

# Tooling for Technical Documentation II

Dominika Borges  
Technical Writer

Eliska Romanova  
Technical Writer

# About the authors



**Alexandra Nikandrova**

Technical Writer.  
Former Devops



**Apurva Bhide**

Senior Technical Writer  
TW



**Chandralekha Balachandran**

Senior Technical Writer  
Partner Ecosystem Team



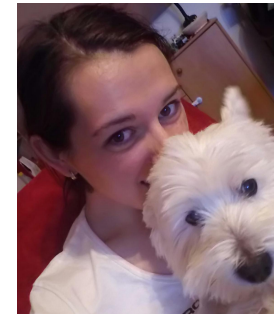
**Dominika Borges**

Technical Writer  
Former journalist



**Petr Hybl**

Technical Writer  
RHEL Security



**Eliška Romanová**

Technical Writer  
OpenShift



**Arati Ajit Belgaonkar**

Technical Writer and an athlete



# Markdown exercise review

# What we'll discuss today

- Version control system
- What is GIT
- GIT Glossary
- GIT Basic Workflow
- Demo
- Exercises
- Additional resources
- Questions



# Version Control System

# What is version control?

**Version control system** is a system for tracking and managing changes to the source code

## Three main capabilities

- Reversibility
- Concurrency
- Annotation

# Benefits of using version control?

## **Project history**

VCS maintains a detailed history of changes, tracking who made what changes, when, and why

## **Collaboration and parallel development**

Multiple team members can work on different parts of the documentation concurrently

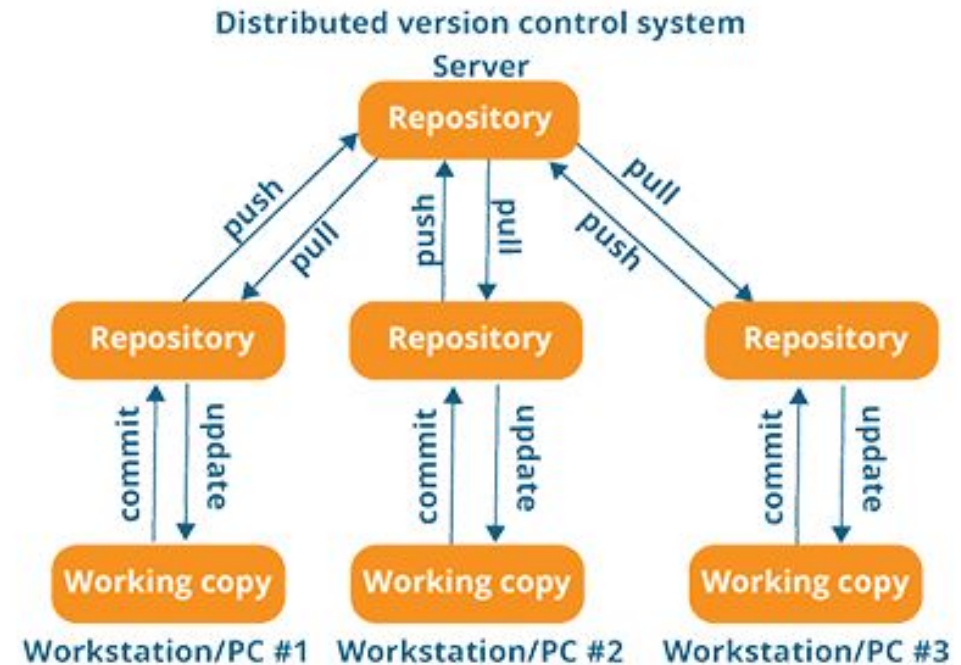
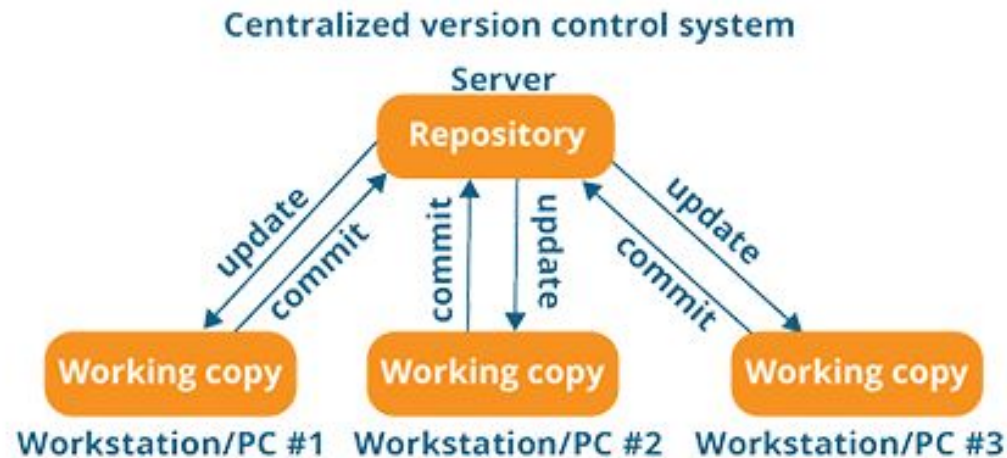
## **Rollback and revert**

In the case of errors or undesired changes, you can revert back to specific version of the document

## **Branching and merging**

Create branches to work on experimental features without affecting the main documentation

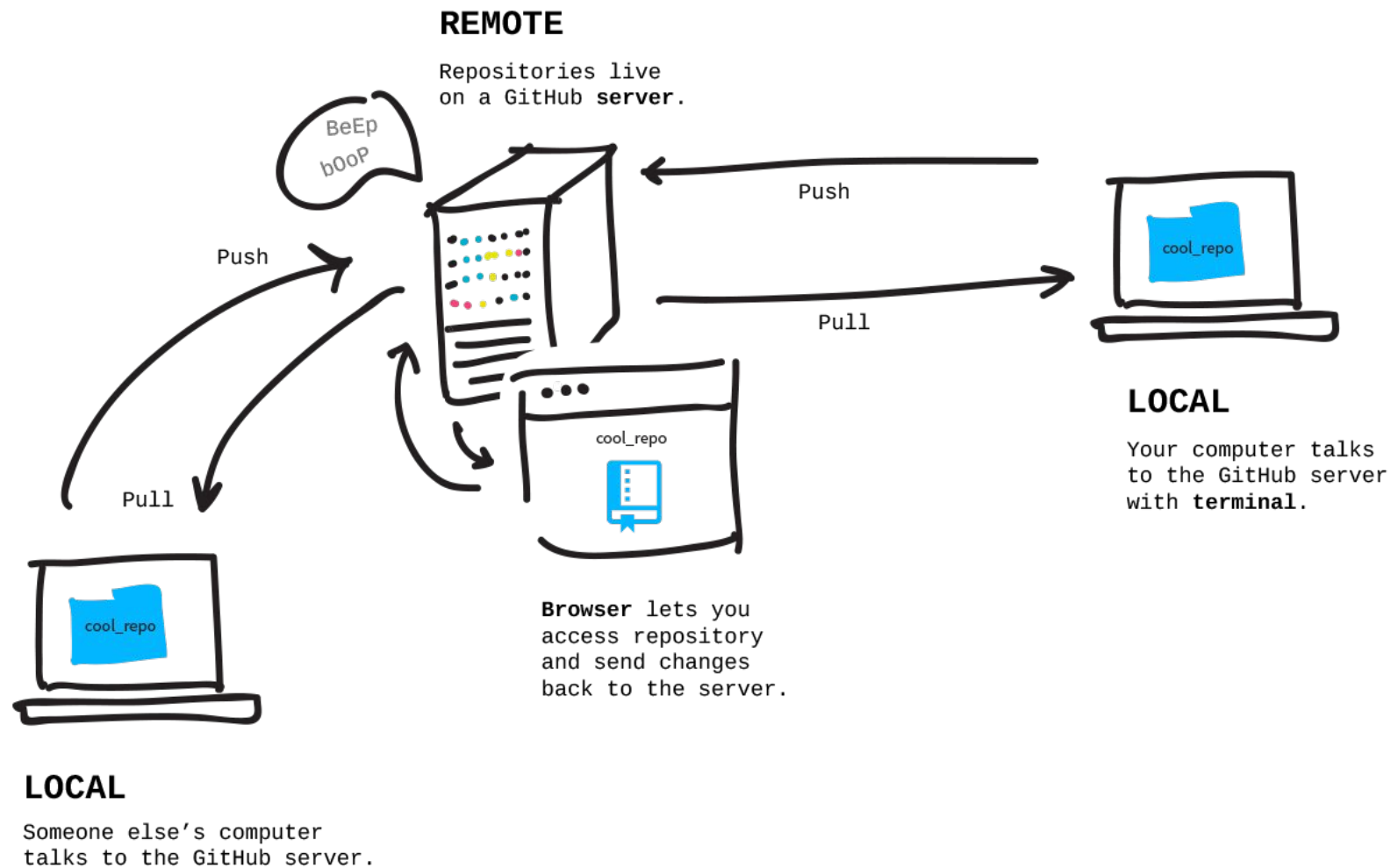
# Centralized & Distributed Version Control System





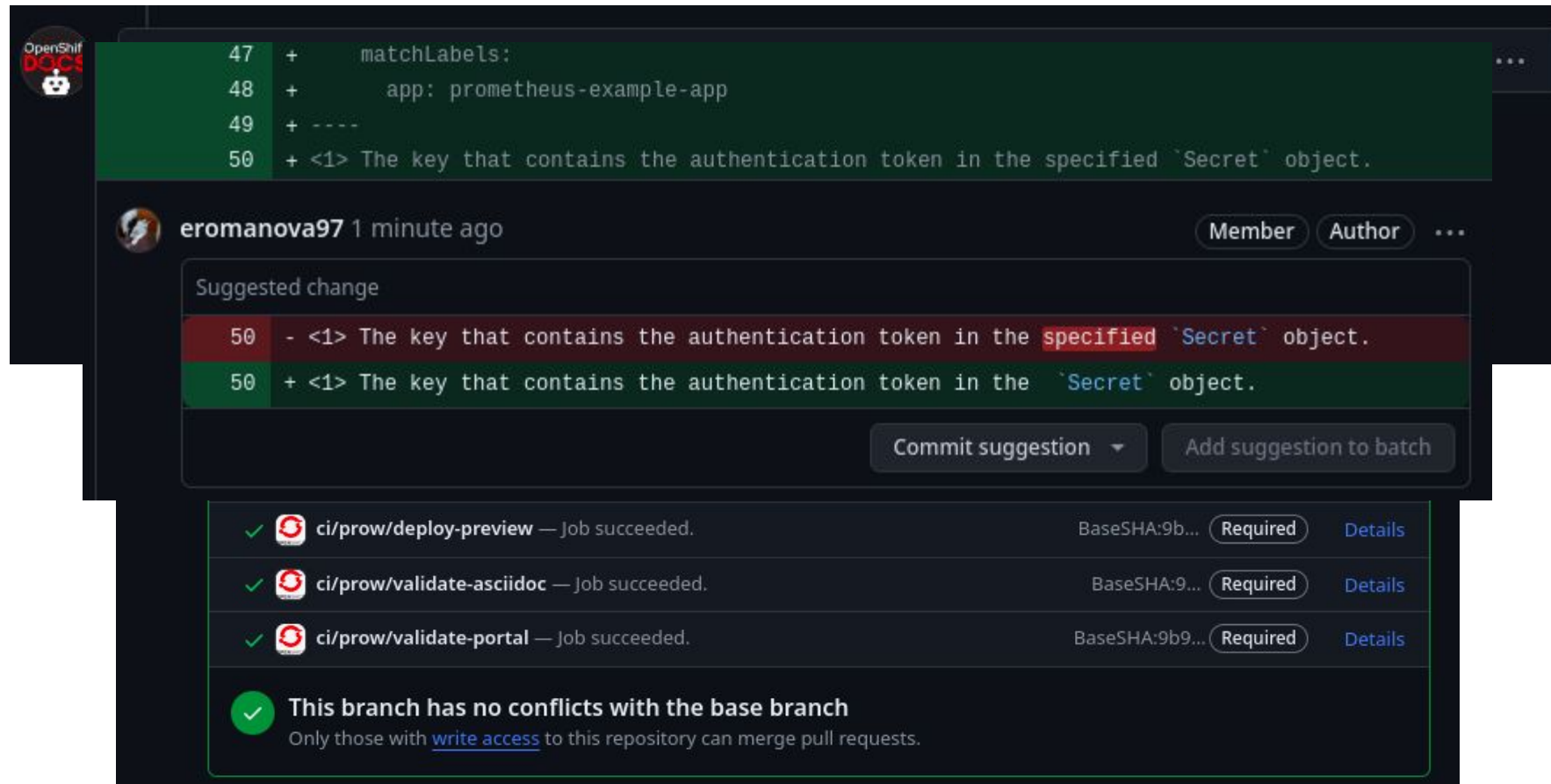
# What is Git?

Git is a version control software application



# Git forges

*An online platform for hosting Git repositories that provides additional features such as issue tracking, code review, continuous integration, and collaboration tools.*



# Setting up Git

Configure git with config command or directly edit the **.gitconfig** file

```
$ git config --global user.name "YOUR_NAME"
```

```
$ git config --global user.email NAME@example.com
```

Need help?

```
$ man git
```

<https://git-scm.com/>



# Git Glossary

# Git Glossary

- Repositories
- Branches
- Git Operations
  - Git Push and Git Pull
  - Git Clone and Git Fork
  - Git Merge and Git Rebase



Image Courtesy [Inflect](#)

# Repositories

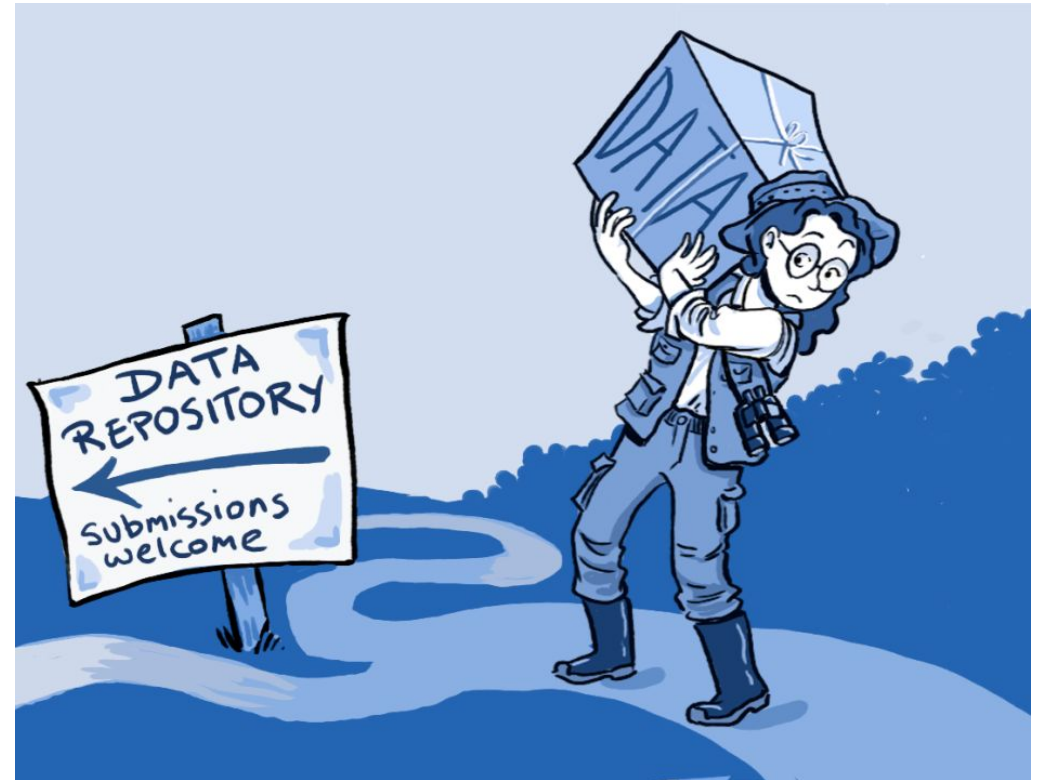


Image courtesy <https://www.linkedin.com/pulse/what-repository-parsa-panahpoor/>

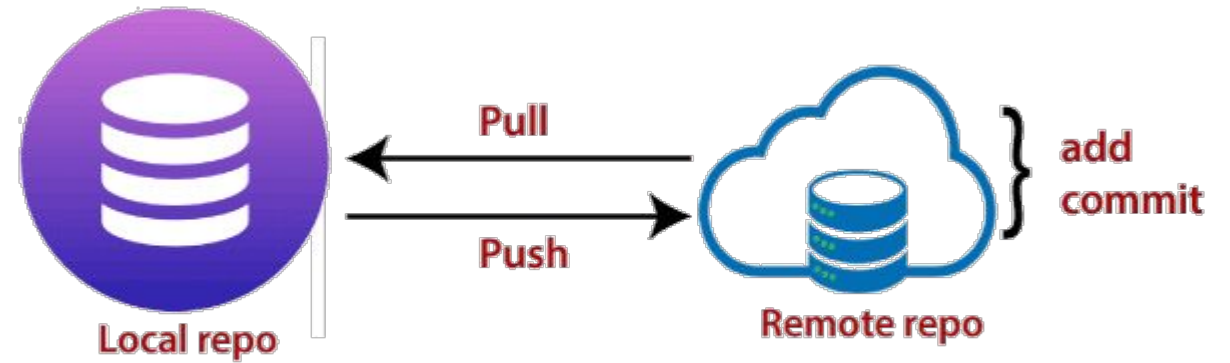
# Git repositories

## Local

- Located on your computer
- You work locally

## Remote

- Located on a server (GitHub, GitLab, etc.)
- Source of truth
- Fetching



# Branches



# Branches

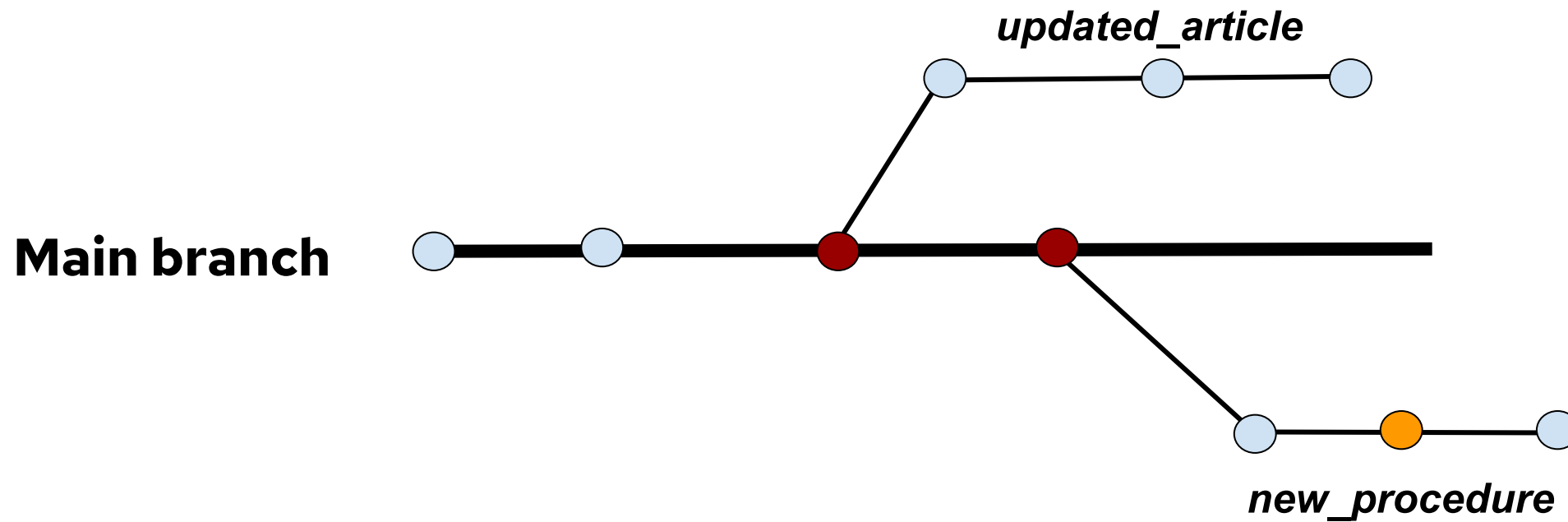
## What?

- An independent line of commits in chronological order to the project
- “Alternate history”
- **Topic branch** or **feature branch** is a lightweight branch for a specific purpose (e.g. new feature or bug fix) which could take some time

## Why?

- Work in parallel
- Keep main branch free from questionable code
- Experiment easily

# Branch workflow



# Git operations

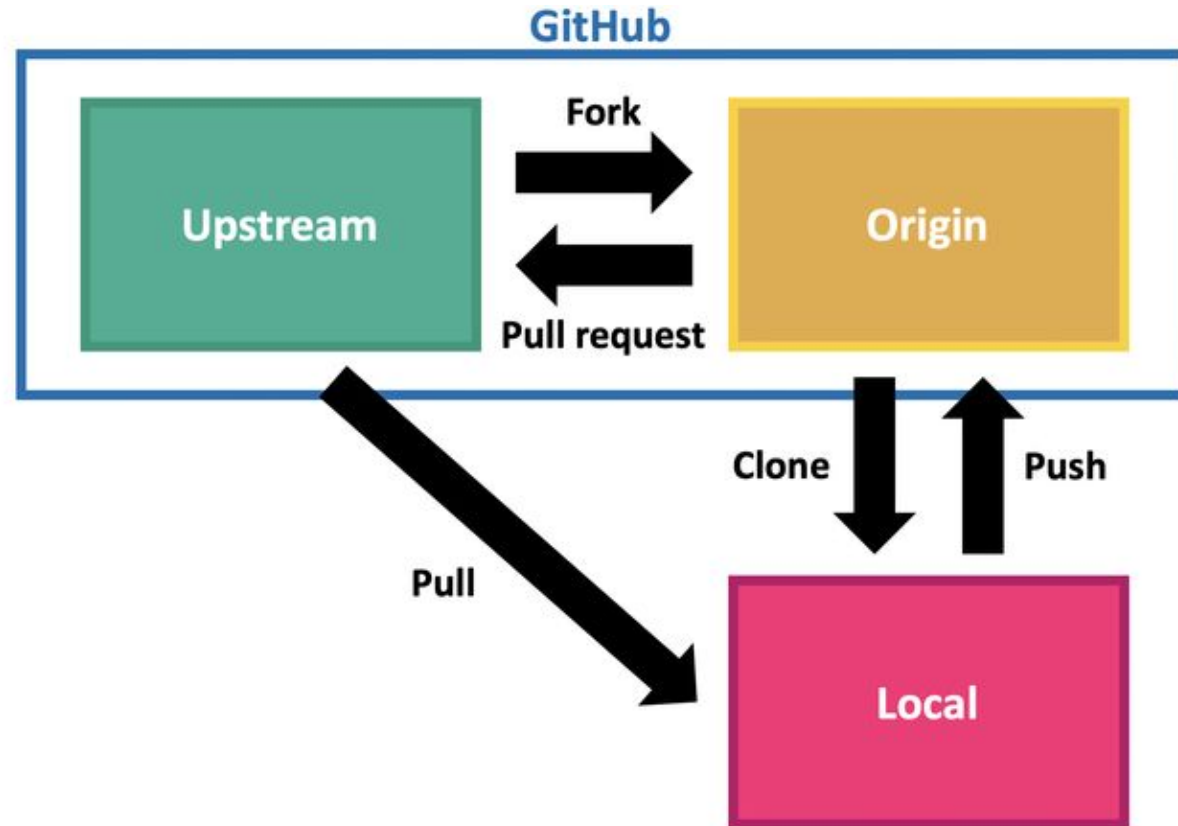
# Git workflow

## Fork

A personal copy of the repository

## Clone

A local copy of the repository/fork

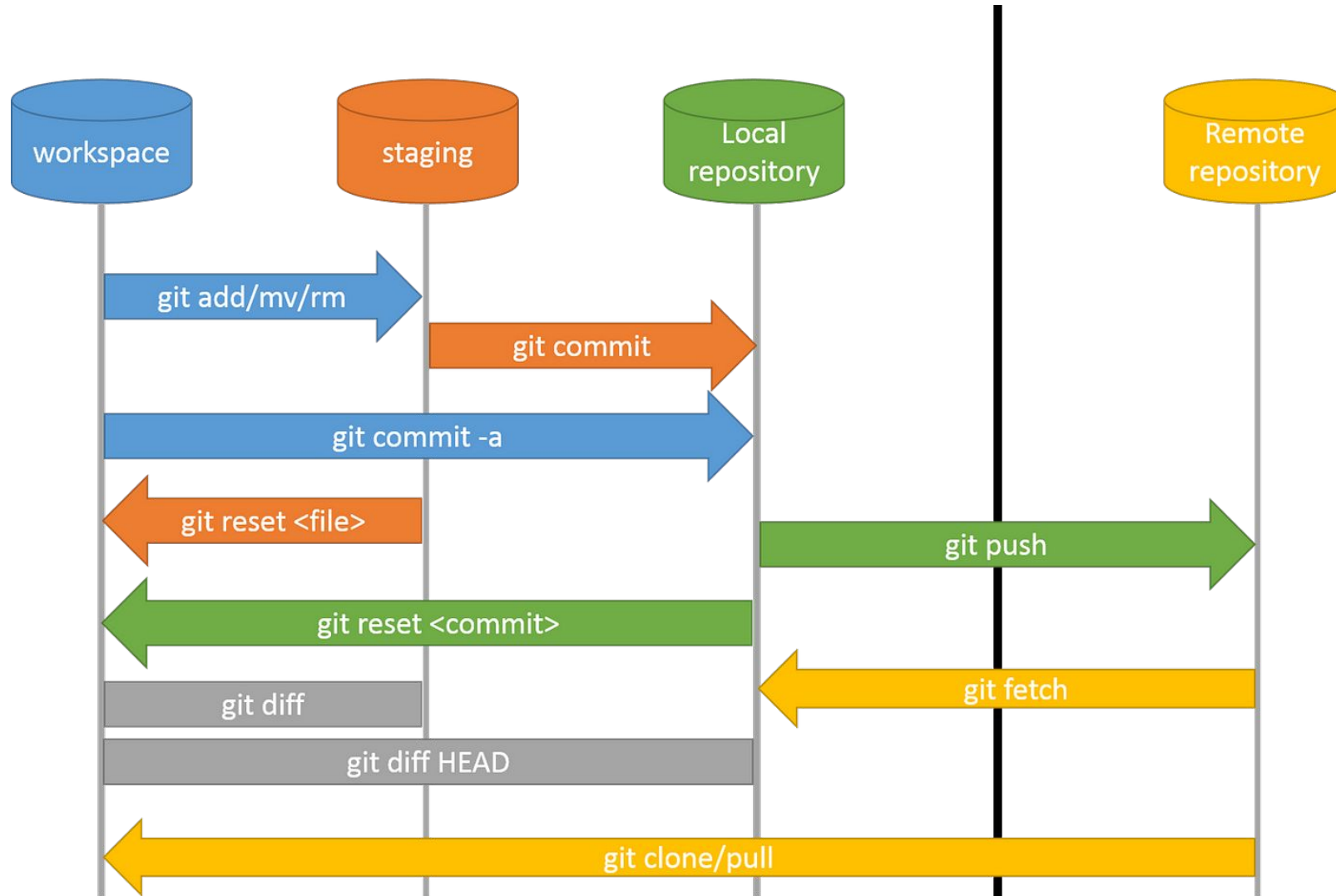


# Simplified Git workflow

Make changes -> stage changes -> commit changes

> Push the changes to the remote repository

# Git workflow



# GitHub - Create SSH key?

# Generating SSH keys - Windows



Verify if **OpenSSH Client** is Installed

1. Open the **Settings** panel, then click **Apps**.
2. Under the ***Apps & features*** heading, click **Optional Features**.
3. Scroll down the list to see if **OpenSSH Client** is listed and installed.



# Generating SSH keys - Windows



1. Open **Git Bash** > **Run as administrator**.
2. In the command prompt, run the following command:  
**ssh-keygen -t ed25519 -C "your-email@example.com"**

This creates a new SSH key, using your email as a label.

3. When you're prompted to "Enter a file in which to save the key", you can press **Enter** to accept the default file location (**C:\Users\your\_username\.ssh/id\_ed25519.pub**)
4. When you're prompted to enter a passphrase, you can press **Enter** to skip as well.

The system will generate the key pair, and display the key fingerprint and a randomart image.

# Generating SSH keys - Windows



```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Domini\.ssh\id_ed25519.
Your public key has been saved in C:\Users\Domini\.ssh\id_ed25519.pub.
The key fingerprint is:
SHA256:pEzGbYUnnIFfpgaxjjHKdkenGRr1zP8virXmIStfepc dvagnero@redhat.com
The key's randomart image is:
+--[ED25519 256]--+
|      o+.+.      |
|     oo*=-.+     |
|    + Bo0*       |
|   . . % 0+.     |
|  + + B.S .      |
|   . . .         |
|                |
|      . + ..     |
|     . BooE.     |
|    o=++o ..     |
+-----[SHA256]-----+
C:\Users\Domini>
```

# Add the SSH public key to your account on GitHub



1. Copy the SSH public key to your clipboard.
  - a. Open **id\_ed25519.pub** in editor and do **CTRL + A** and **CTRL + C**.
2. Navigate to your profile on Github and open **Settings**.
3. In the **Access** section of the sidebar, click **SSH and GPG keys**.
4. In the **Title field**, add a descriptive label for the new key. For example, if you're using a personal laptop, you might call this key **Personal laptop**.
5. In the **Key** field, paste your public key.
6. Click **Add SSH key**.

# Git Workflow - live demo

# Task



- Fork the upstream repository <https://github.com/rh-writers/BUT-technical-writing-course>
- Open Git Bash and go to the folder where you want to clone your repository.
- Clone your forked repository over ssh.
- Go to your newly created **BUT-technical-writing-course** repository.
- List the current remote repositories:  
**\$ git remote -v**
- Add the upstream repository as a remote:  
**\$ git remote add upstream <SSH>**
- Verify the new remote was added:  
**\$ git remote -v**

## Task - continue



- Create a branch in your local repository.
- Inside the **exercises** directory, create a subdirectory named **firstname-lastname**.
- Add your assignment, the `.md` file, and any image files in your subdirectory.
- Stage the changes.
- Commit the change.
- Push the changes to your fork and create a pull request against the repository.
- Tag **@domiborges** and **@eromanova97** to review and merge your contribution.

**IMPORTANT:** Ensure that you are on main branch when you are creating you new topic branch.

## Fixing mistakes

**Scenario:** You've accidentally staged files that you didn't intend to include in your commit.

Use **git reset** to unstage the files without losing your changes.

```
$ git reset <filename>
```

**TIP:** Always use **git status** and **git diff** to review your changes before staging and committing them.

## Fixing mistakes

**Scenario:** You've ve staged files directly on the **main** branch instead of creating a topic branch.

Unstage the files \$ **git reset <filename>**

Stash your changes \$ **git stash**

Create and switch to the correct branch \$ **git checkout -b <topic-branch>**

And apply the stashed changes to the correct branch \$ **git stash pop**



# Fixing mistakes



I committed something to main that should have been on a brand new branch!

**Scenario:** You have made a commit to main. You realize you made a mistake. You want to undo the commit. You want to create a new branch.

instead of creating a topic

① Make sure you have `main` checked out:

`git checkout main`

```
dvagnero@fedora:~/git_repos/BUT-technical-writing-course$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
dvagnero@fedora:~/git_repos/BUT-technical-writing-course$
```

② Remove the unwanted commit from main.

Always start with **\$ git status**

`git status`

`git reset --hard HEAD~`

careful!

④ Check out the new branch!

`git checkout my-new-branch`



'`git branch`' and '`git checkout -b`' both create a new branch. The difference is '`git checkout -b`' also checks out the branch

## Fixing mistakes

**Scenario:** You've committed a file with a typo in its name **dminikaborges.md**.

Rename the file using git **mv <FileName> <NewFileName>**:

```
$ git mv dminikaborges.md dominikaborges.md
```

Amend the previous commit to include the filename update:

```
$ git commit --amend --no-edit
```

If the commit has already been pushed to a remote repository, force-push the changes:

```
git push origin your-branch-name --force
```

**Warning:** Force pushing can overwrite the commit history on the remote repository, potentially causing issues for other collaborators.

# Fixing mistakes

**Scenario:** Oops, unwanted commit! You've made a commit that you don't want.

```
dvagnero@fedora:~/git_repos/BUT-technical-writing-course$ git log
commit f29add87007f98cc4476c95a07124afceddf36fb (HEAD -> text)
Author: Dominika Borges <dvagnero@redhat.com>
Date: Thu May 23 09:23:53 2024 +0200

    Oops, unwanted commit

commit 06030726168d33f2caacb4f9e561c67c343fdce4 (upstream/main, origin/main, origin/HEAD, main)
Author: domiborges <dvagnero@redhat.com>
Date: Wed May 22 13:38:20 2024 +0200

    Update README.md
```

To remove the commit and keep the working directory changes:

**\$ git reset --soft HEAD~1**

# Fixing mistakes

**Scenario:** Oops, unwanted commit! You've made a commit that you don't want.

```
dvagnero@fedora:~/git_repos/BUT-technical-writing-course$ git log
commit f29add87007f98cc4476c95a07124afceddf36fb (HEAD -> text)
Author: Dominika Borges <dvagnero@redhat.com>
Date: Thu May 23 09:23:53 2024 +0200

    Oops, unwanted commit

commit 06030726168d33f2caacb4f9e561c67c343fdce4 (upstream/main, origin/main, origin/HEAD, main)
Author: domiborges <dvagnero@redhat.com>
Date: Wed May 22 13:38:20 2024 +0200

    Update README.md
```

To remove the commit and **discard the changes**:

**\$ git reset --hard HEAD~1**

## Fixing mistakes

**Scenario:** You have a topic branch with several small commits that logically belong together. You want to combine these commits into a single, more descriptive commit and send a pull request.

Review the commit history to identify the range of commits you want to squash: **\$ git log**

Start with the interactive rebase: **\$ git rebase -i HEAD~3**

Modify the commit list in the text editor. Choose which commits you want to squash.

Git will prompt you to write a combined commit message.

# Fixing mistakes

## Example interactive rebase - squashing commit

```
commit d4065b941d9ddc77e74487df246a8c26507dc246 (HEAD -> rebase-example)
Author: Dominika Borges <dvagnero@redhat.com>
Date: Thu May 23 13:52:29 2024 +0200
    # This is a combination of 2 commits
commit 27193a3f752e0fca5b7e8705b811f212f0fbf4e0 (HEAD -> rebase-example)
Author: Dominika Borges <dvagnero@redhat.com>
Date: Thu May 23 13:51:23 2024 +0200
    add materials for git lesson
    add slides for git lesson
    add exercise for git lesson
    add homework for git lesson
    add slides for
```

# Git rebase

Rewrite commit history, adjust the order of commits, resolve conflicts, and more.

- Before merging or pushing your changes, rebase your topic branch onto the main branch to incorporate the latest changes.
- Remove unwanted or incorrect commits from your branch.
- Combine multiple small or related commits into a single commit. Split a large commit into smaller, logical commits.
- Revisit previous commits and edit them.

# Best practices for Git

- Follow the best practices and guidelines for contributing to a given project.
- Use hyphens as separators along with task details while naming your branch.
- Write meaningful commit message.
- Write an useful description for your PR.
- Use .gitignore file to ignore any unnecessary files.
- Rebase your local commits before your pushing changes.
- Build documentation locally and review files before inviting SMEs and reviewers.
- Push changes as soon as your file is ready instead of keeping it in a local machine.



# Additional resources

- <https://about.gitlab.com/topics/version-control/version-control-best-practices/>
- <https://learngitbranching.js.org/>
- <https://git-scm.com/doc>
- <https://docs.github.com/en/get-started/quickstart/hello-world>
- <https://www.atlassian.com/git>
- <https://www.w3schools.com/git/>
- <https://git-scm.com/docs/gitignore>
- <https://www.theodinproject.com/lessons/foundations-commit-messages>

# Q&A



# Homework