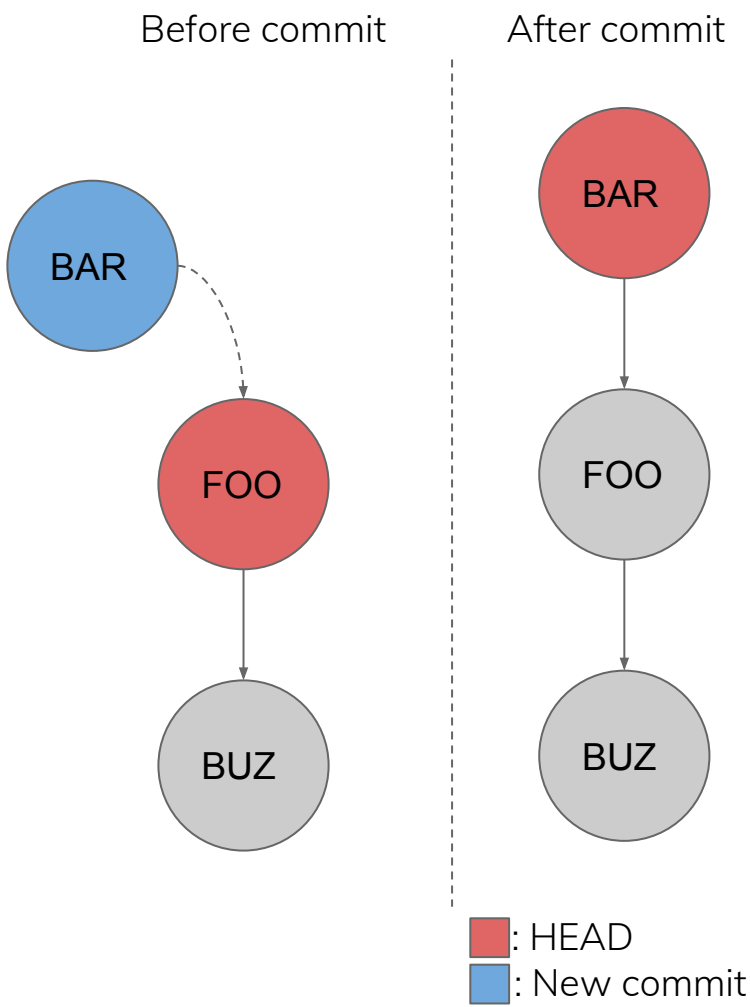
```
commit e0b433d2a0e5e343bd9d85bed6327bd8ba341441 (HEAD -> copr/11237)
Author: Jose Alamos <jose@alamos.cc>
Date:   Mon Mar 21 15:28:57 2022 +0100

   My commit

diff --git a/examples/hello-world/main.c b/examples/hello-world/main.c
index f51bf8c0a0..186a39dada 100644
--- a/examples/hello-world/main.c
+++ b/examples/hello-world/main.c
@@ -23,8 +23,7 @@
int main(void)
{
-   puts("Hello World!");
-
+   printf("FOO");
   printf("You are running RIOT on a(n) %s board.\n", RIOT_BOARD);
   printf("This board features a(n) %s MCU.\n", RIOT_MCU);
```

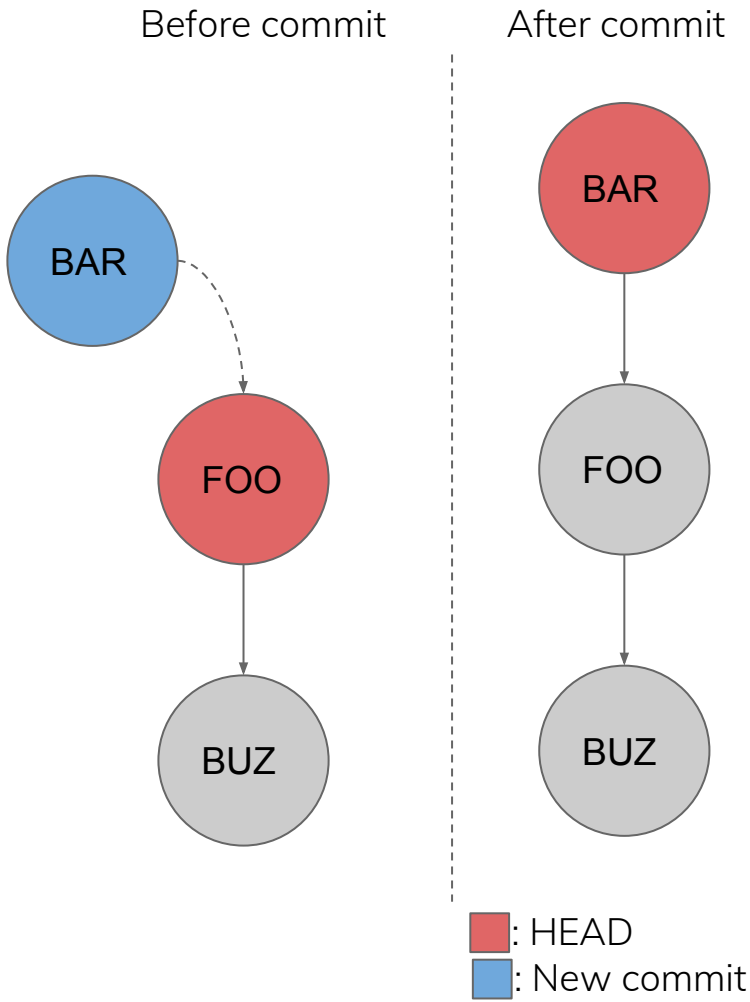
Git basics: Commit process

- Each commit links to its parent
  - It creates a linked lists of commits
- Git maintains a **HEAD** reference pointing to a (last) commit
- When a developer commits:
  - The commit parent points to **HEAD**
  - **HEAD** “updates” to the new commit



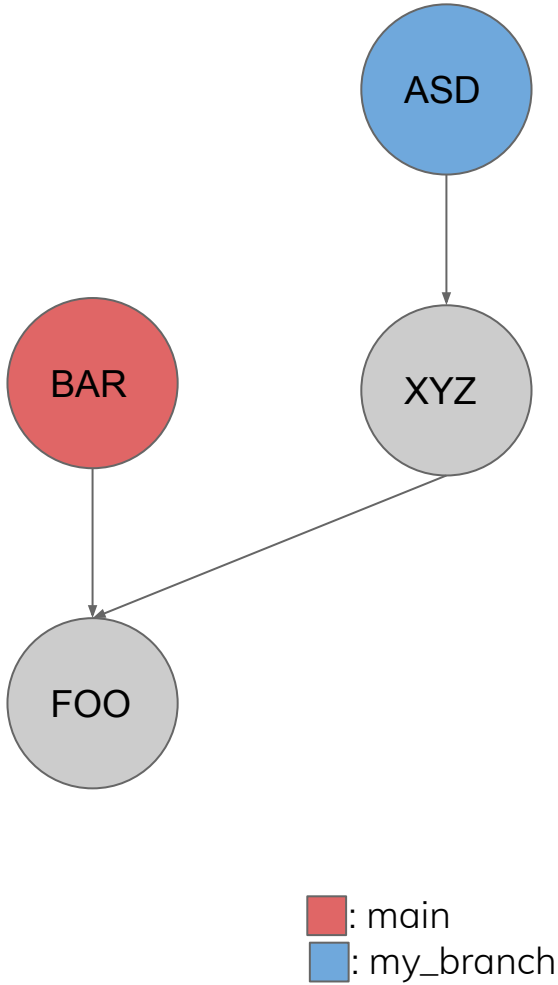
Git basics: Commit process

- E.g: “A developer commits a set of changes **BAR**”
  - **BAR** parent is **FOO** (Last **HEAD**)
  - The new **HEAD** points to **BAR**



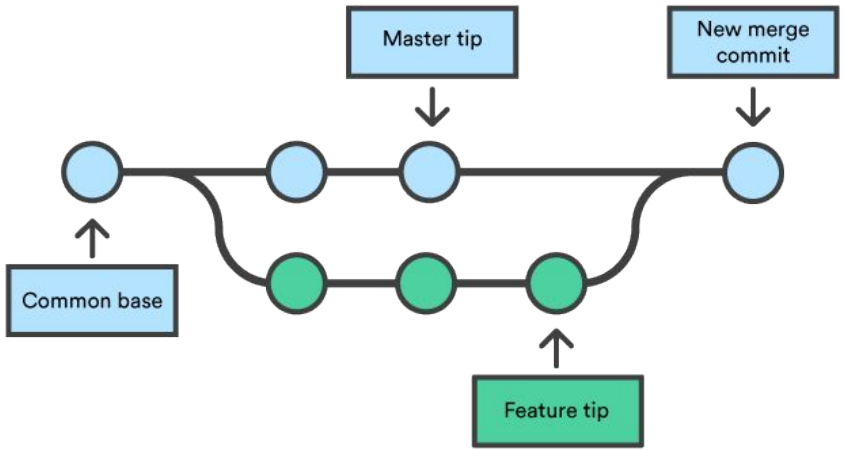
Git basics: branches

- We can change **HEAD** in order create branches.
  - Branches are pointers to a commit.
  - On commit, Git updates the branch to the new commit.
- We “checkout” a branch by pointing **HEAD** to its commit hash
- E.g: Checkout *my_branch*
  - **HEAD** points to **ASD**
- Of course, a developer can checkout any commit
  - Allows to go back in history



Git basics: merge

- Git provides a mechanism to merge two branches
    - Useful for developing new features without breaking the **main** branch



Source: Atlassian Git Tutorial

Git basics: push and pull

- So far we only talked about local repository changes...
- Developers can merge branches from other repositories (remote) with local branches (a.k.a “pull”)
- Similarly, developers can “push” a local branch to a remote repository to “update” a remote branch
- E.g Developer A wants to collaborate with Developer B , in the repository of Developer B at http://foo.bar/dev_b.git
    - Developer A adds http://foo.bar/dev_b.git as remote.
    - Developer A can pull a branch from developer B directly.
    - Of course, Developer A cannot push to the repository if it does not have enough privileges.

Git: a hands on example

- Fred Flintstone and Barney Rubble are two developers of the FlyBNB project with the project repository at http://thejetsons.com/flyBNB.git.
- Both create local copies of the repository in order to start development

  git clone http://thejetsons.com/flyBNB.git
- By default, Git adds an “origin” remote pointing to the repository.
- Fred wants to implement “Feat A” but does not want to break the **main** branch with on-going development. He branches **feat_a** from **main** and checks out**.**

  git checkout -b feat_a

Git: a hands on example

- Fred checks “unstaged changes”, adds “foo.c” to staged changes and commits with message “add feature A”

    git status

    git add foo.c

    git commit -m “add feature A”

- He finishes feature A and merges changes to the **main** branch.

    git checkout main

    git merge feat_a

- He finally synchronizes the **main** branch and pushes changes back to “origin”

    git pull origin main

    git push origin main

Git: a hands on example

- While adding feature B, Barney realizes there's a critical error in the **main**.
  He stops development of feature B, fixes the error and goes back to feat B

  git checkout main

  ## ... fixes the error ... ##

  git commit -am "hotfix ⇐ ("-a" commits all unstaged changes)

  git pull origin main

  git push origin main

  git checkout feat_b

Git cookbook

- Clone a remote repository

  git clone <repo_url>

- Add a remote repository

  git remote add <remote> <repo_url>

- Pull remote branch from remote repository into a local branch

  git checkout <local_branch>

  git pull <remote> <remote_branch>

- Synchronize local branch to remote branch and push local changes

  git checkout <local_branch>

  git pull <remote> <remote_branch>

  git push <remote> <remote_branch>

Git cookbook

- Add a single file to to the staged changes (a.k.a changes to be committed)
  - *git add <file>*
- Commit staged changes in the current branch with a message
  - *git commit -m <commit message>*
- Shortcut to commit all unstaged changes
  - *git commit -am <commit message>*
- Create branch **foo** from branch **bar**
  - *git checkout bar*
  - *git checkout -b foo*

Git cookbook

- Merge **branch_a** into **branch_b**
  - *git checkout branch_b*
  - *git merge branch_a*
- For more information, visit the Atlassian Git Tutorials
  - https://www.atlassian.com/git/tutorials

GitHub

- It provides
  - Hosting for Git repositories
  - Workflows to ease collaboration
- We focus on two Github specific features
  - Forks
  - Pull Requests

GitHub: Forks

- Clone of a repository on the Github server.
- E.g user *wilma* creates a fork of the RIOT repository using the Github UI
  - RIOT repo URL (“upstream”) is github.com/RIOT-OS/RIOT.git
  - Fork URL: github.com/wilma/RIOT.git
- *wilma* clones the fork on her local machine

  git clone github.com/wilma/RIOT.git
- By default, remote “origin” points to her fork
- **NOTE:** . RIOT-OS/RIOT.git, wilma/RIOT.git and the repository in the local machine are three different repositories!
- She develops “feat C” in branch **feat_c** and wants to integrate changes into “upstream”
  - She is not the owner of RIOT-OS/RIOT and therefore does not have push access privileges.

GitHub: Pull Request (PR)

- Feature to request the owner of a repository to “pull” a branch from another (remote) branch
  - The Github UI handles these operations internally.
- In the previous case, *wilma* must push the branch to the fork (“origin”)

  git push origin <branch>
- Then, use the Github UI to open a PR pointing to the upstream branch
- Owner of the repository can request changes, accept or reject the PR
- Visit the GitHub PR documentation for more information

Questions?