The fundamentals of technical writing

Tooling II

Dominika Borges
Technical Writer

Petr Hybl
Technical Writer

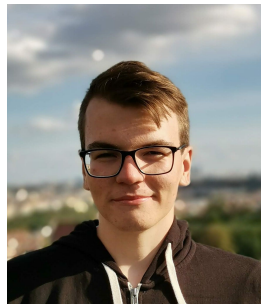
About the authors

Alexandra Nikandrova
Technical Writer.
Former Devops

Apurva Bhide
Senior Technical Writer
TW

Chandralekha Balachandran
Senior Technical Writer
Partner Ecosystem Team

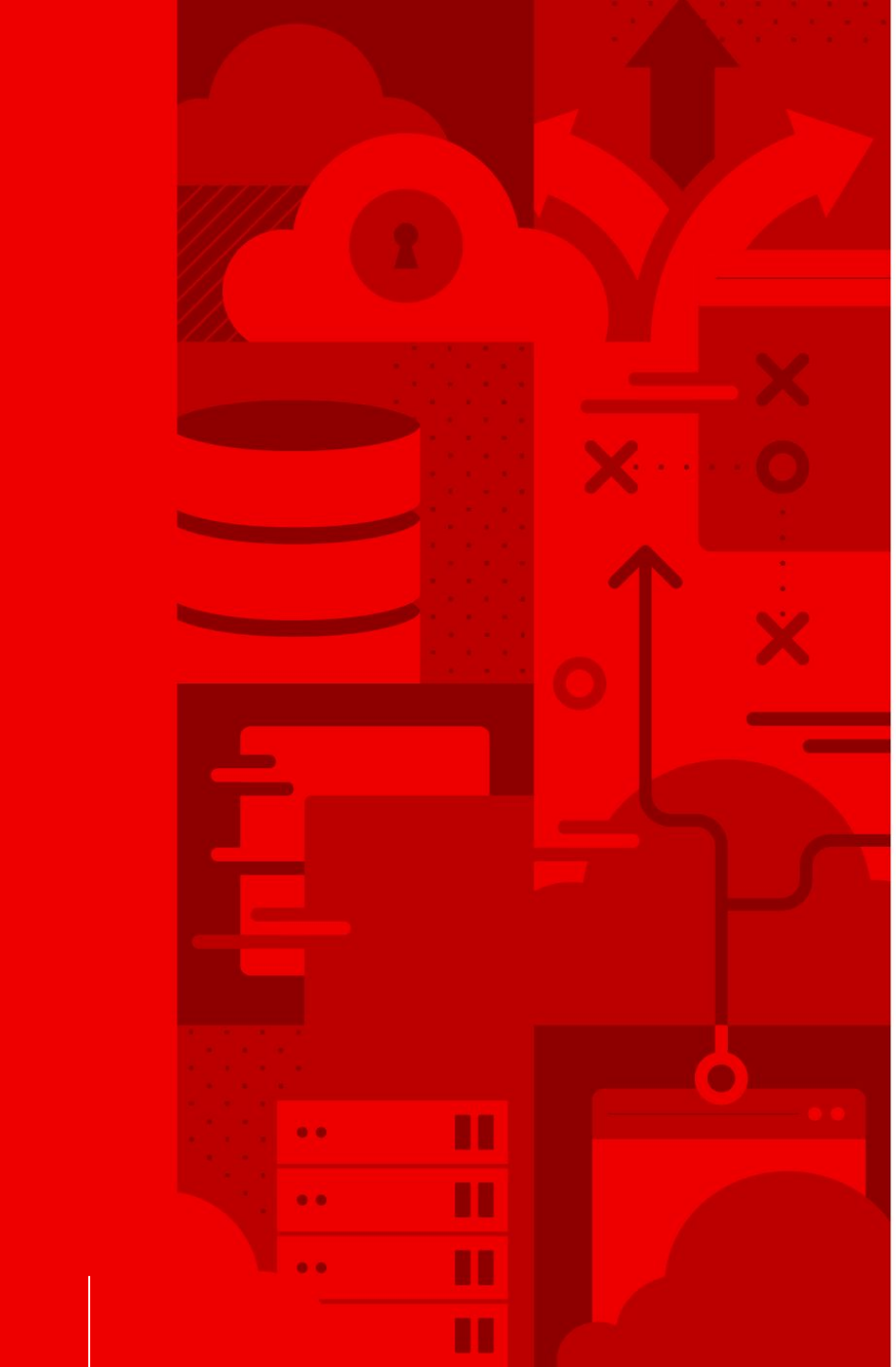
Dominika Borges
Technical Writer
Former journalist



Petr Hybl
Technical Writer
RHEL Security

What we'll discuss today

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



Final project

Week 1

Final project introduction

Apply acquired skills from lessons to complete a technical writing task within the selected project:

- Join a software project as a technical writer
- Follow the established processes for the given project
- Use specific set of tools including different markup languages
- Research information and interview SMEs to complete your task(s)

Create a pull request and undergo reviews to ensure that your content is technically accurate and written according to tech writing standards

Final project requirements

- Successful completion of the final project is mandatory to pass the course
- Maximum points: 200 (**Minimum of 130 points necessary to pass**)
- Duration: Six weeks

Final project schedule

- Week 1, March 26 (after class 6, Tools II): **Select a project, sign up for issues, fork the repository, and join the communication channel to introduce yourself.**
- Week 2, April 2 (after class 7, Tools III): Conduct thorough research on the subject. Ask questions to your SMEs.
- Week 3, April 9 (after class 8, Usability): Create a draft.
- Week 4, April 16 (after class 9, Soft Skills): Create a pull request and undergo SME review.
- Week 5, April 23 (after class 10, LLMs): Close the SME review.
- Week 6, April 30: Peer review.
- **Week 7, May 7 at 9:59**: Deadline to submit your content.

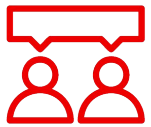
  Present the project.

Final project minimum requirements

- **Create documentation that sufficiently addresses the issue in the ticket**
  - Demonstrate an understanding of the subject matter and how your documentation will help to address user needs
  - Use technical writing style (focus on the UX, minimalism)
  - Use the appropriate markup correctly
- **Create a Pull Request (PR) for merging the updates and request reviews from the appropriate stakeholders**
  - Use well-structured commits and informative messages, name and label the PR correctly
  - Be direct but respectful towards the stakeholders
- **Address the feedback received from reviewers**
  - Constructive discussion with the stakeholders
  - Apply reviews correctly
- **Optional: Present your work in the final class of the course (5 min/15 points)**
  - Lessons learned and key takeaways
  - Address ensuing questions and discussion points

Final project

Engage with real-world open source projects and contribute technical documentation to one of the following:



- Ansible
- Foreman
- GNOME User documentation

Ansible

Prerequisites: reStructuredText, Python, YAML, and Git

Ansible community documentation at docs.ansible.com

Join the Documentation Working Group on Matrix (Communication guide)

Follow Contributing to the Ansible Documentation & Ansible documentation style guide

Want to contribute to Ansible documentation?

➔ Fork the documentation repository

➔ Select from available documentation issues

GNOME Settings

Prerequisites: Mallard (XML), Git, Linux or ability to run Linux OS in a virtual machine

Visit User & system settings documentation

Follow Contributing to the GNOME user docs

Join the Chat? https://matrix.to/#/#docs:gnome.org

Want to contribute to GNOME user documentation?

➔ Clone the documentation repository

➔ Select from available documentation issues

Foreman

Prerequisites: AsciiDoc, Git, the courage to contribute to a complex project

Visit Foreman documentation (find the guides under the dropdown menus

Foreman on EL, Foreman on Debian, Katello on EL, etc.)

Read the README, especially its Contributing section

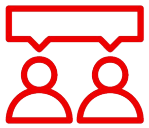
Find devs and writers on Matrix: dev chat room, doc chat room

Want to contribute to Foreman user documentation?

➔ Fork & clone the documentation repository

➔ Select from available documentation issues labeled *tw uni course*

Where to start?



- Review the projects
- Select a project and sign up for issues
  - Each repository contains a list of issues labeled **muni-tech-writers**
  - One writer per issue, one pull request per issue
  - Issues are categorized according to estimated effort required to complete them **Large (L), Medium (M), and Small (S)**
  - You need to spend equal amount of time **L = 2*M = 4*S**
  - For example, Anna signs up for one Larger task and Will selects one Medium and two Small tasks.
- Fork and clone the repository
- Join the communication channel to introduce yourself

Need help?

General queries, organization, and grading: Discord #general

Final project queries

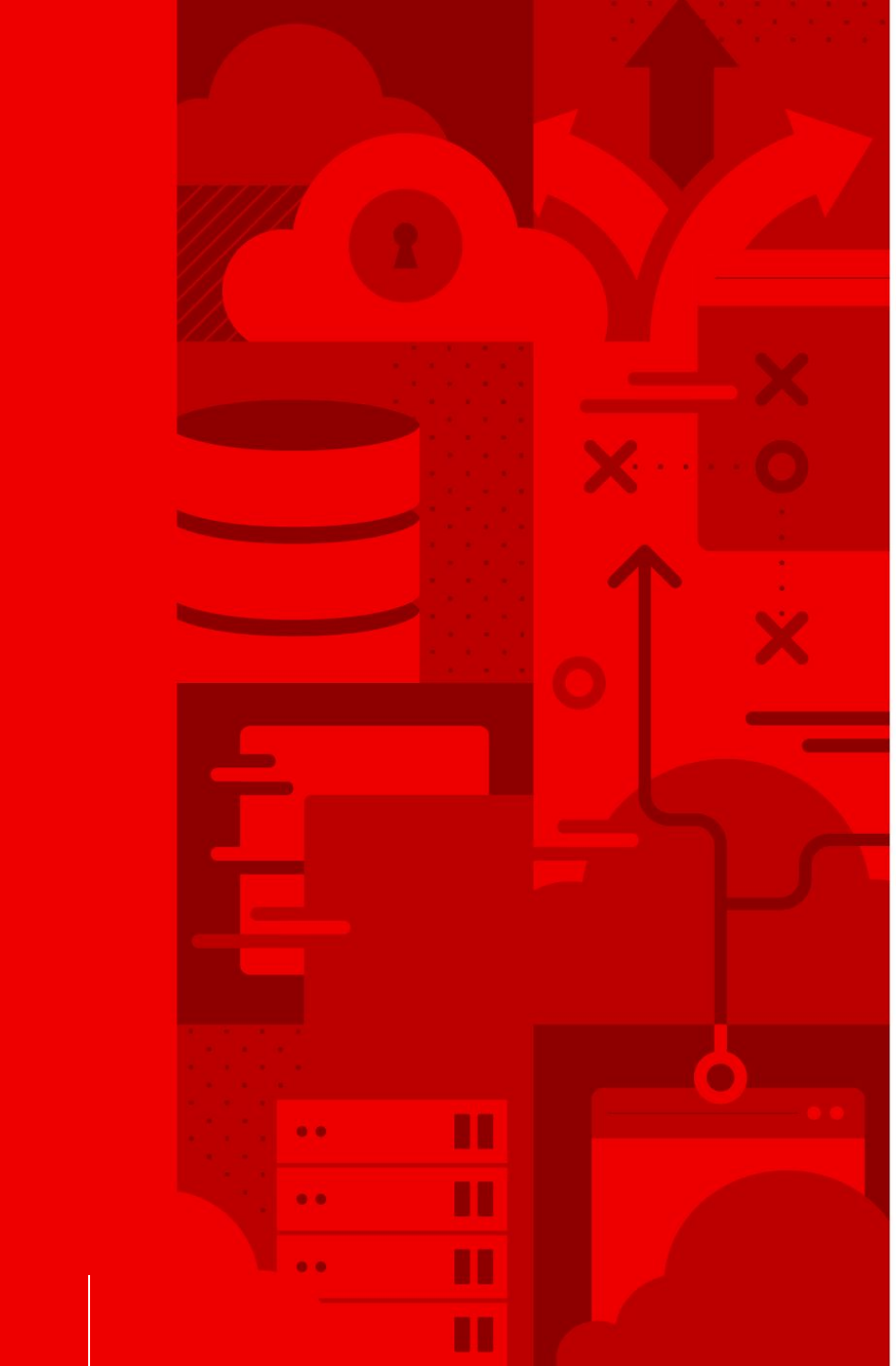
- Ansible #ansible-final-projects
- Foreman #foreman-final-projects
- GNOME User docs #gnome-final-projects

Technical and project specific queries

Use the community forums, alternatively reach out to SMEs directly:

- Ansible dnaro@redhat.com
- Foreman apetrova@redhat.com
- GNOME Settings feborges@redhat.com

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

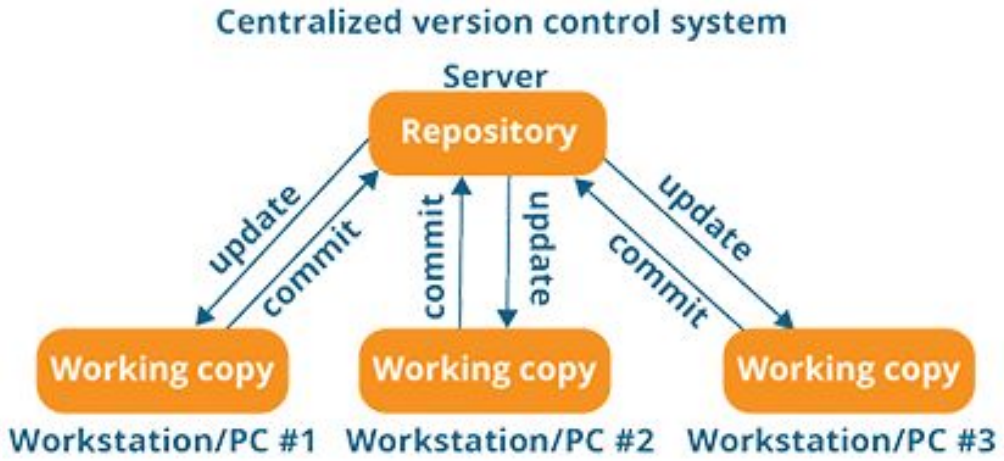
Rollback and revert

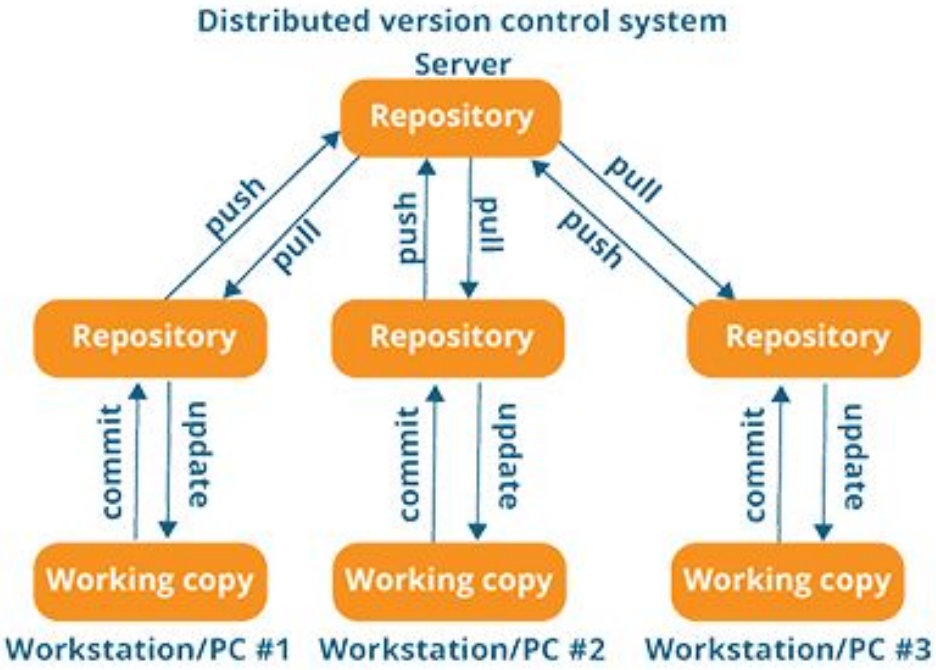
In the case of errors or undesired changes, you can revert back to specific version of the document

Branching and merging

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

Centralized & Distributed Version Control System

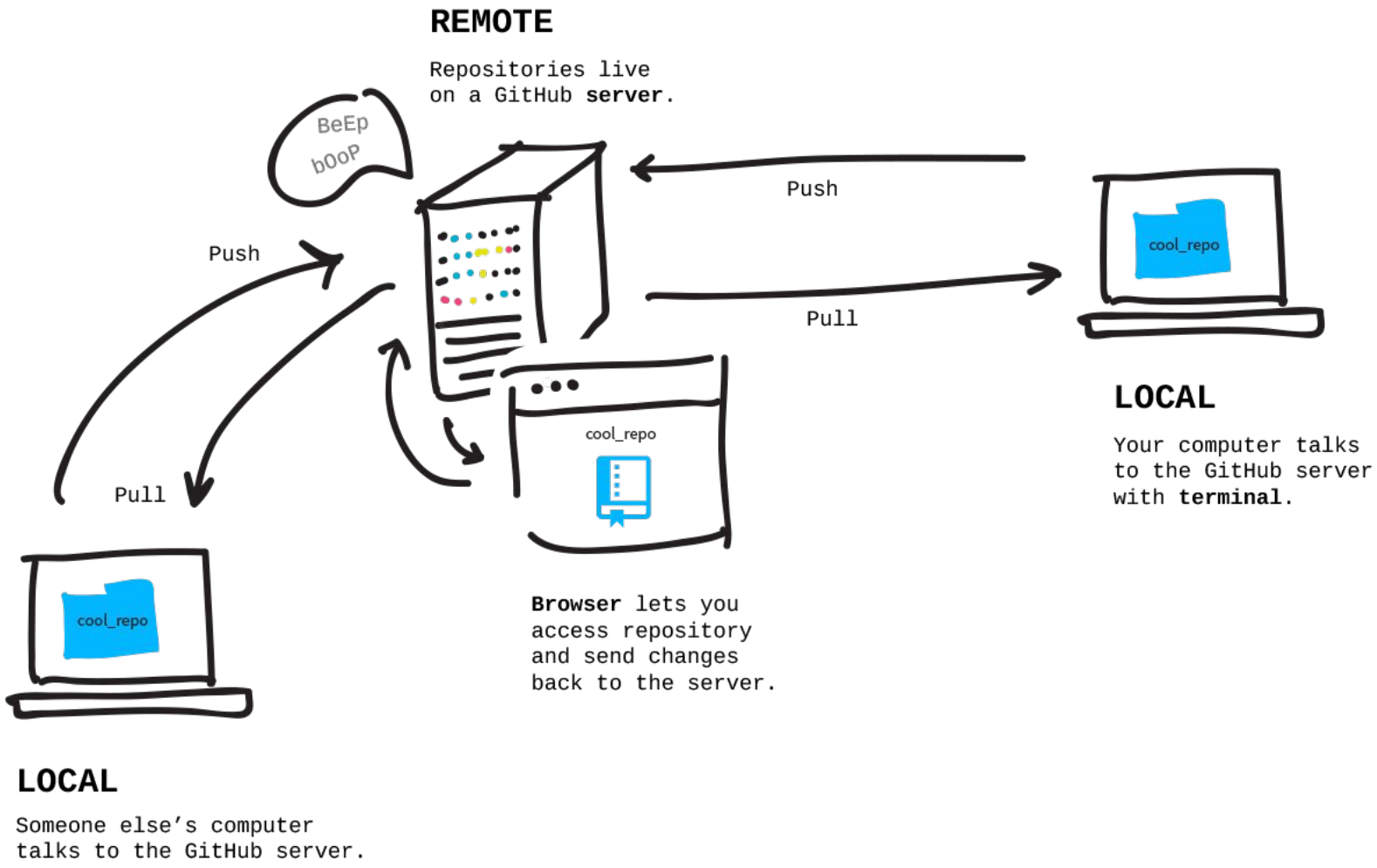




Source https://www.edureka.co/blog/what-is-git/

What is Git?

Git is a version control software application



Source https://www.freecodecamp.org/news/the-beginners-guide-to-git-github/

Git forges

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

Suggested change

```
-  * A simple wrapper for a configuration represented as a String. The configuration can be represented in any
-  * of the supported formats: XML, JSON and YAML.
+  * A simple wrapper for a configuration represented as a String. The configuration can be in any
+  * of the supported formats: XML, JSON, and YAML.
```

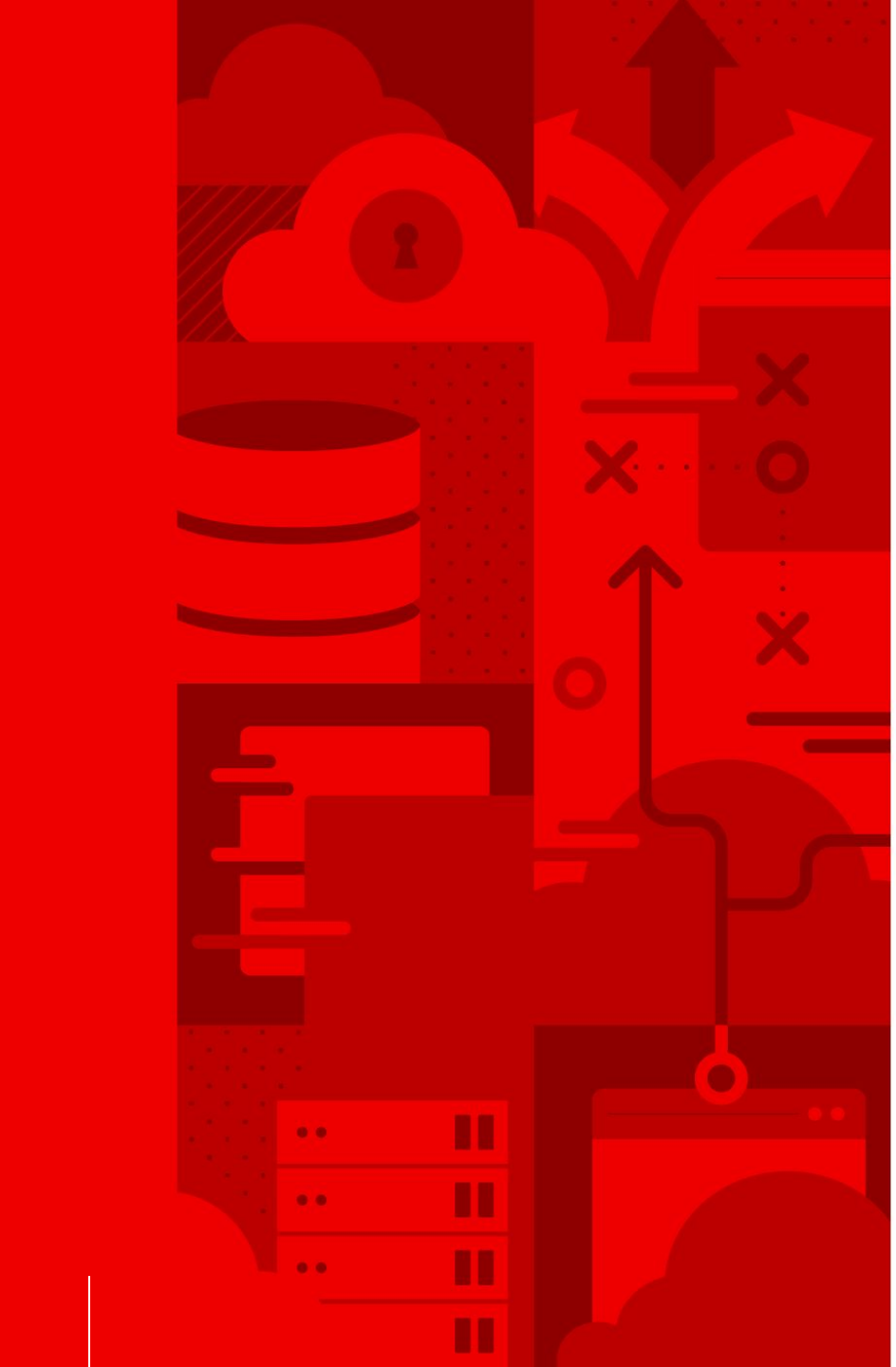
CP Ops Service Account @cp-ops-service · 2 days ago Developer

✅ Preview for title rhel-8/titles/configuring-and-maintaining/configuring-and-managing-networking

✅ Preview for title rhel-9/titles/configuring-and-maintaining/configuring-and-managing-networking

Previews are retained for 10 days. If you get a 404 error from the above link(s), make a "rebuild" comment and the preview will be regenerated

Source https://www.freecodecamp.org/news/the-beginners-guide-to-git-github/



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 Infilect

Repositories

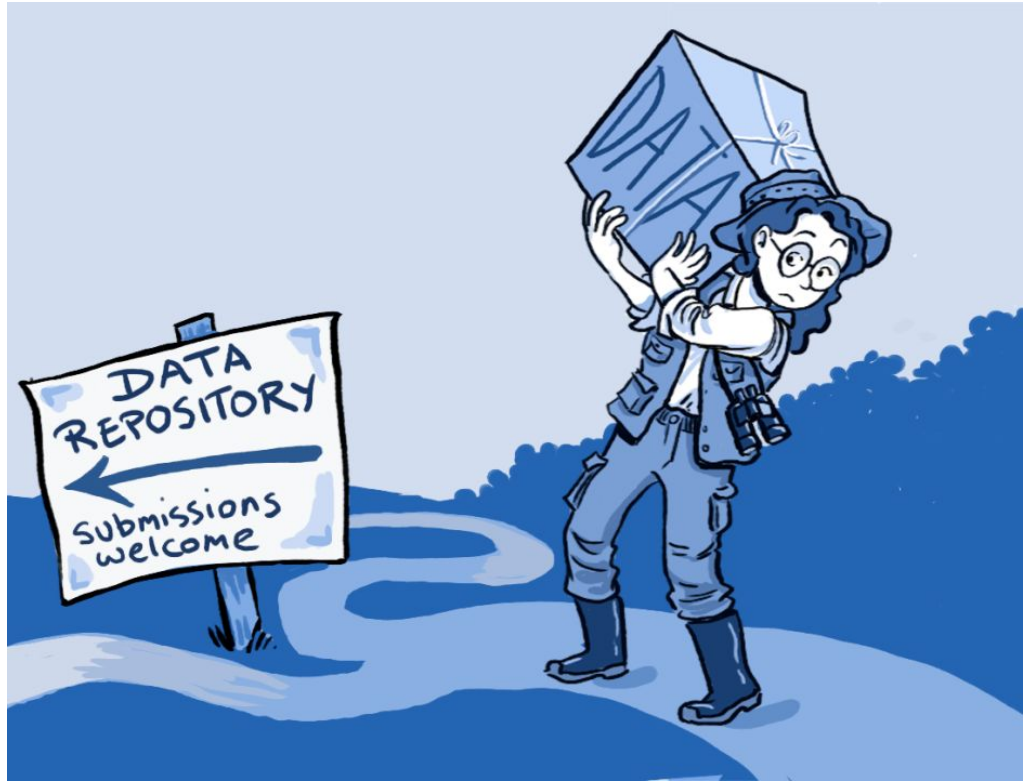


Image courtesy https://www.linkedin.com/pulse/what-repository-parsa-panahpoor/

Git repositories

Local

- Located on your computer
- You work locally

Remote

- Located on a server (GitHub, GitLab, etc.)
- More than one can be linked to your local repository
- You must synchronize your local repository with the remote one\
- Forking

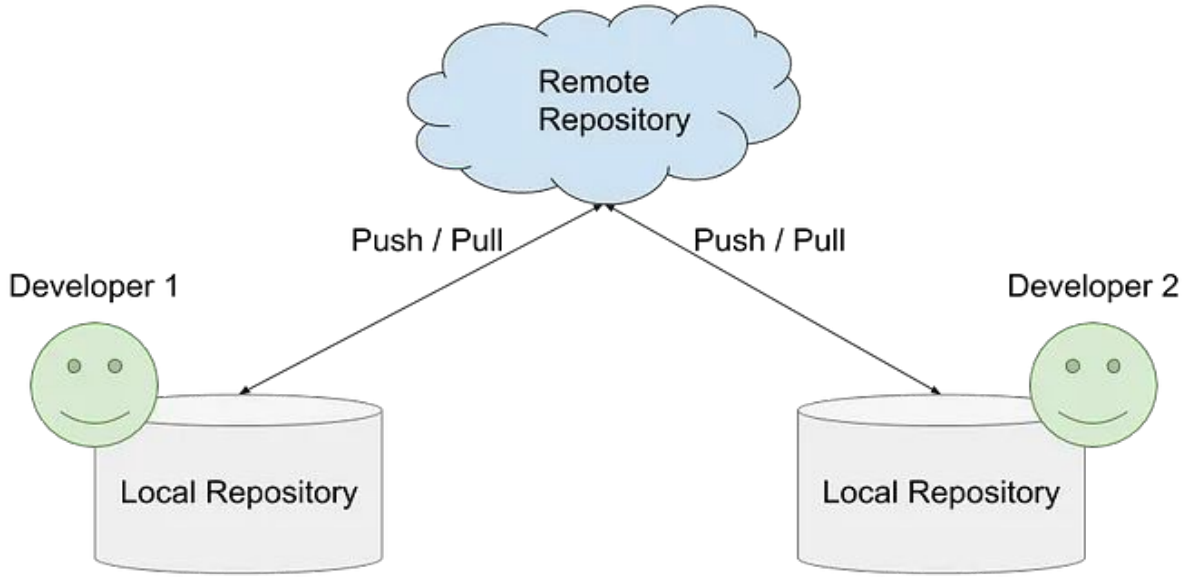


Image courtesy https://medium.com/it-developers/git-tutorial-for-beginners-remote-repository-management-490fa4937fab

Branches

Branches

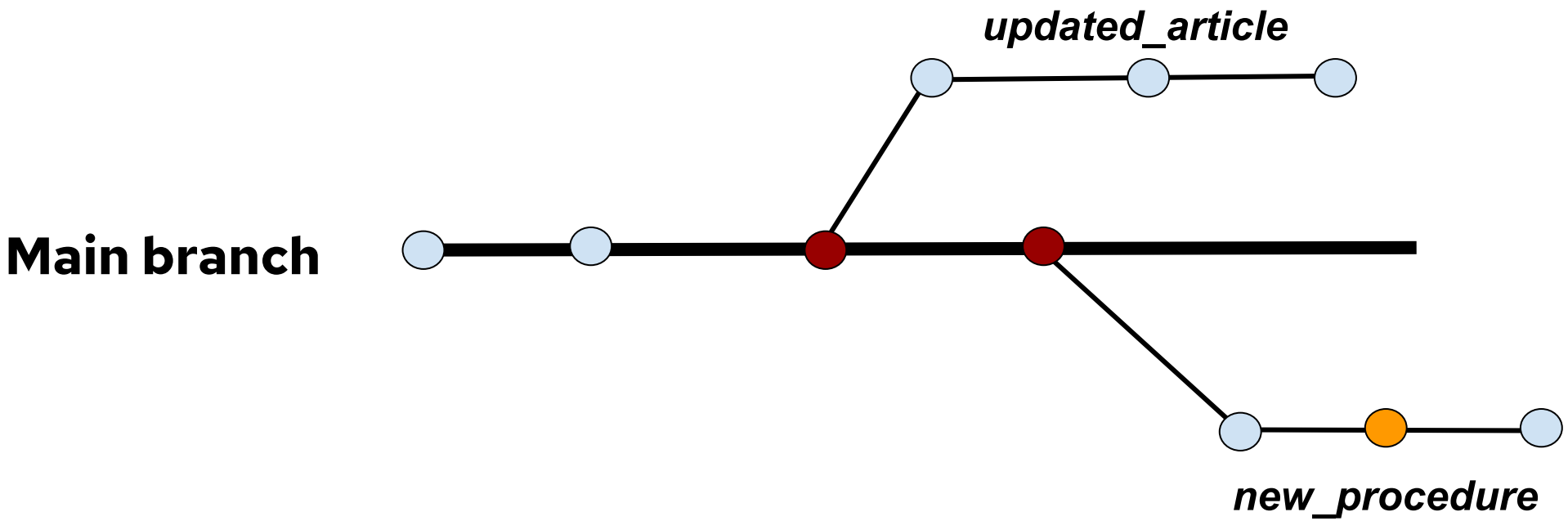
What?

- 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

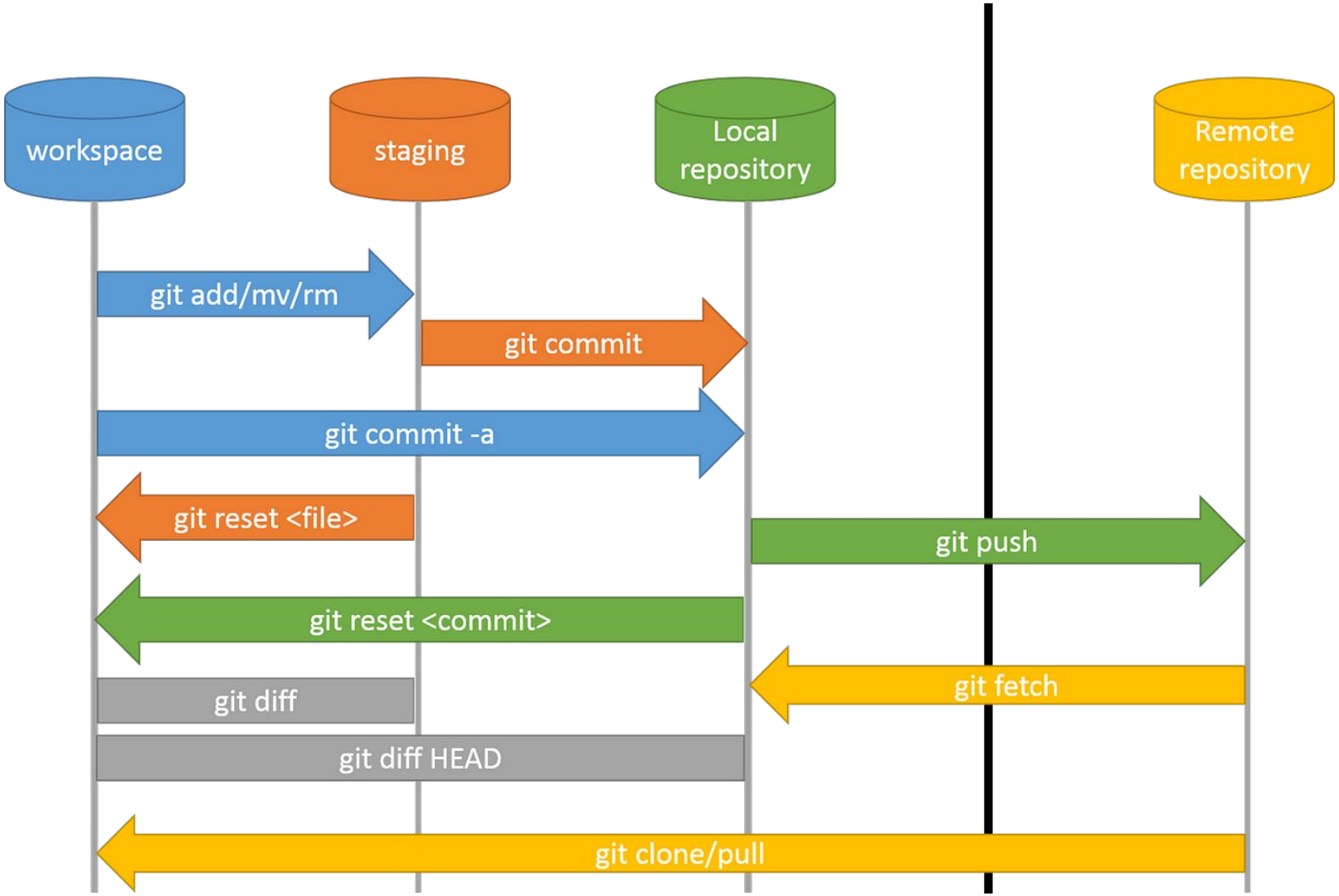


Image courtesy
https://qavalidation.com/2018/07/git-push-pull-commands-local-repository-to-github.html/#google_vignette

Git Merge and Rebase

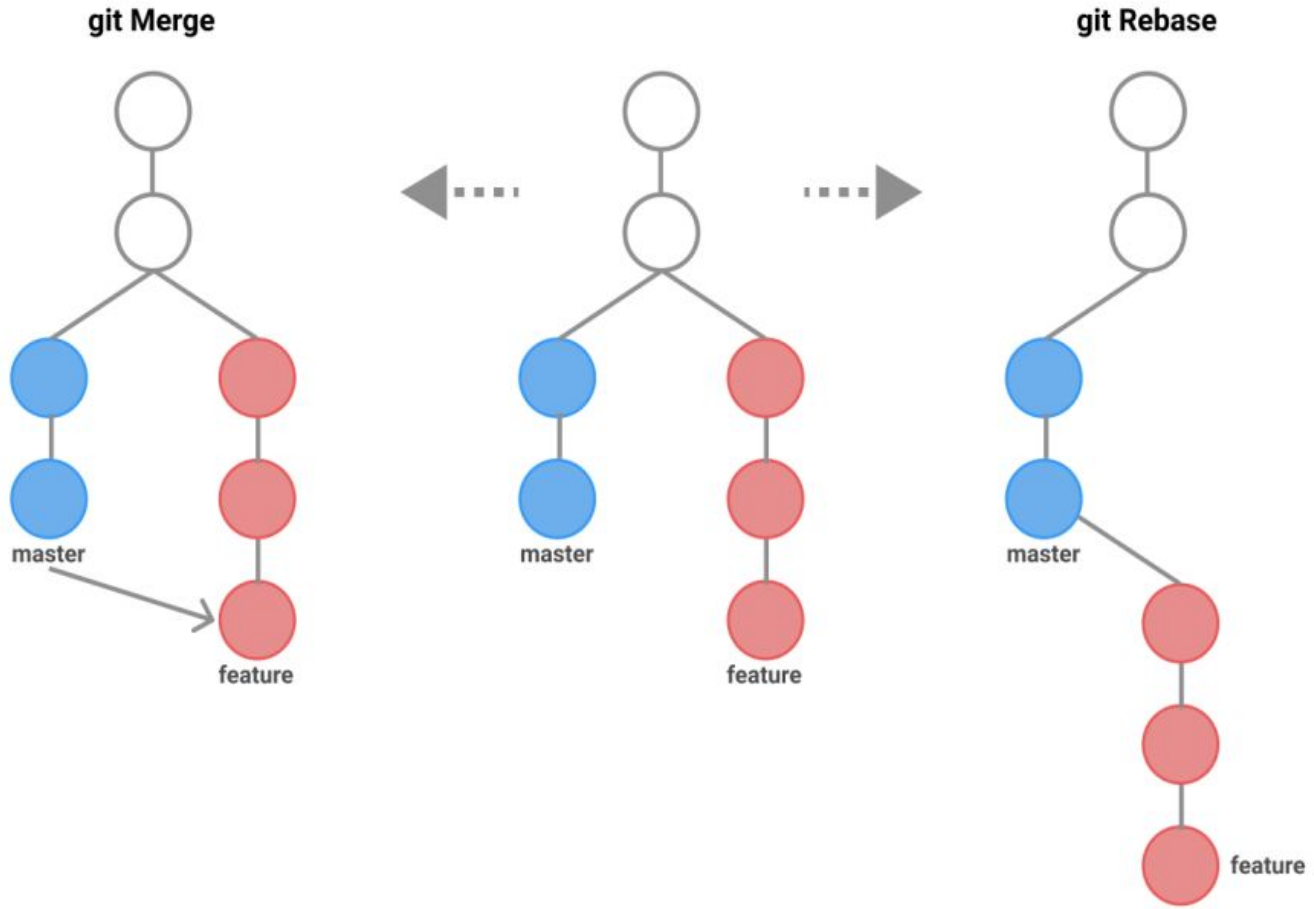
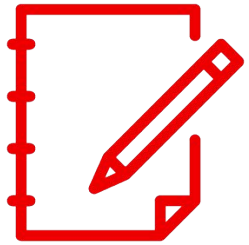


Image courtesy https://www.linkedin.com/pulse/understanding-difference-between-git-merge-rebase-which-garg-/

Git Workflow - live demo

Task

- Clone this repository (use SSH) https://github.com/rh-writers/technical-writing-course-brno
- Create a branch in your local repository.
- List branches.
- Create another branch and switch to it.
- Create a subdirectory with your name under the homework-projects directory.
- Create a .md file in your subdirectory.
- Commit the change.
- Create a pull request against the main repository.
- Tag **@domiborges** and **@PetrsGamers** to review and merge your contribution.



Git Workflow - live demo

Task II - Forking

- Fork the upstream repository https://github.com/rh-writers/technical-writing-course-brno.
- Clone your forked repository over ssh.
- Navigate to the newly created technical-writing-course-brno repository.
- List the current remote repositories:

  $ git remote -v

- Add the upstream repository as a remote:

  $ git remote add upstream

  git@github.com:rh-writers/technical-writing-course-brno.git

- Verify the new remote was added:

  $ git remote -v

Task II - Collaborating

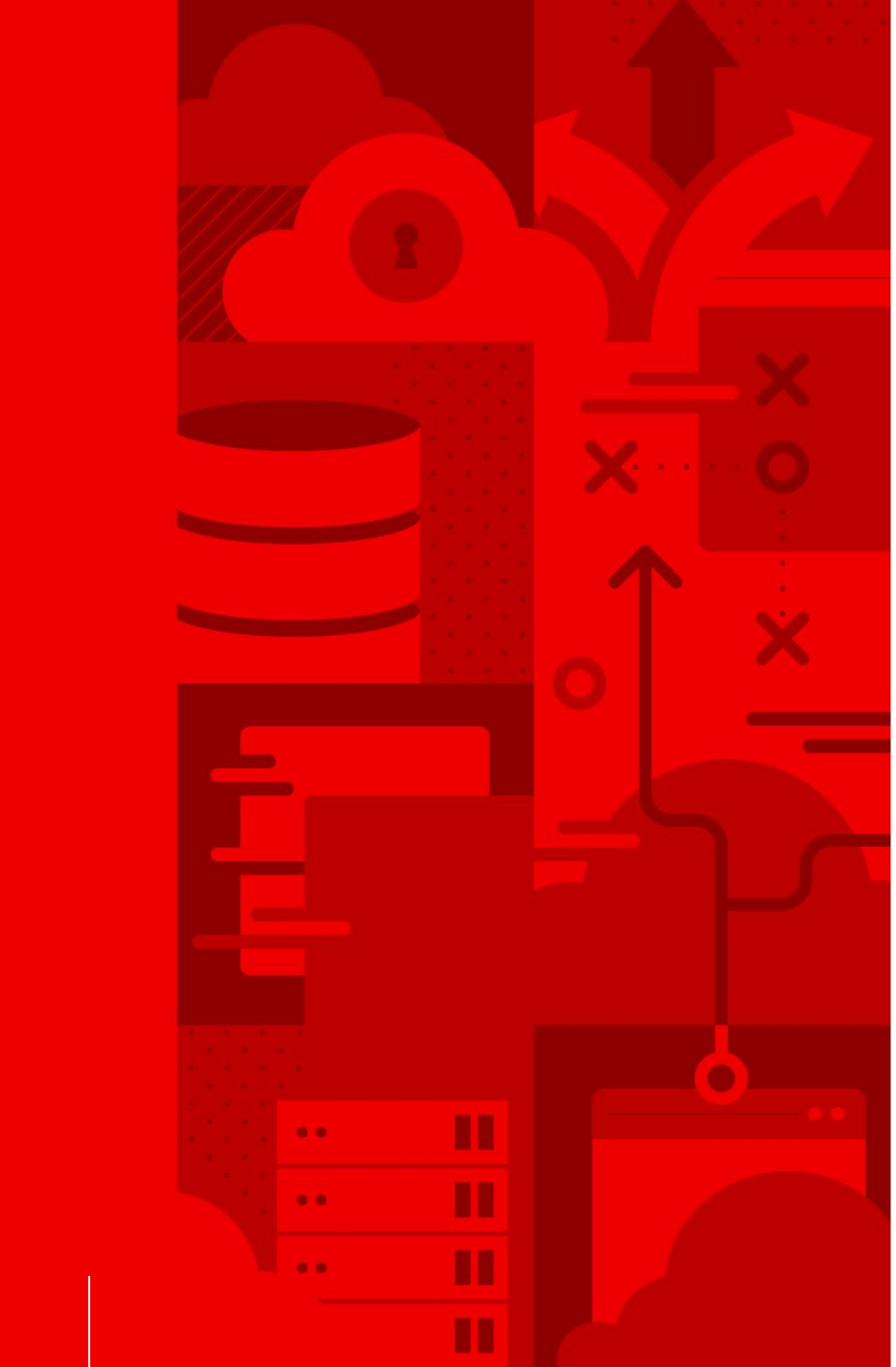
- Fetch the upstream repository.
- Create a branch in your local repository.
- Edit the favourite-editors.md file inside the **homework-projects** directory.
  - Add a name and a URL to your favourite text editor.
  - For example, [Vim] (https://www.vim.org/)
- Commit the change
- Push the changes to your fork and create a merge request against the main repository.
- Tag **@domiborges** and **@PetrsGamers** to review and merge your contribution.

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