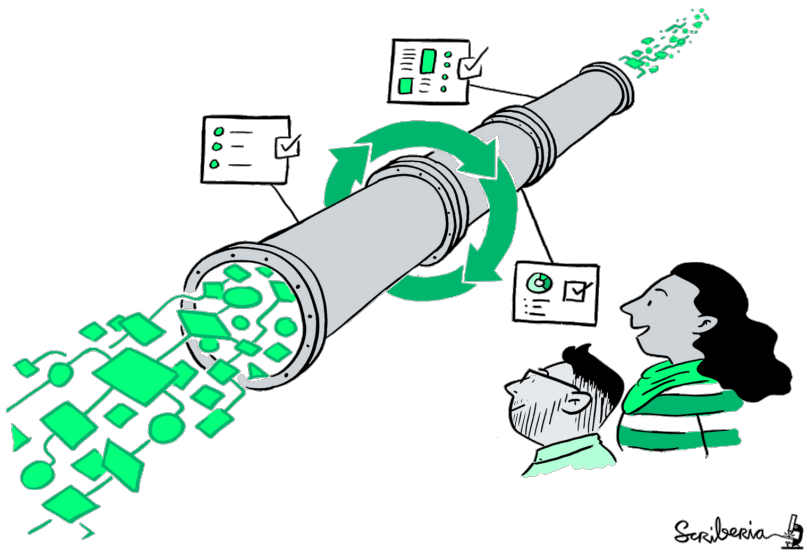
Training on Web Scraping Prices for CPI

How to RAP

Christophe Bontemps & Serge Goussev

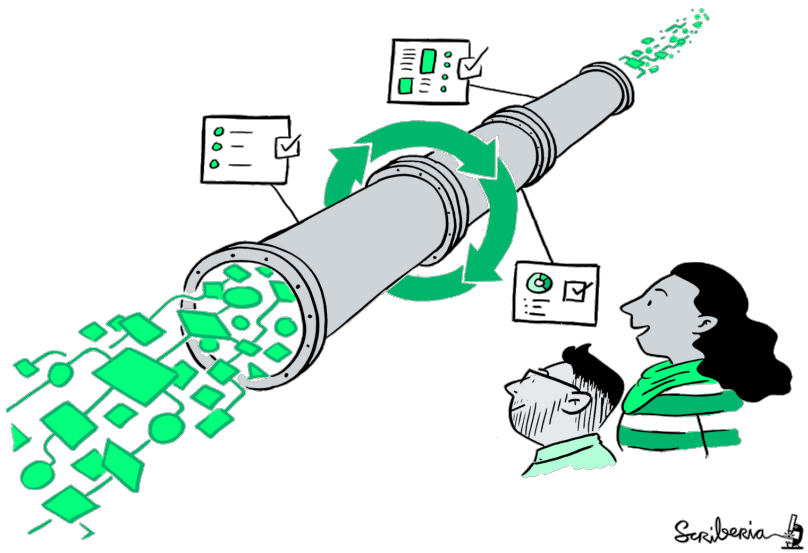
Reproducible Analytical Pipeline

- It is a process



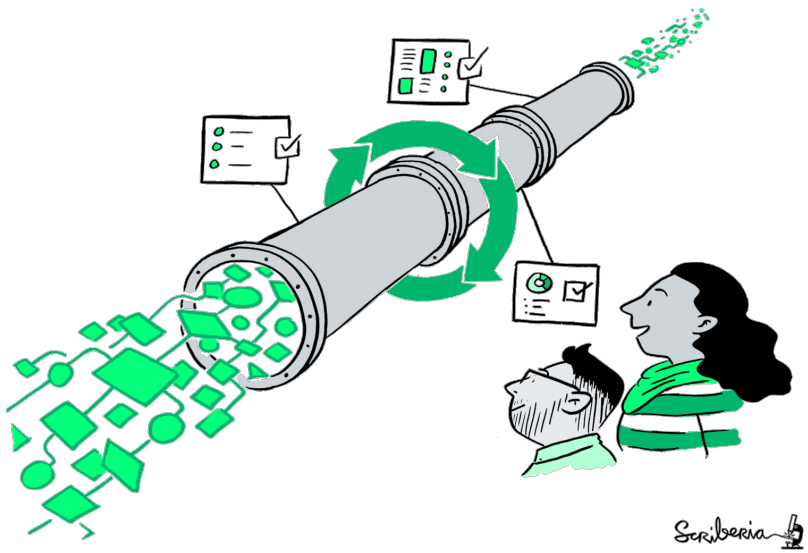
Reproducible Analytical Pipeline

- It is a process
- It is (*quite*) automated



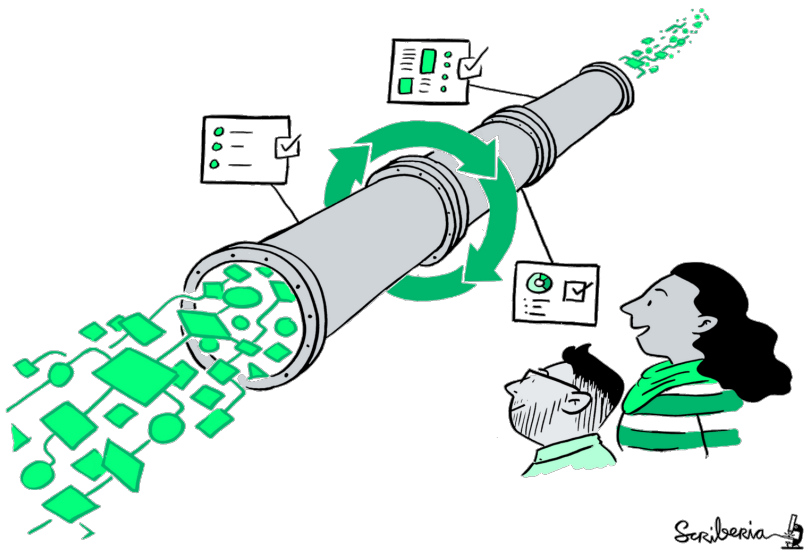
Reproducible Analytical Pipeline

- It is a process
- It is (*quite*) automated
- It is (*easily*) reproducible



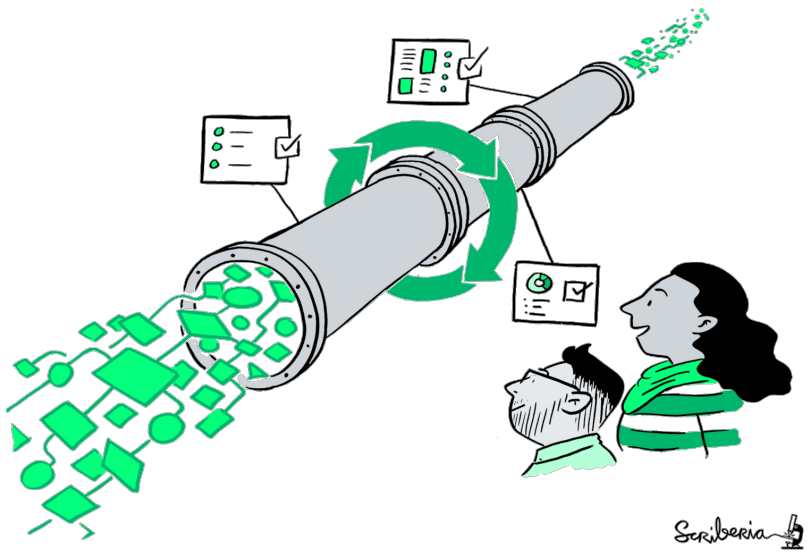
Reproducible Analytical Pipeline

- It is a process
- It is (*quite*) automated
- It is (*easily*) reproducible
- It minimises mistakes



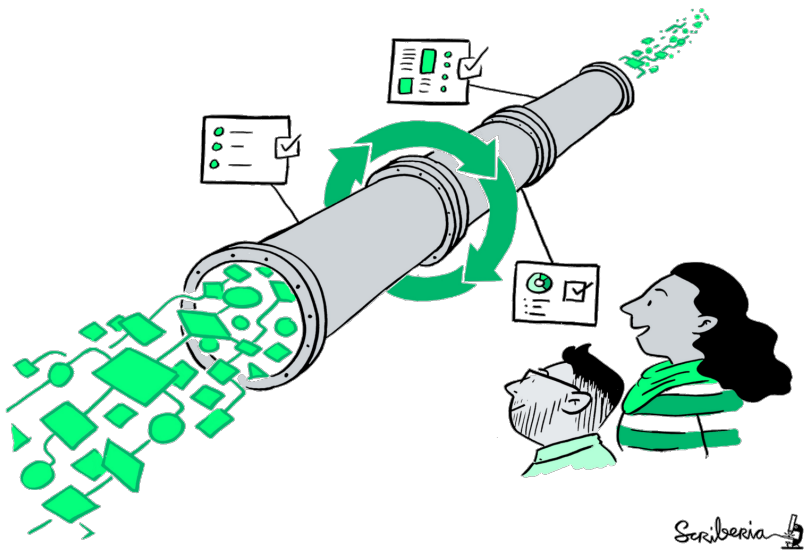
Reproducible Analytical Pipeline

- It is a process
- It is (*quite*) automated
- It is (*easily*) reproducible
- It minimises mistakes
- It leads to fast processes (see Vanuatu Experience)



Reproducible Analytical Pipeline

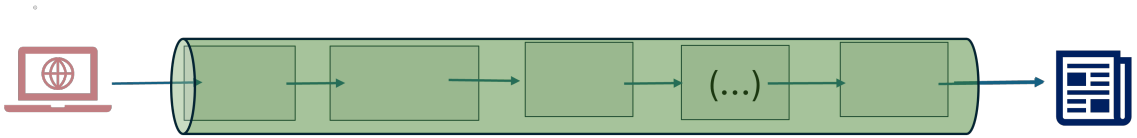
- It is a process
- It is (*quite*) automated
- It is (*easily*) reproducible
- It minimises mistakes
- It leads to fast processes (see Vanuatu Experience)
- It builds trust



What does a RAP look like?

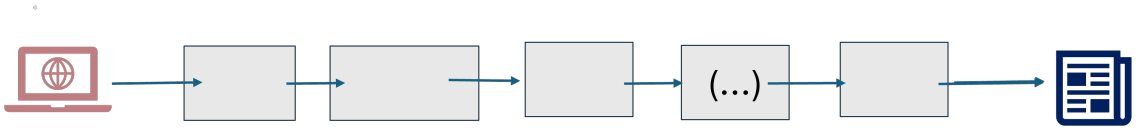
Idealy, Input (website) and output (report) are linked

What does a RAP look like?



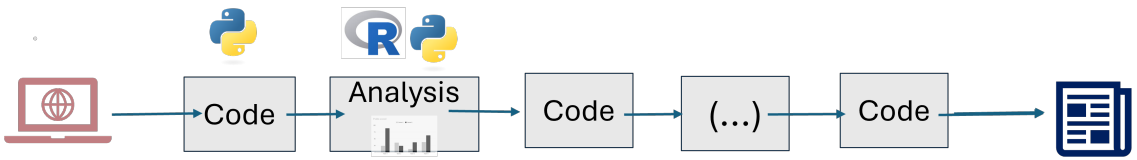
Many steps are needed to create a report

What does a RAP look like?



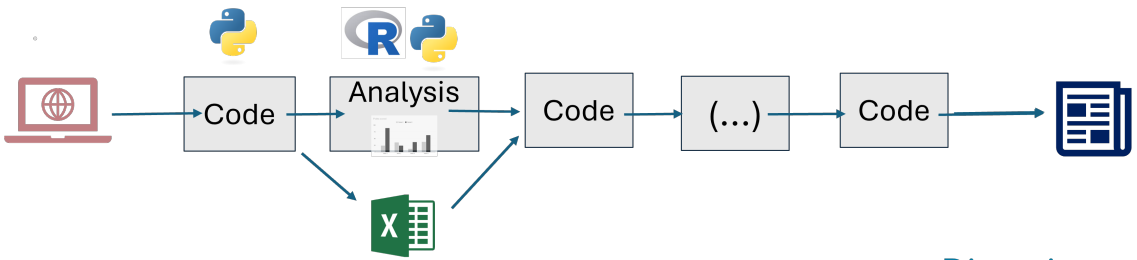
All steps should be linked in a structured process

What does a RAP look like?



And only through code (Python, R, *Stata*), no copy/paste

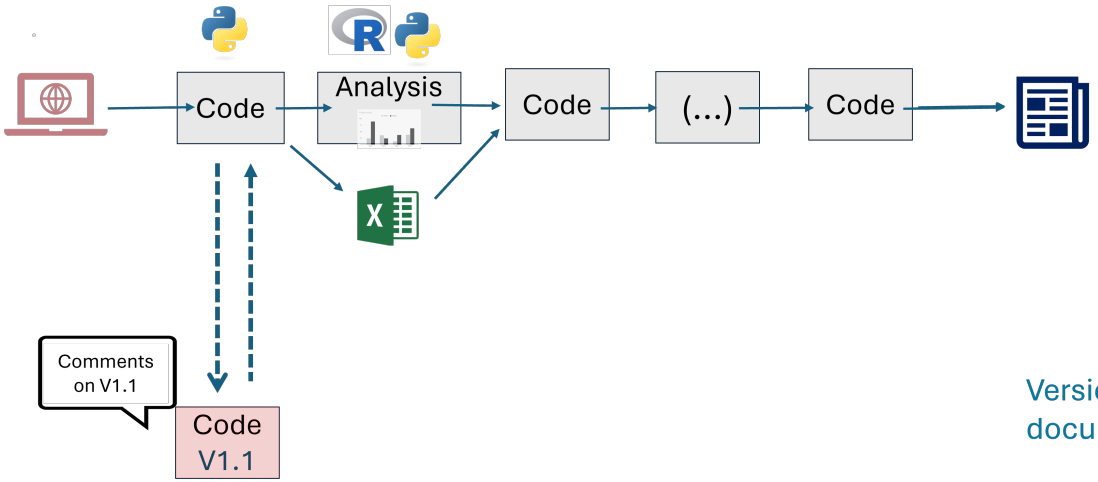
What does a RAP look like?



Direct input-output of data

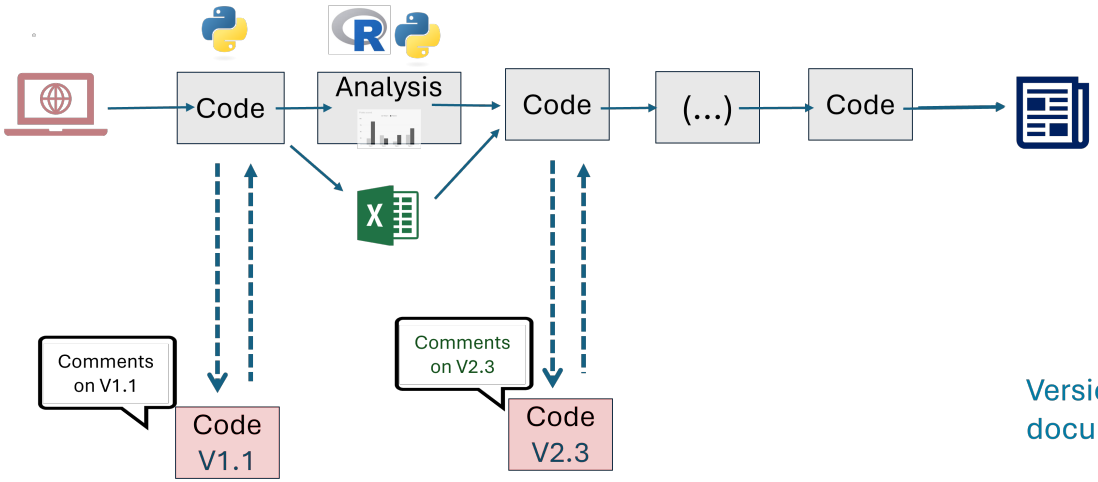
There may be side-products, but with explicit output-input links

What does a RAP look like?



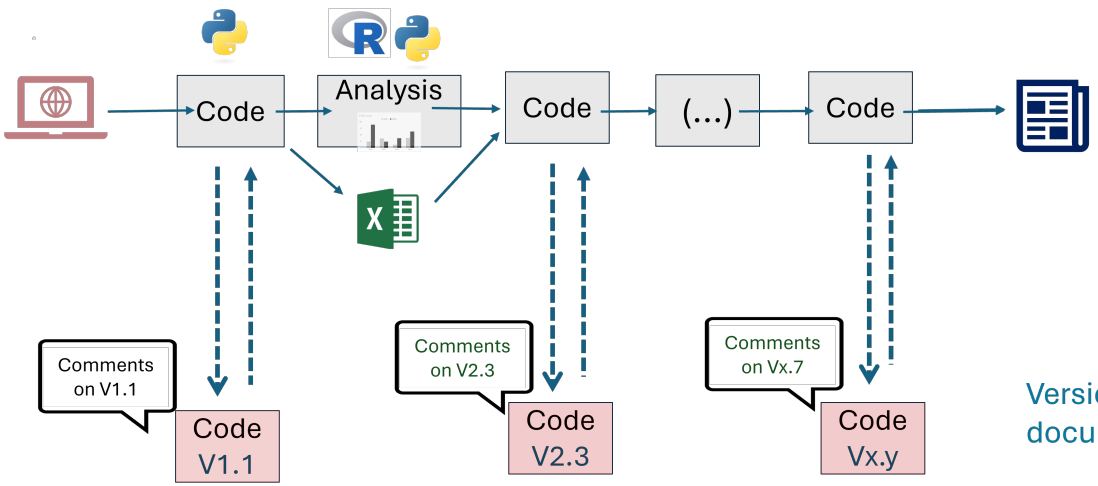
If needed, code can be updated (new versions)

What does a RAP look like?



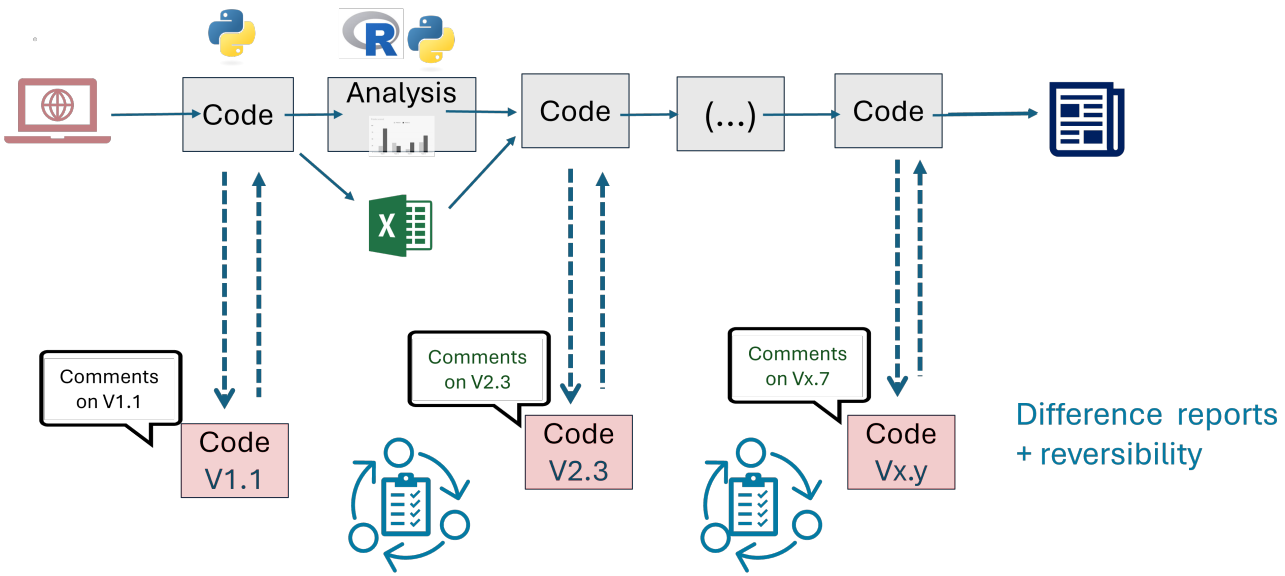
And comments added for each change

What does a RAP look like?



Documentation on the process builds up with code changes

What does a RAP look like?



Other contributors are welcome!

3 MAIN PRINCIPLES:

3 MAIN PRINCIPLES:

1. Organize your work

3 MAIN PRINCIPLES:

1. Organize your work
2. Code for others (including your future self)

3 MAIN PRINCIPLES:

1. Organize your work
2. Code for others (including your future self)
3. DRY: **D**o not **R**epeat **Y**ourself

3 MAIN PRINCIPLES:

1. Organize your work
2. Code for others (including your future self)
3. DRY: **D**o not **R**epeat **Y**ourself

3 MAIN PRINCIPLES:

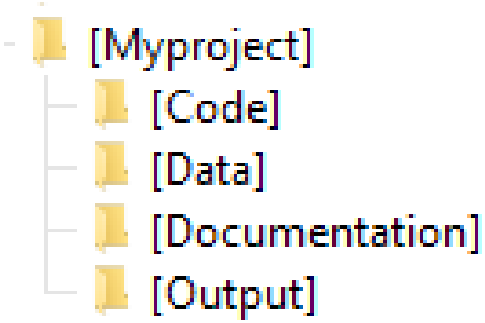
1. Organize your work
2. Code for others (including your future self)
3. DRY: **D**o not **R**epeat **Y**ourself

Apply this in context (colleagues, code, software,...)

Organize your work

Organize your work

- Have a clear directory structure



Example of a well-organized directory structure.

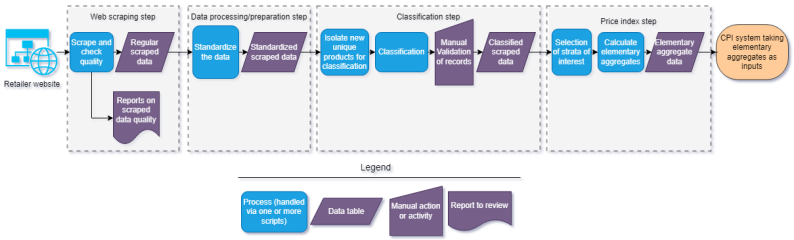
Organize your work

- Have a clear directory structure
- Use naming conventions

```
01_Scraping_data.ipynb
02_Cleaning_data.ipynb
03_Classification.ipynb
04_Stats_Tables.ipynb
04_Price_CPI.ipynb
```

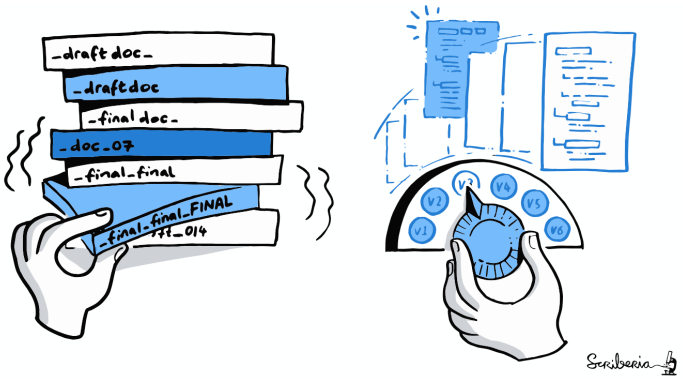
Organize your work

- Have a clear directory structure
- Use naming conventions
- Keep track of the workflow



Organize your work

- Have a clear directory structure
- Use naming conventions
- Keep track of the workflow
- Use a version control system



Code for others

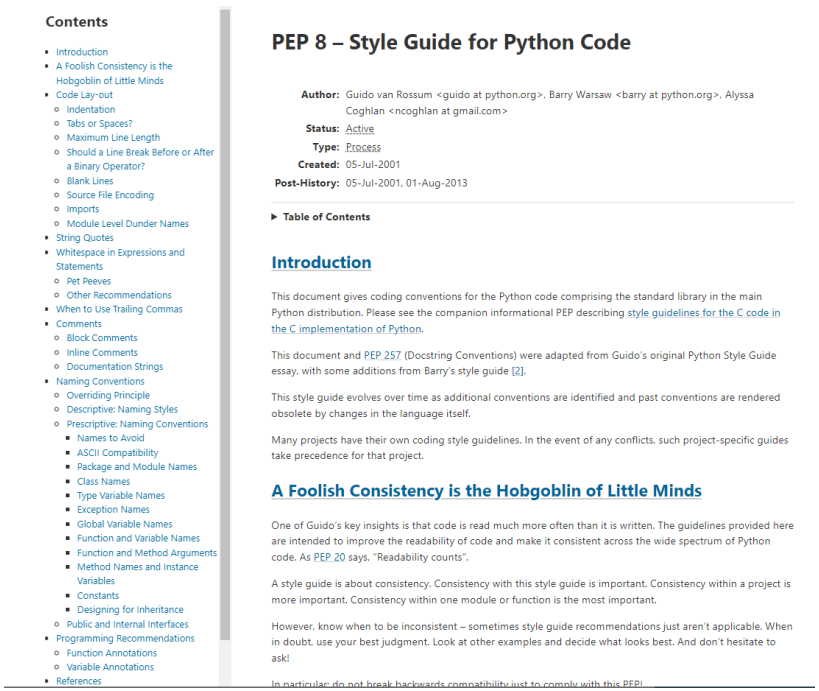
Code for others

- Program with style

"(. . .) code is read much more often than it is written"
Guido van Rossum (2013 -PEP8)

Code for others

- Program with style
- Use conventions (PEP8)



Contents

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Alyssa Coghlan <ncoghlan at gmail.com>

Status: Active
Type: Process
Created: 05-Jul-2001
Post-History: 05-Jul-2001, 01-Aug-2013

▸ Table of Contents

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python.

This document and PEP 257 (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

A Foolish Consistency is the Hobgoblin of Little Minds

One of Guido's key insights is that code is read much more often than it is written. The guidelines provided here are intended to improve the readability of code and make it consistent across the wide spectrum of Python code. As PEP 20 says, "Readability counts".

A style guide is about consistency. Consistency with this style guide is important. Consistency within a project is more important. Consistency within one module or function is the most important.

However, know when to be inconsistent – sometimes style guide recommendations just aren't applicable. When in doubt, use your best judgment. Look at other examples and decide what looks best. And don't hesitate to ask!

In particular: do not break backwards compatibility just to comply with this PEP!

Code for others

- Program with style
- Use conventions (PEP8)
- Avoid ambiguities

Usual

```
df['sex'] = np.where(df['gender']
== '1001', 1, 2)
```

Better

```
df['female'] =
np.where(df['gender'] == '1001',
1, 0)
df['male'] =
np.where(df['gender'] != '1001',
1, 0)
```

DO NOT REPEAT YOURSELF

DO NOT REPEAT YOURSELF

- Create reusable objects

Usual

```
Current_Data = Mydata[Mydata['year'] == 2023]
```

Better

```
Current_year = 2023
Current_Data= Mydata[Mydata['year'] ==
Current_year]
```

Usual

```
# - Exports for Beef -
data = Mydata[Mydata['export'] == 'Beef']
plt.plot(data['Year'], data['Value'])
plt.title('Export for Beef')
# - Also for Kava -
data = Mydata[Mydata['export'] == 'Kava']
plt.plot(data['Year'], data['Value'])
plt.title('Export for Kava')
# - Also for ...  -
```

DO NOT REPEAT YOURSELF

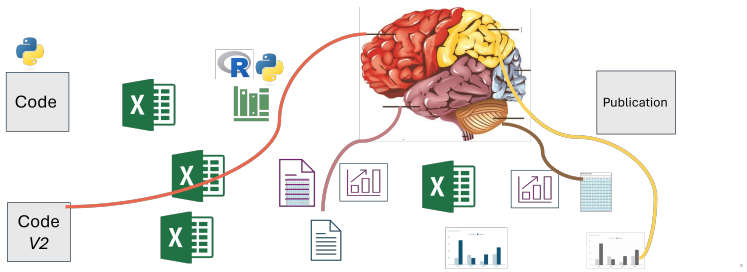
- Create reusable objects
- Avoid repetitions

Better

```
# Defining a generic function
def plot_export(export_type):
 data = Mydata[Mydata['export'] ==
                                    export_type]
 plt.plot(data['Year'], data['Value'])
 plt.title(f'Export for {export_type}')
 plt.show()
# Applying function to several products
plot_export("Beef")
plot_export("Kava")
```

DO NOT REPEAT YOURSELF

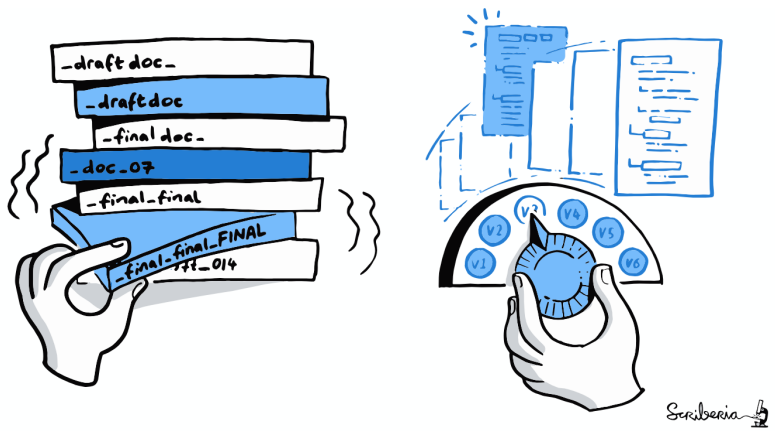
- Create reusable objects
- Avoid repetitions
- Automate as much as you can



VERSION CONTROL KEEPS TRACKS OF YOUR WORK

Tracking three **W** questions:

What changes?



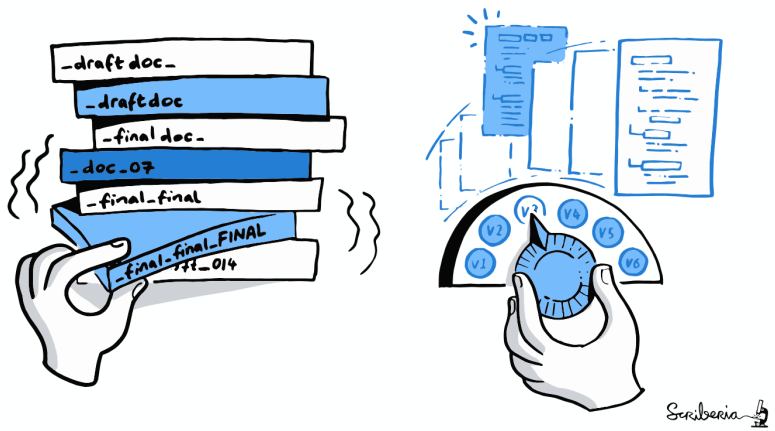
Source: The Turing Way project

VERSION CONTROL KEEPS TRACKS OF YOUR WORK

Tracking three **W** questions:

What changes?

Who made the changes?



Source: The Turing Way project

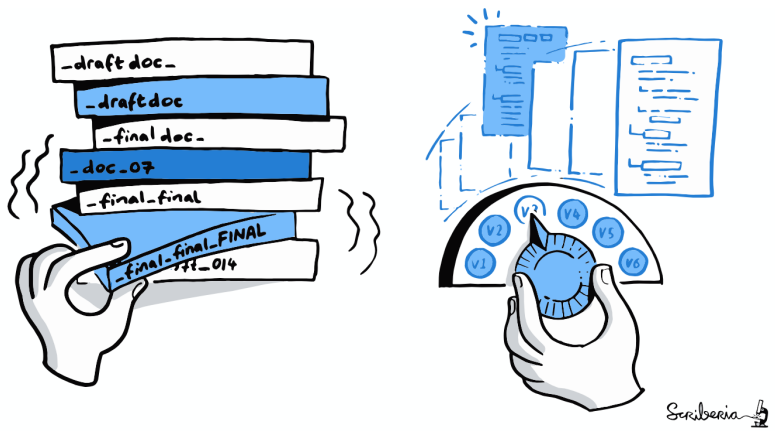
VERSION CONTROL KEEPS TRACKS OF YOUR WORK

Tracking three **W** questions:

What changes?

Who made the changes?

When were the changes made?



Source: The Turing Way project

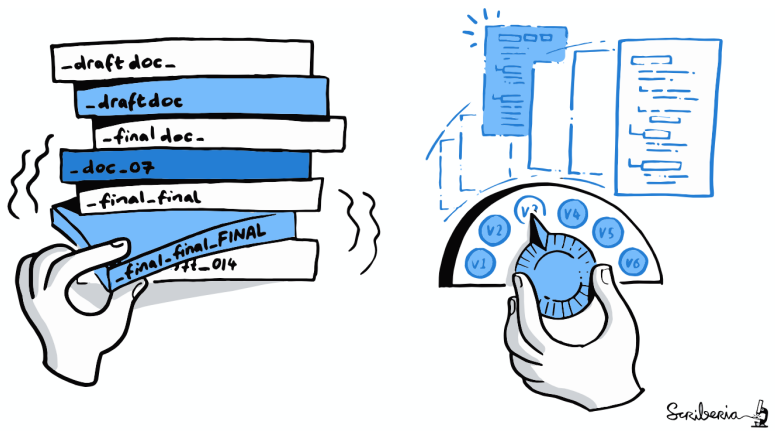
VERSION CONTROL KEEPS TRACKS OF YOUR WORK

Tracking three **W** questions:

What changes?

Who made the changes?

When were the changes made?



Source: The Turing Way project

Git & GitHub

Git & GitHub are different tools

Git & GitHub

Git & GitHub are different tools

- Git is a software

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode
- Git is complex and unfriendly!

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode
- Git is complex and unfriendly!

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode
- Git is complex and unfriendly!

- GitHub is a platform

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode
- Git is complex and unfriendly!

- GitHub is a platform
- GitHub needs an account

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode
- Git is complex and unfriendly!

- GitHub is a platform
- GitHub needs an account
- GitHub can only be accessed remotely

Git & GitHub

Git & GitHub are different tools

- Git is a software
- Git needs to be installed
- Git works mostly in command mode
- Git is complex and unfriendly!

- GitHub is a platform
- GitHub needs an account
- GitHub can only be accessed remotely
- GitHub is (more) friendly

SETUP

What you'll need to do...

- Download Git

SETUP

What you'll need to do...

- Download Git
- Install Git on your computer

Setup

What you'll need to do...

- Download Git
- Install Git on your computer

↪ Depends on your OS

Setup

What you'll need to do...

- Download Git
- Install Git on your computer
  ↪ Depends on your OS
- Create a GitHub account

SETUP

What you'll need to do...

- Download Git
- Install Git on your computer
  - ↪ Depends on your OS
- Create a GitHub account
- Link Git with GitHub account

Setup

What you'll need to do...

- Download Git
- Install Git on your computer
  - ↪ Depends on your OS
- Create a GitHub account
- Link Git with GitHub account
- Link VStudio with Git

Setup

What you'll need to do...

- Download Git
- Install Git on your computer

↪ Depends on your OS

- Create a GitHub account
- Link Git with GitHub account
- Link VStudio with Git

↪ May need IT support

Setup

What you'll need to do...

- Download Git
- Install Git on your computer

↪ Depends on your OS

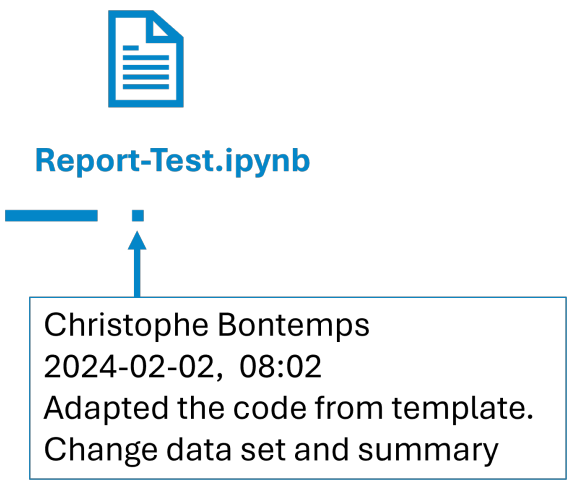
- Create a GitHub account
- Link Git with GitHub account
- Link VStudio with Git

↪ May need IT support

↪ Many tutorials exist

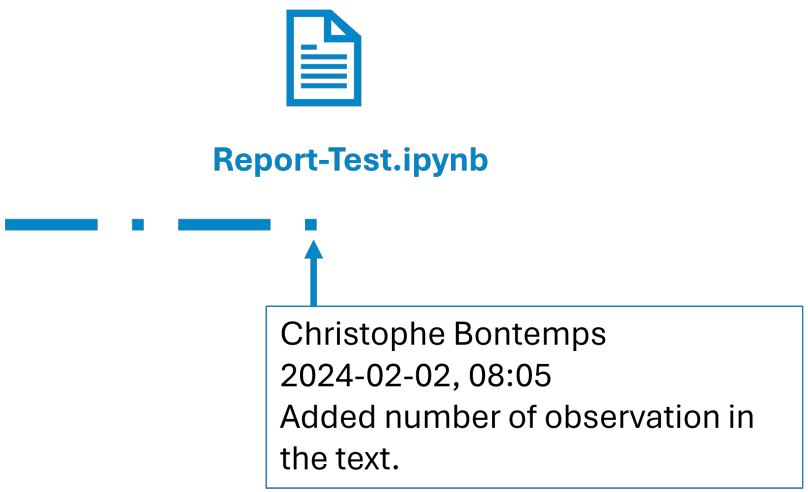
Version Control Main commands

commit for every change!



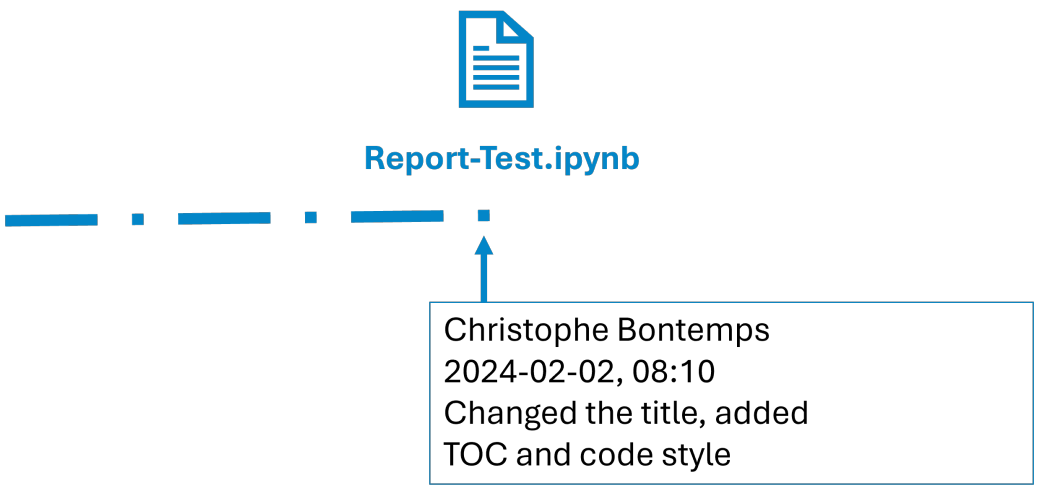
Version Control Main commands

commit for every change!



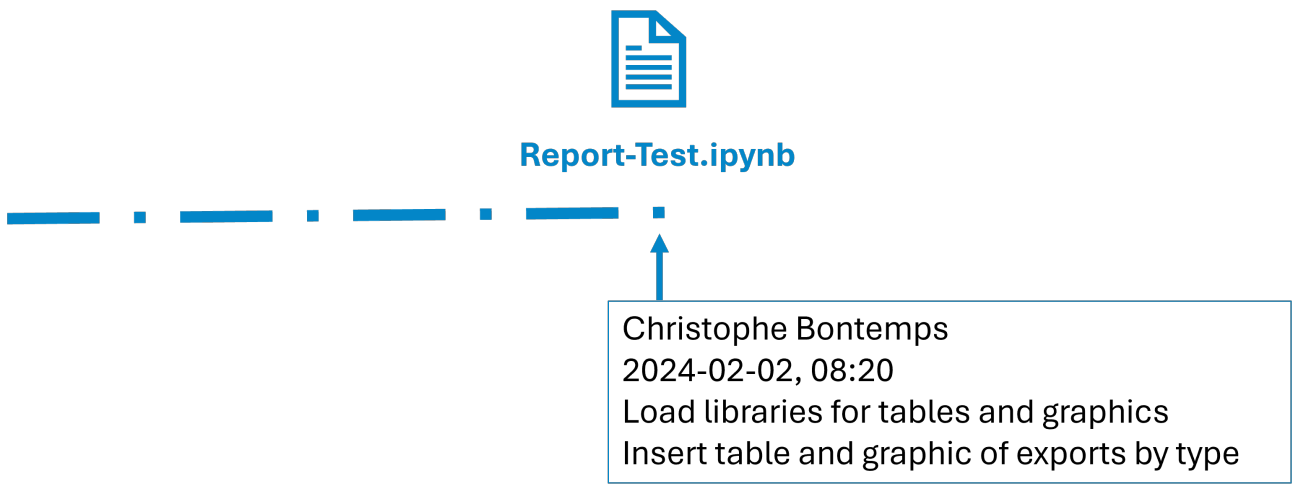
Version Control Main commands

commit for every change!



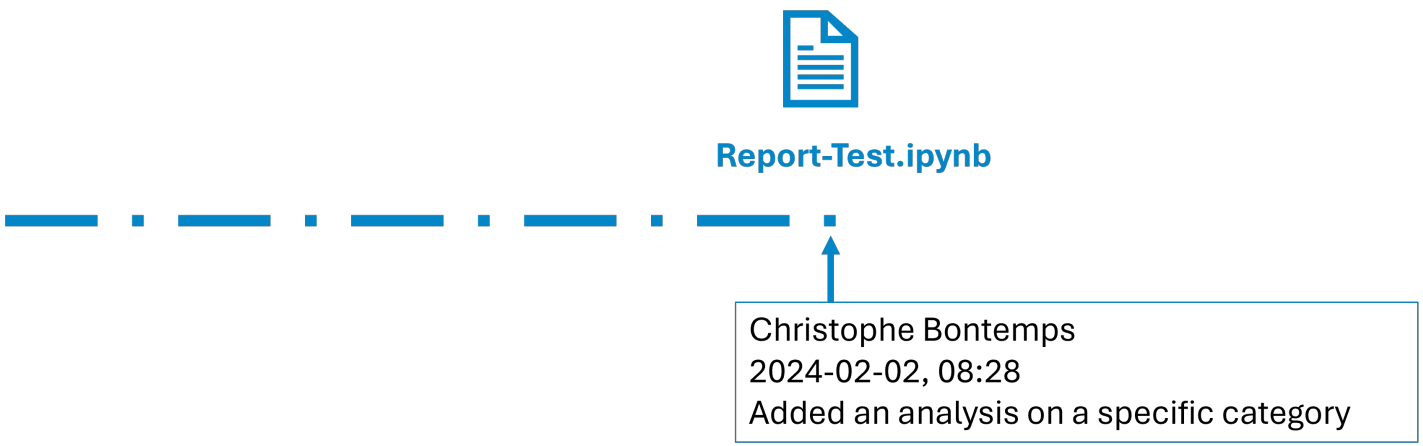
Version Control Main commands

commit for every change!



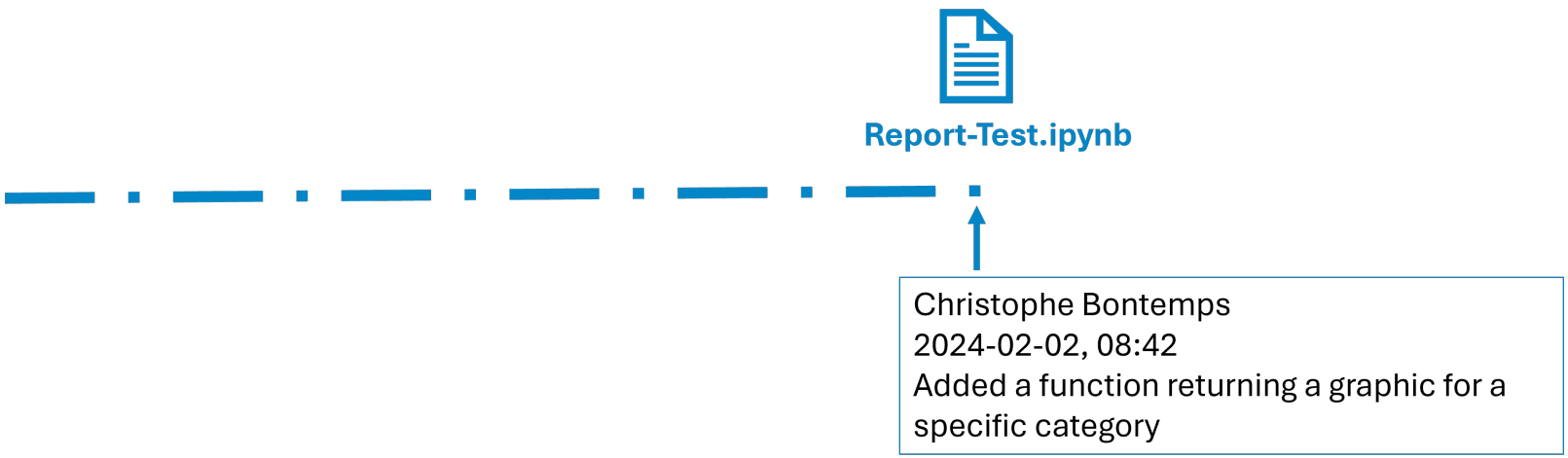
Version Control Main commands

commit for every change!



Version Control Main commands

commit for every change!



Version Control Main commands

commit for every change!

THE HISTORY OF THE FILE IS RECORDED!

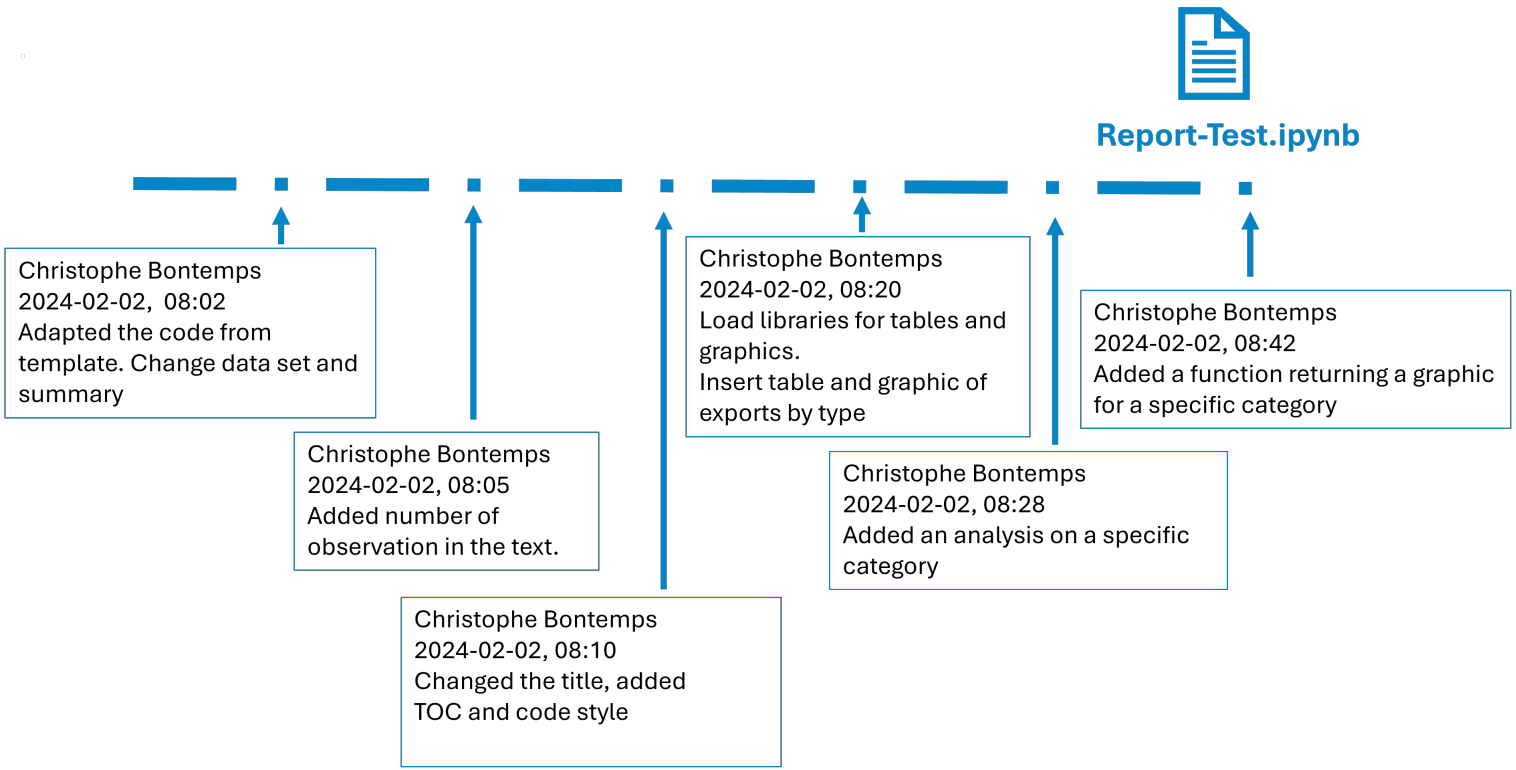
Each version is documented (with *commits*)

THE HISTORY OF THE FILE IS RECORDED!

Each version is documented (with *commits*)

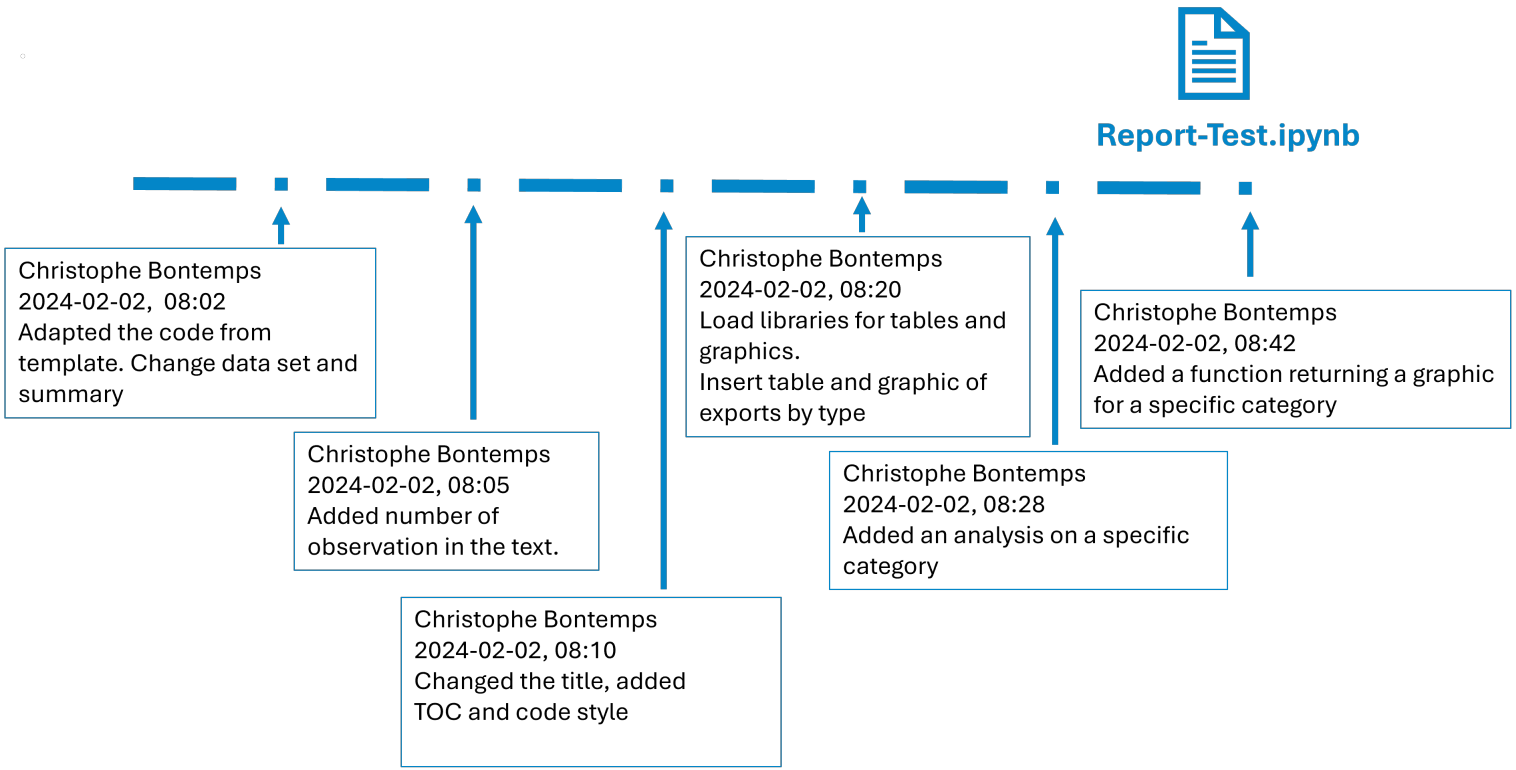
THE HISTORY OF THE FILE IS RECORDED!

Each version embeds the full history!



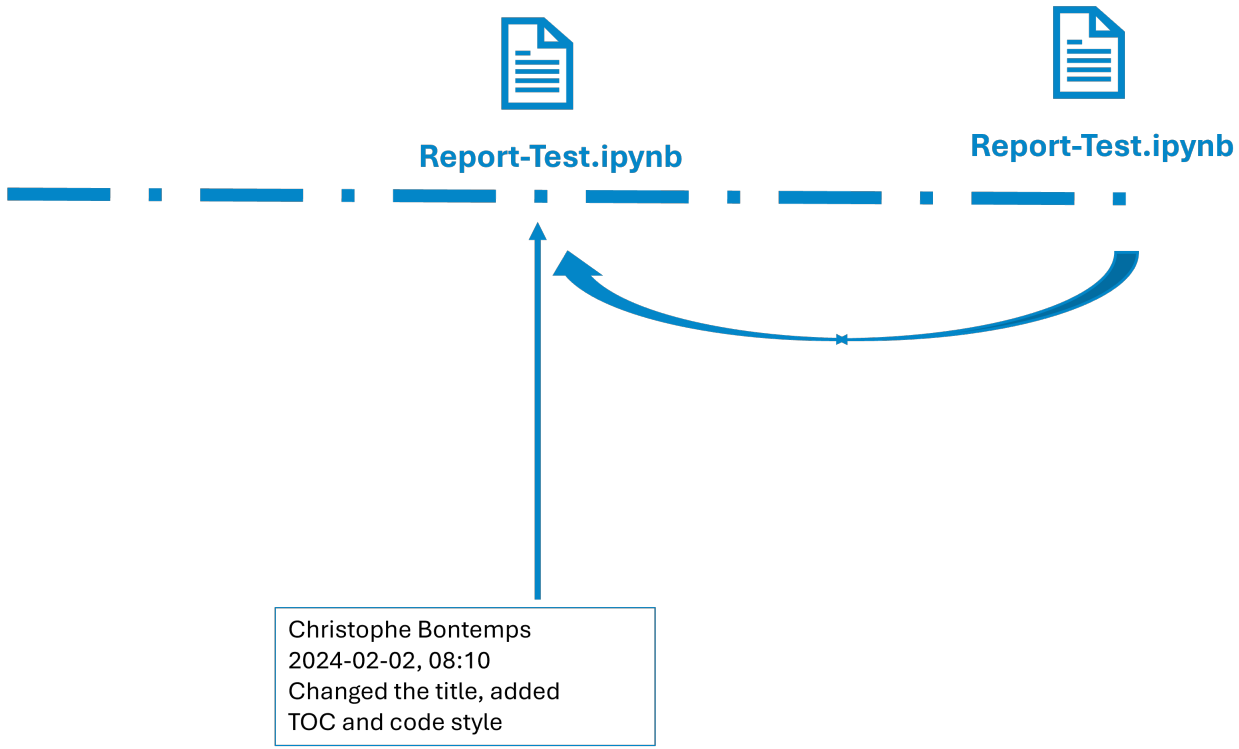
THE HISTORY OF THE FILE IS RECORDED!

Each version embeds the full history!



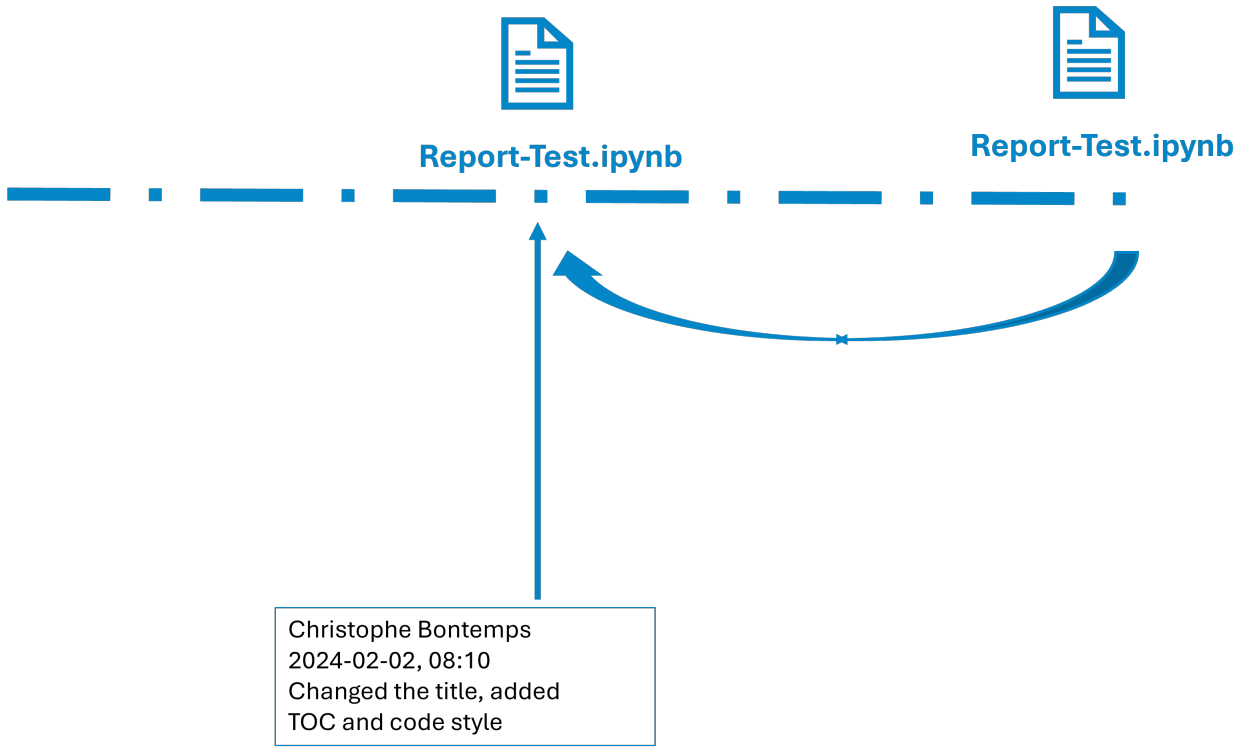
GOING BACK AND "UNDO" IS POSSIBLE

It is possible to review previous version...



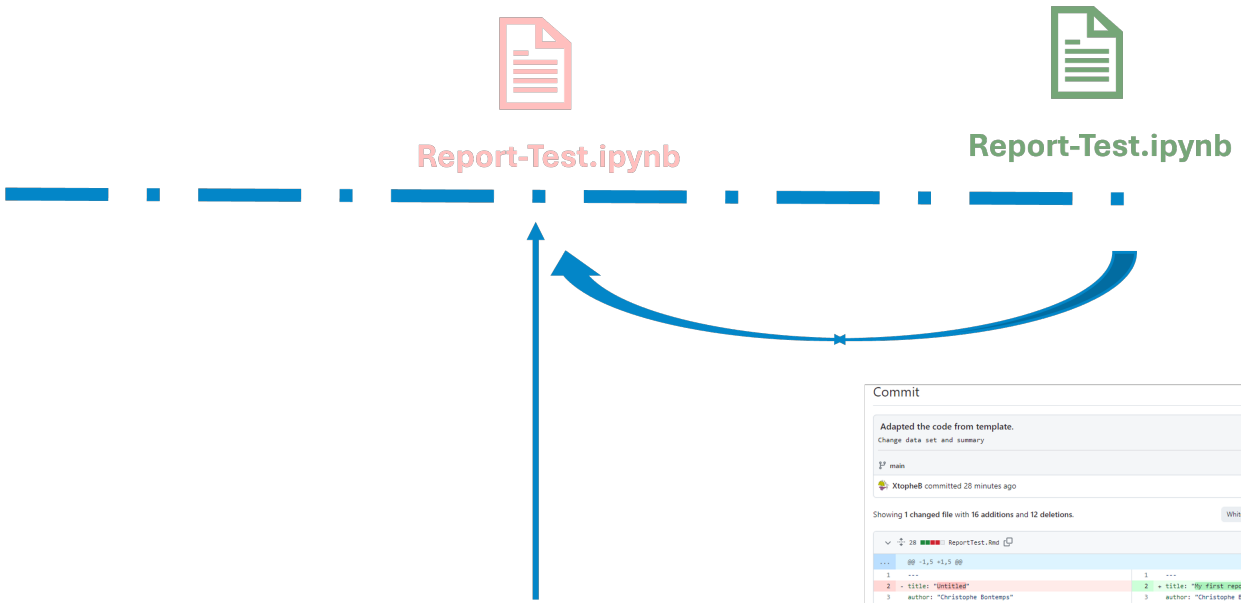
GOING BACK AND "UNDO" IS POSSIBLE

It is possible to review previous version...



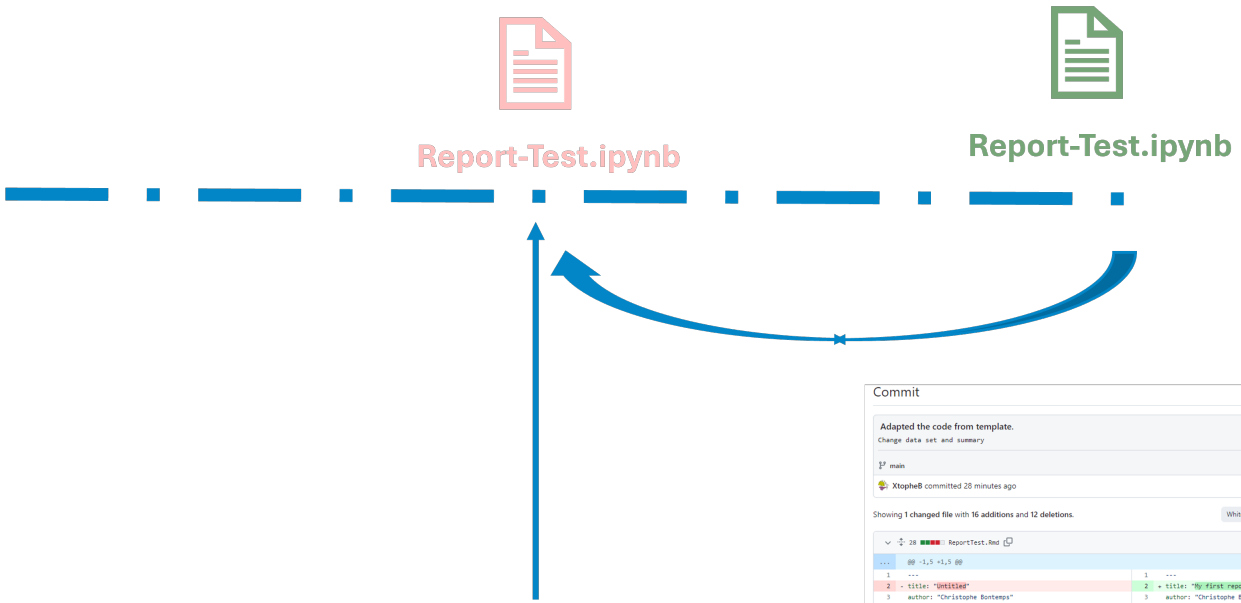
GOING BACK AND "UNDO" IS POSSIBLE

...to compare the changes...



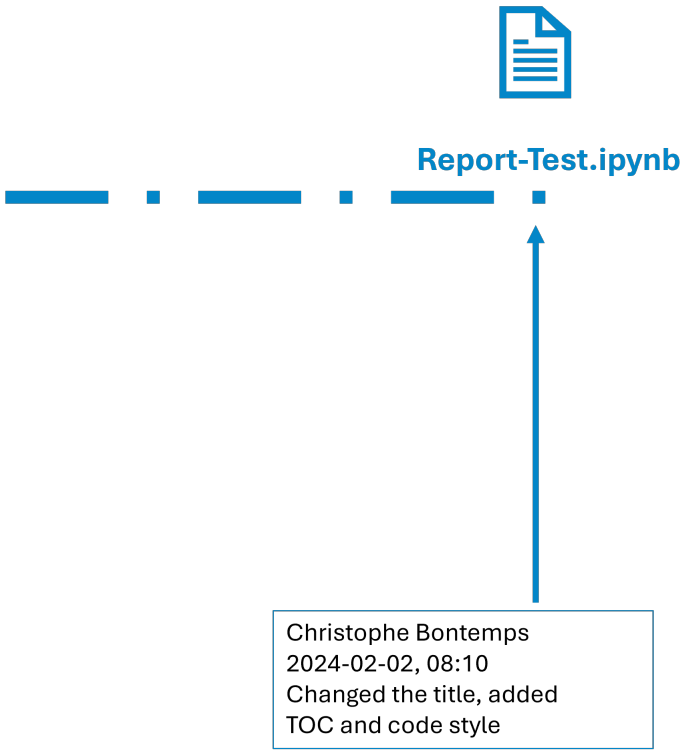
GOING BACK AND "UNDO" IS POSSIBLE

...to compare the changes...



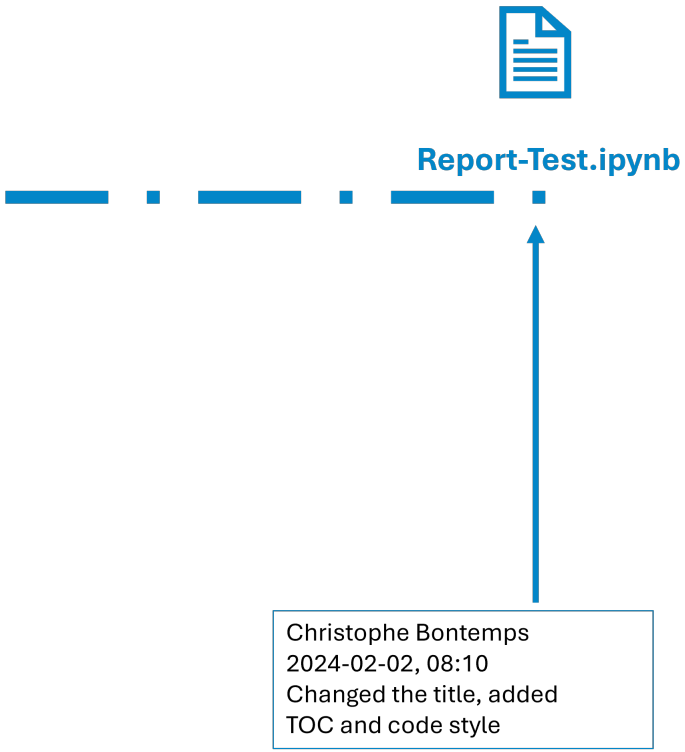
GOING BACK AND "UNDO" IS POSSIBLE

... and to revert to a previous version...



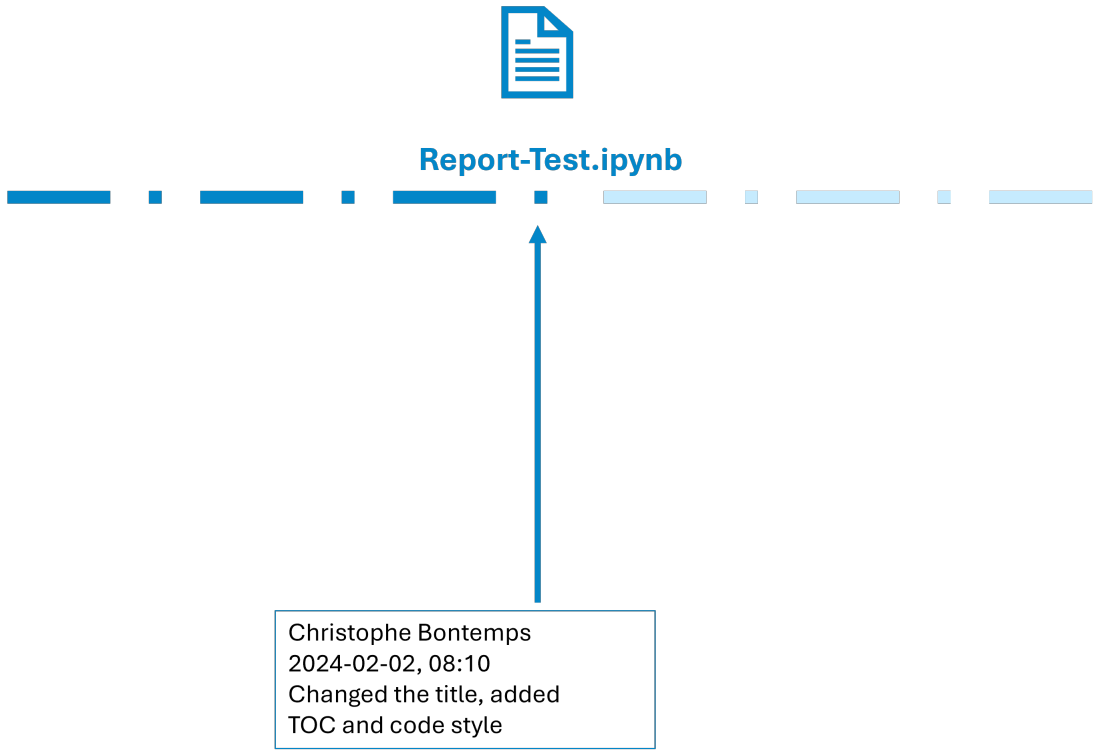
GOING BACK AND "UNDO" IS POSSIBLE

... and to revert to a previous version...



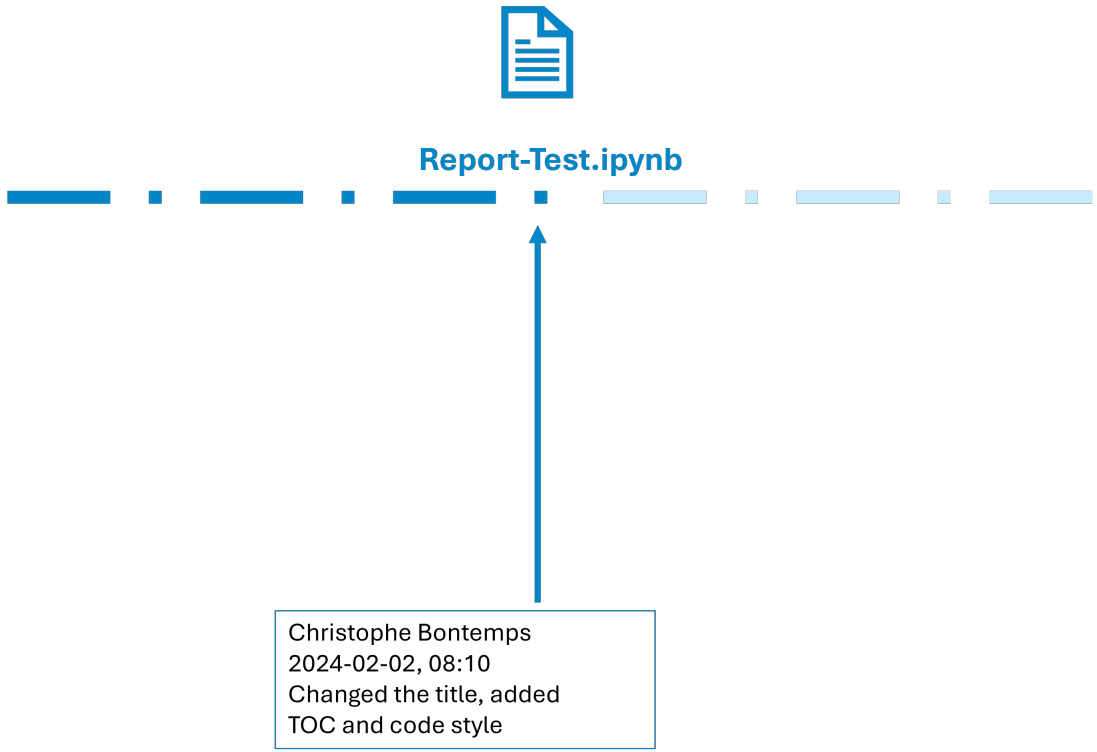
GOING BACK AND "UNDO" IS POSSIBLE

... or *undo* as if nothing happened



GOING BACK AND "UNDO" IS POSSIBLE

... or *undo* as if nothing happened



GIT FOR COLLABORATING

Alice and **Bob** work on the same file

GIT FOR COLLABORATING

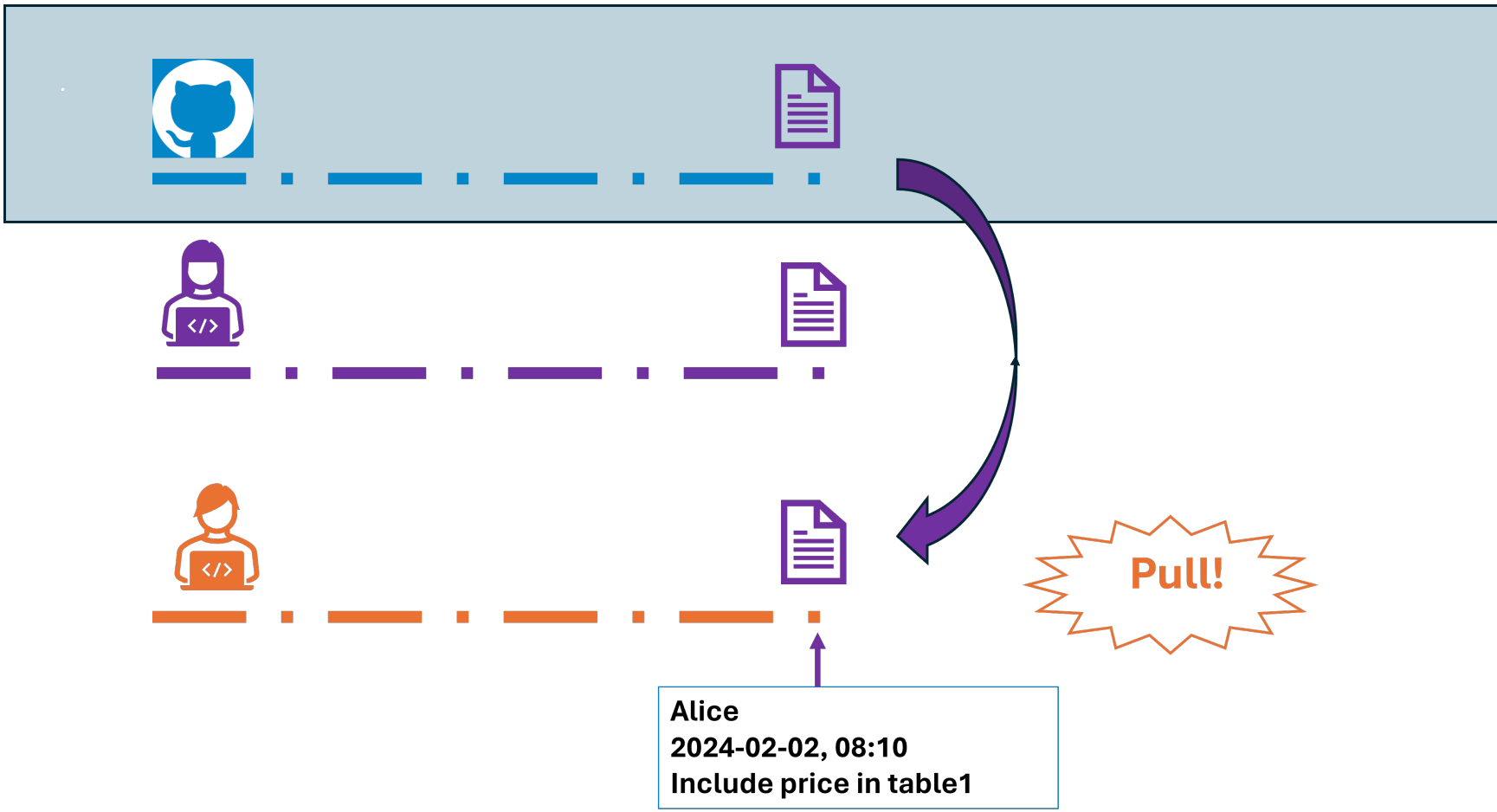
Alice and **Bob** work on the same file

GIT FOR COLLABORATING

Alice and **Bob** work on the same file

GIT FOR COLLABORATING

Alice and **Bob** work on the same file



Important notions

Spaces

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

- **Commit message:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

- **Commit message:**

↪ A concise description of the changes made

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

- **Commit message:**

↪ A concise description of the changes made

- **Push:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

- **Commit message:**

↪ A concise description of the changes made

- **Push:**

↪ Action of sending modified file(s) to GitHub

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

- **Commit message:**

↪ A concise description of the changes made

- **Push:**

↪ Action of sending modified file(s) to GitHub

- **Pull:**

IMPORTANT NOTIONS

Spaces

- **Local working directory:**

↪ Folder with your file(s)

- **Staging area:**

↪ Local "space" where modified file(s) are stored before commit

- **Remote repository:**

↪ Folder on GitHub platform

Actions

- **Commit:**

↪ A snapshot of file changes

- **Commit message:**

↪ A concise description of the changes made

- **Push:**

↪ Action of sending modified file(s) to GitHub

- **Pull:**

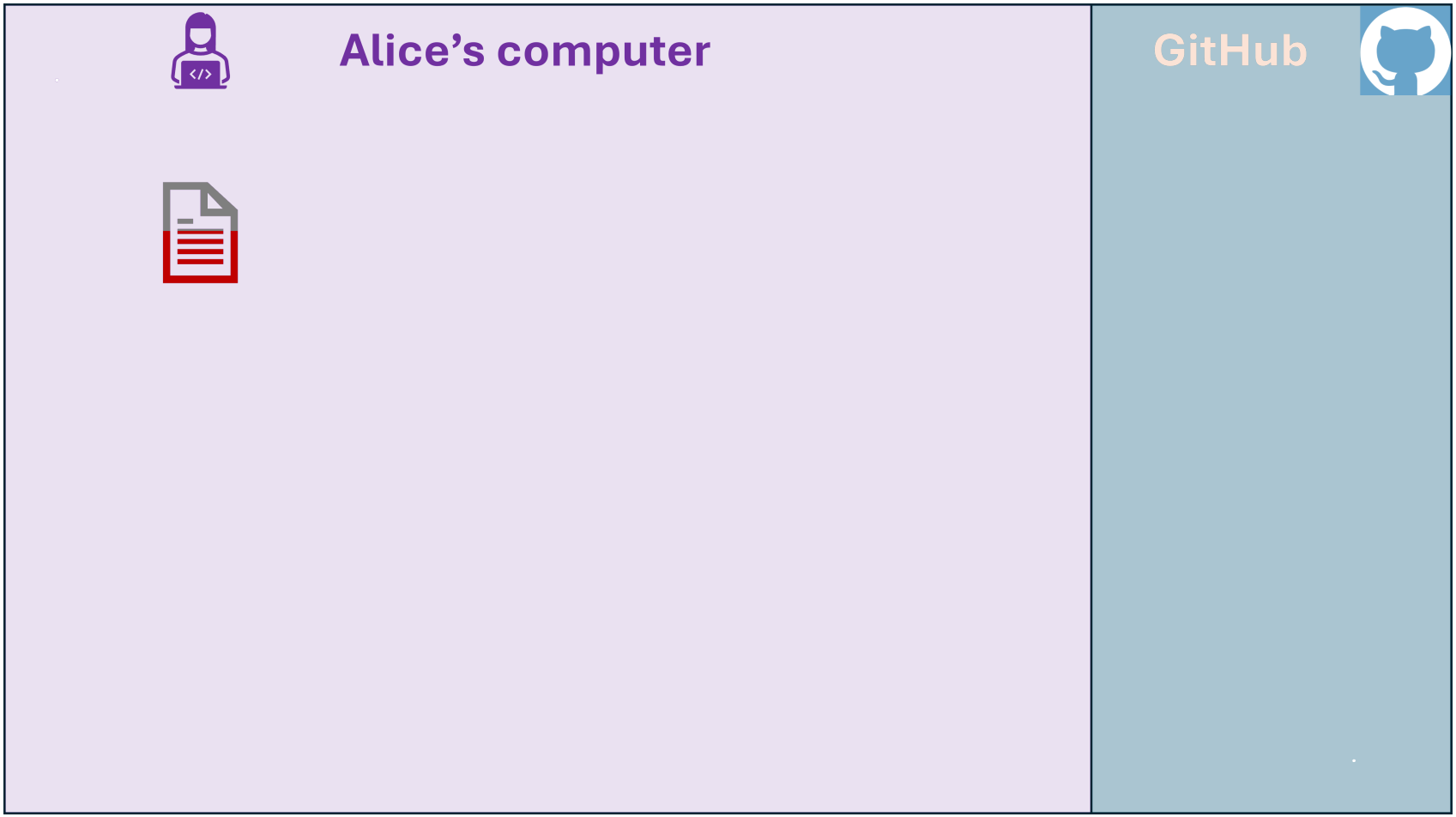
↪ Action of retrieving modified file(s) to local working directory

Git for collaborating: Details

Alice and **Bob** work on the same file (Details)

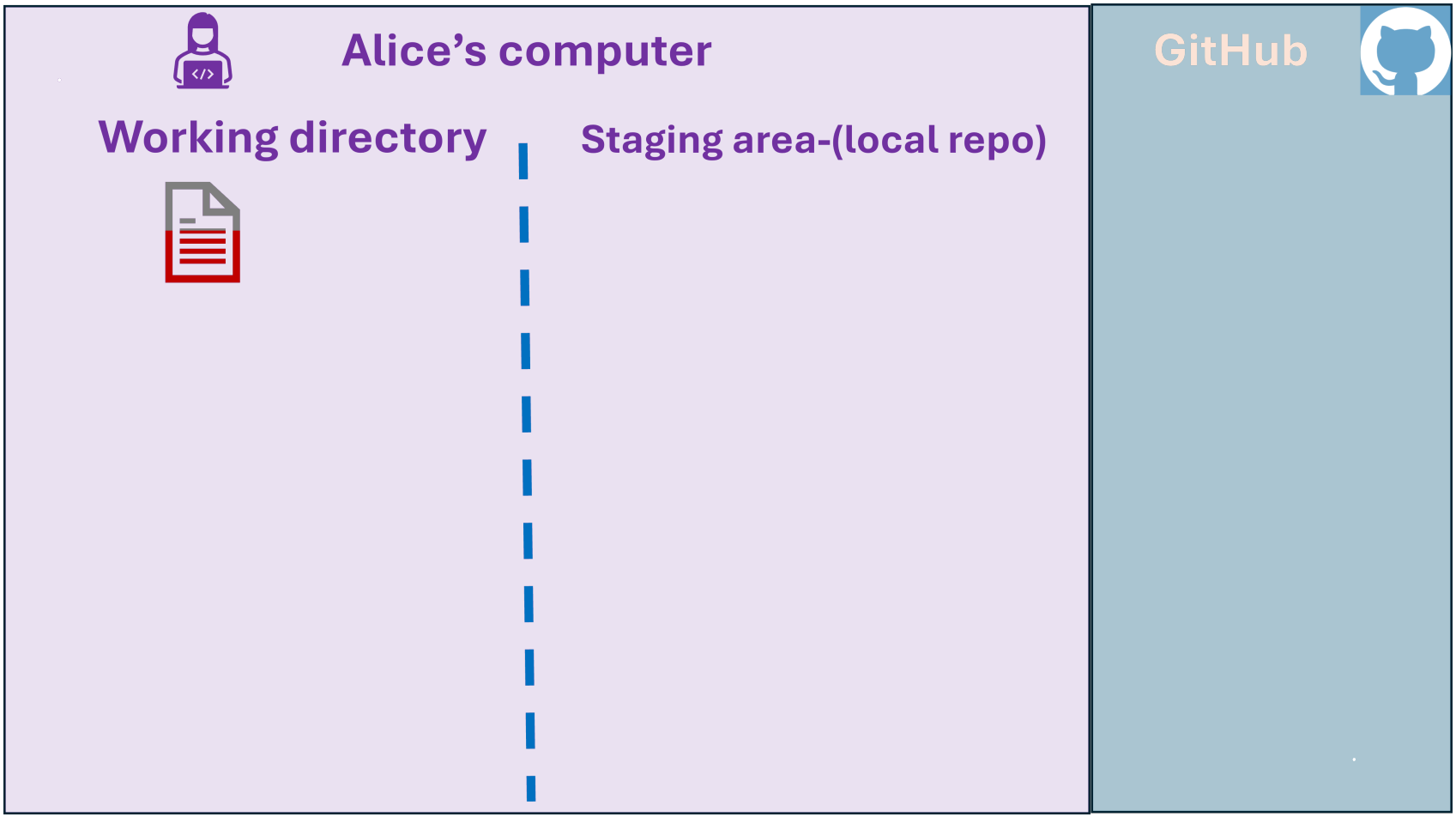
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



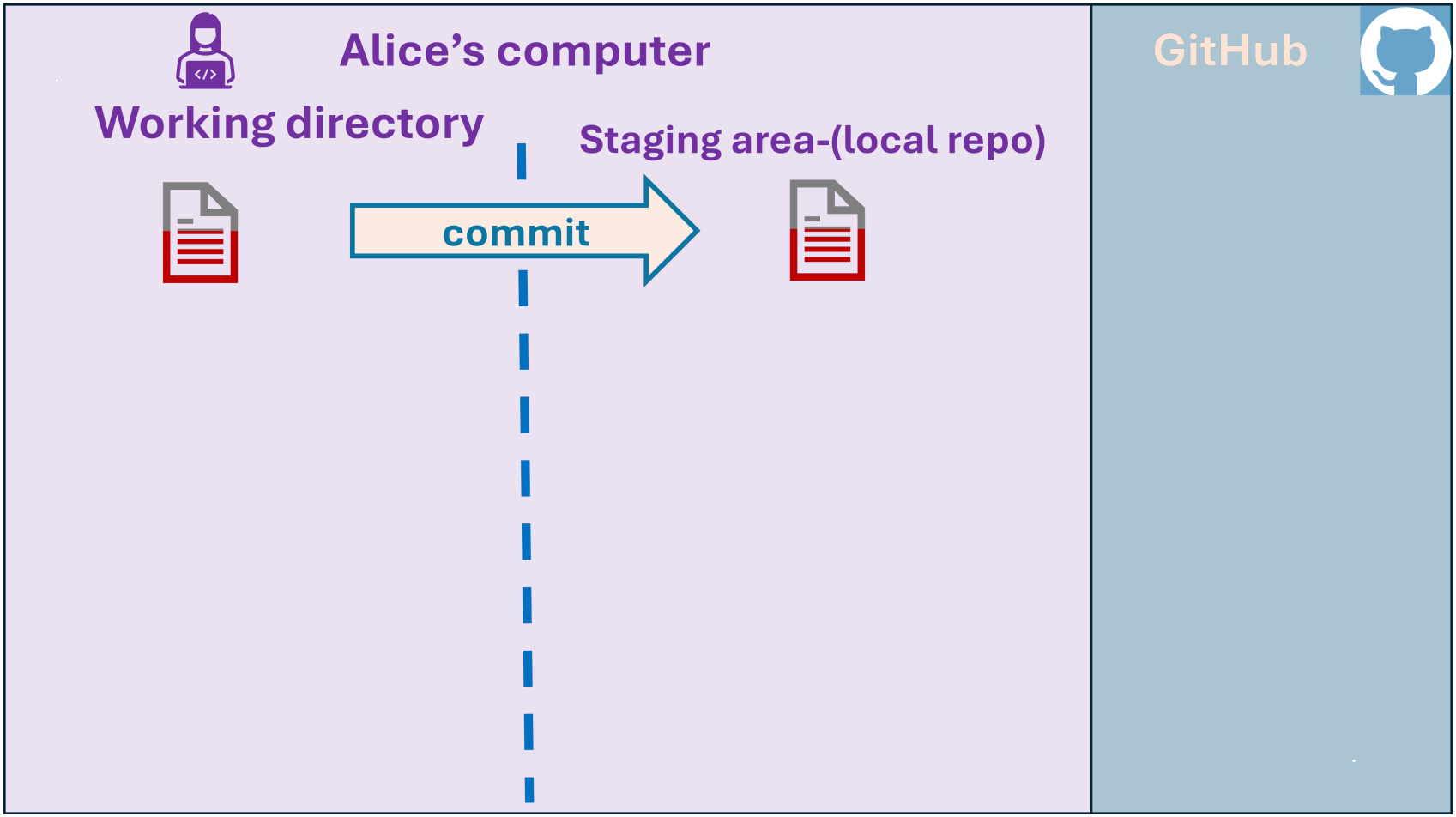
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



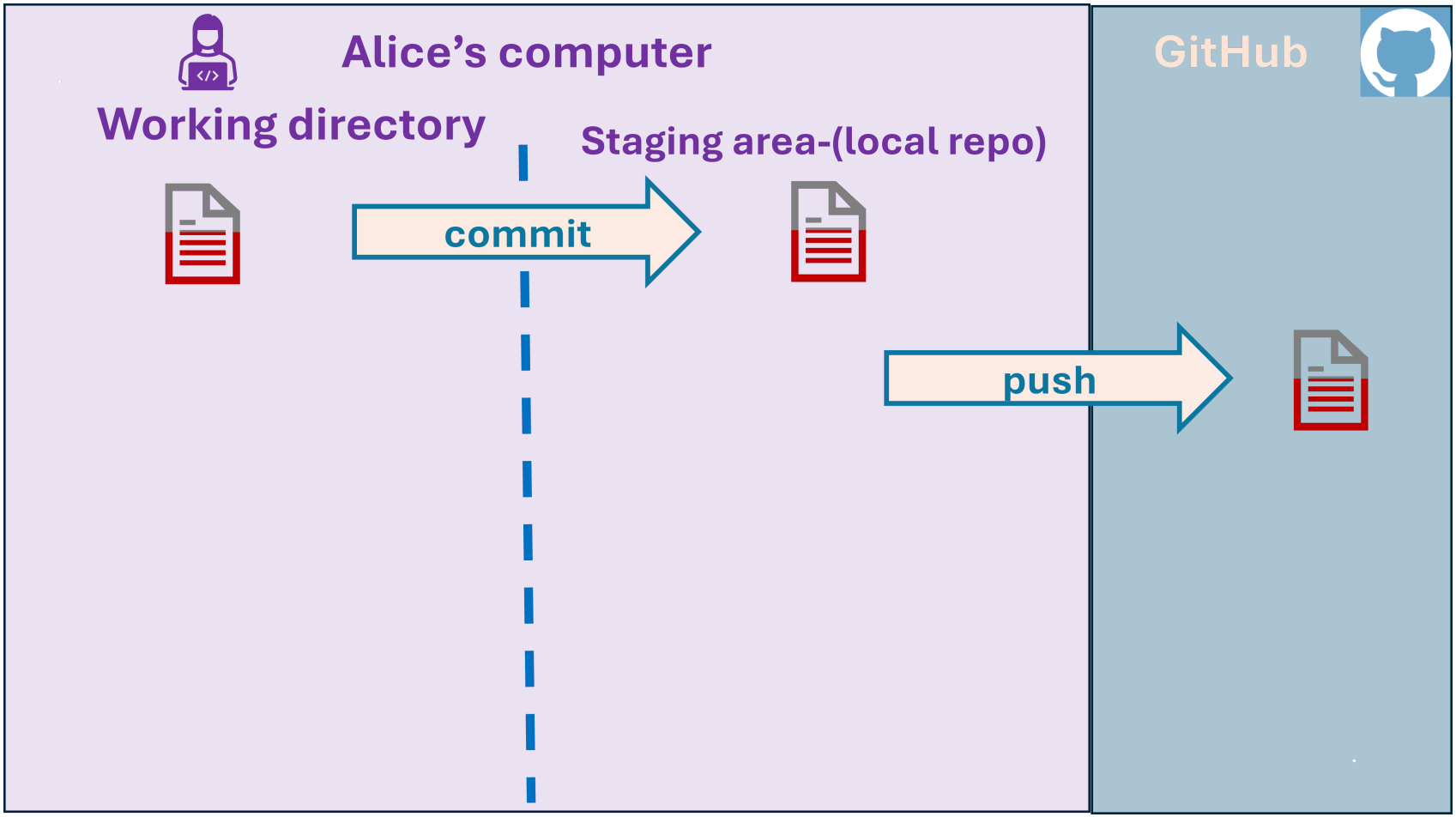
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



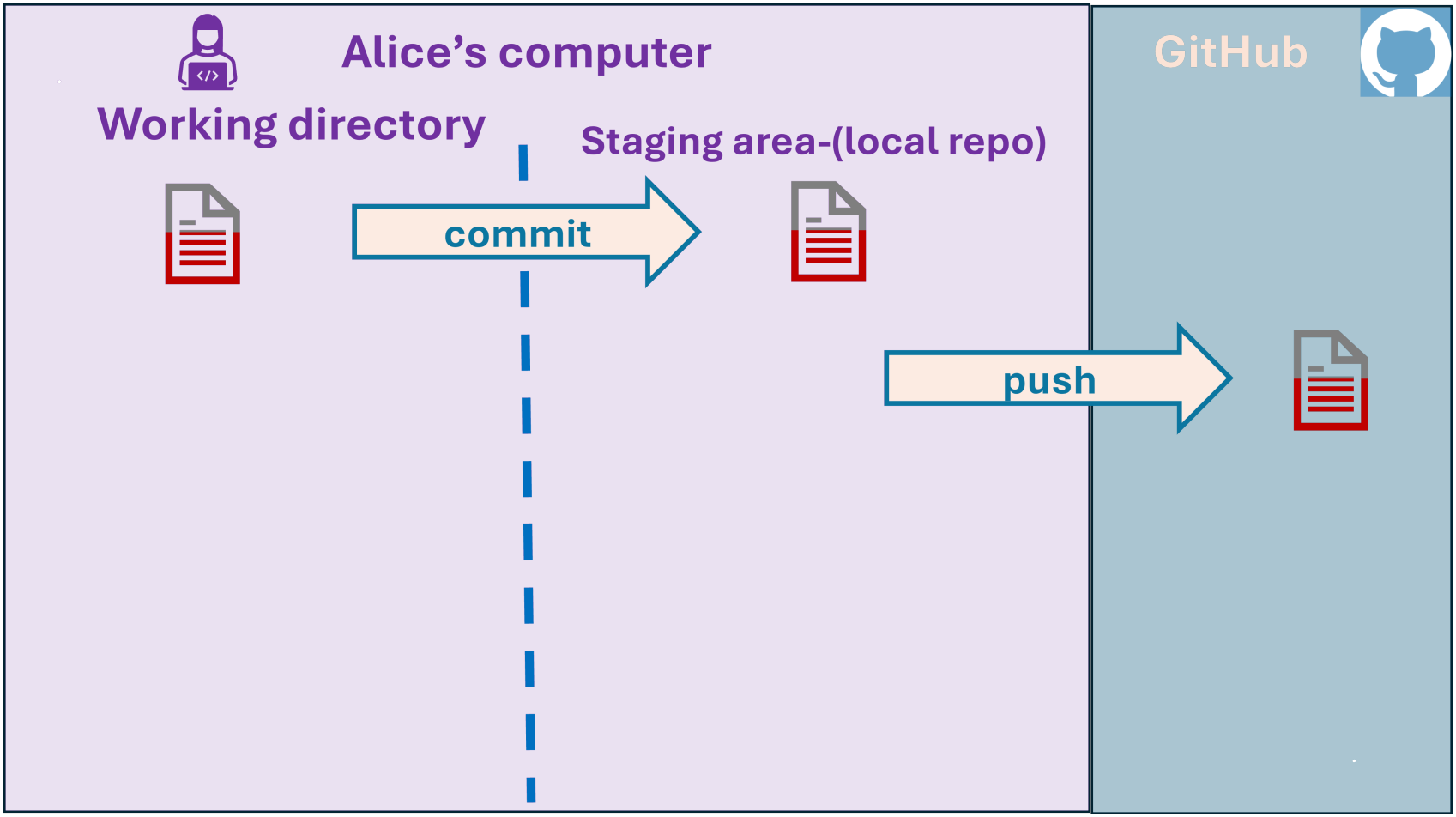
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



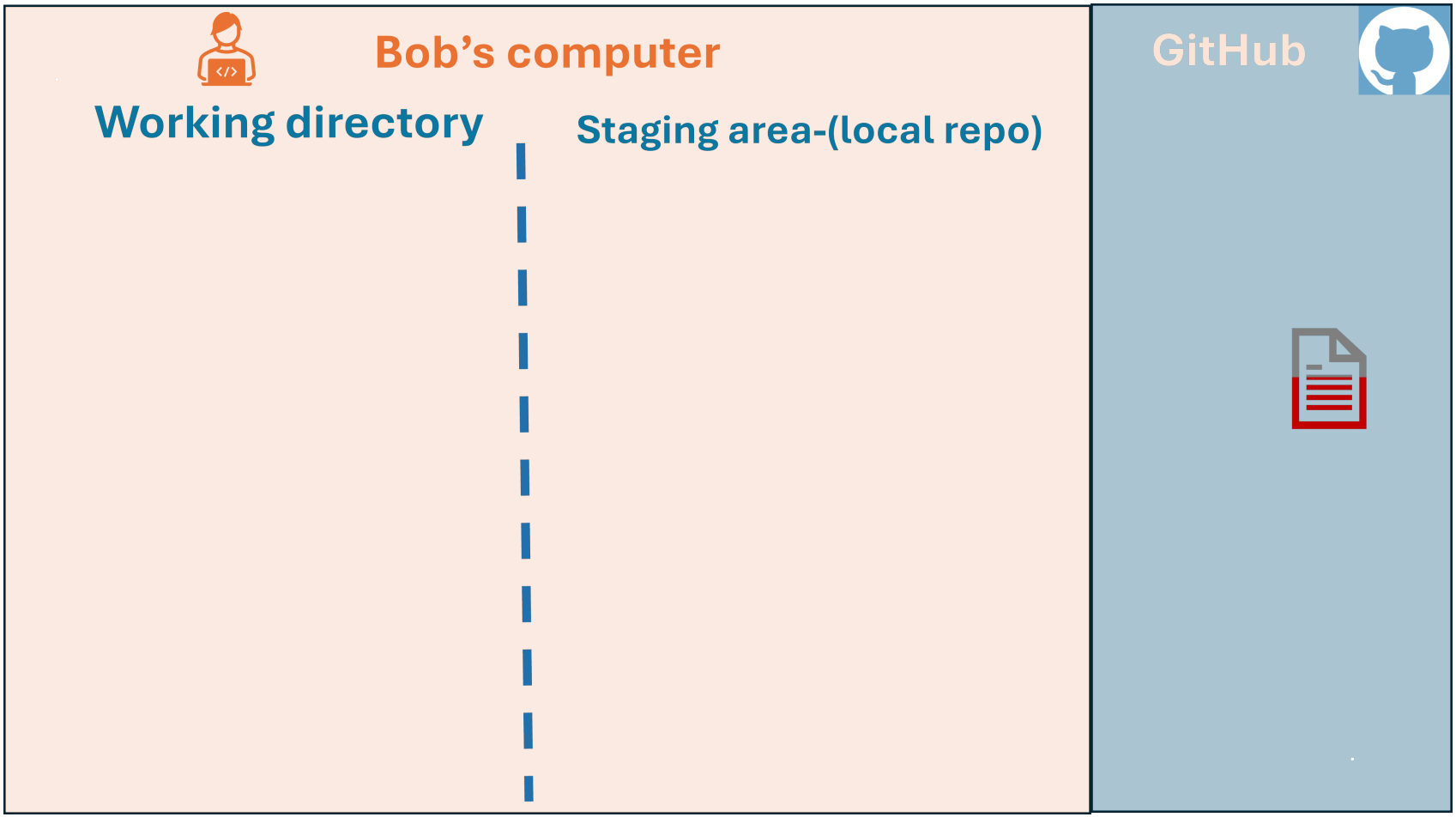
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



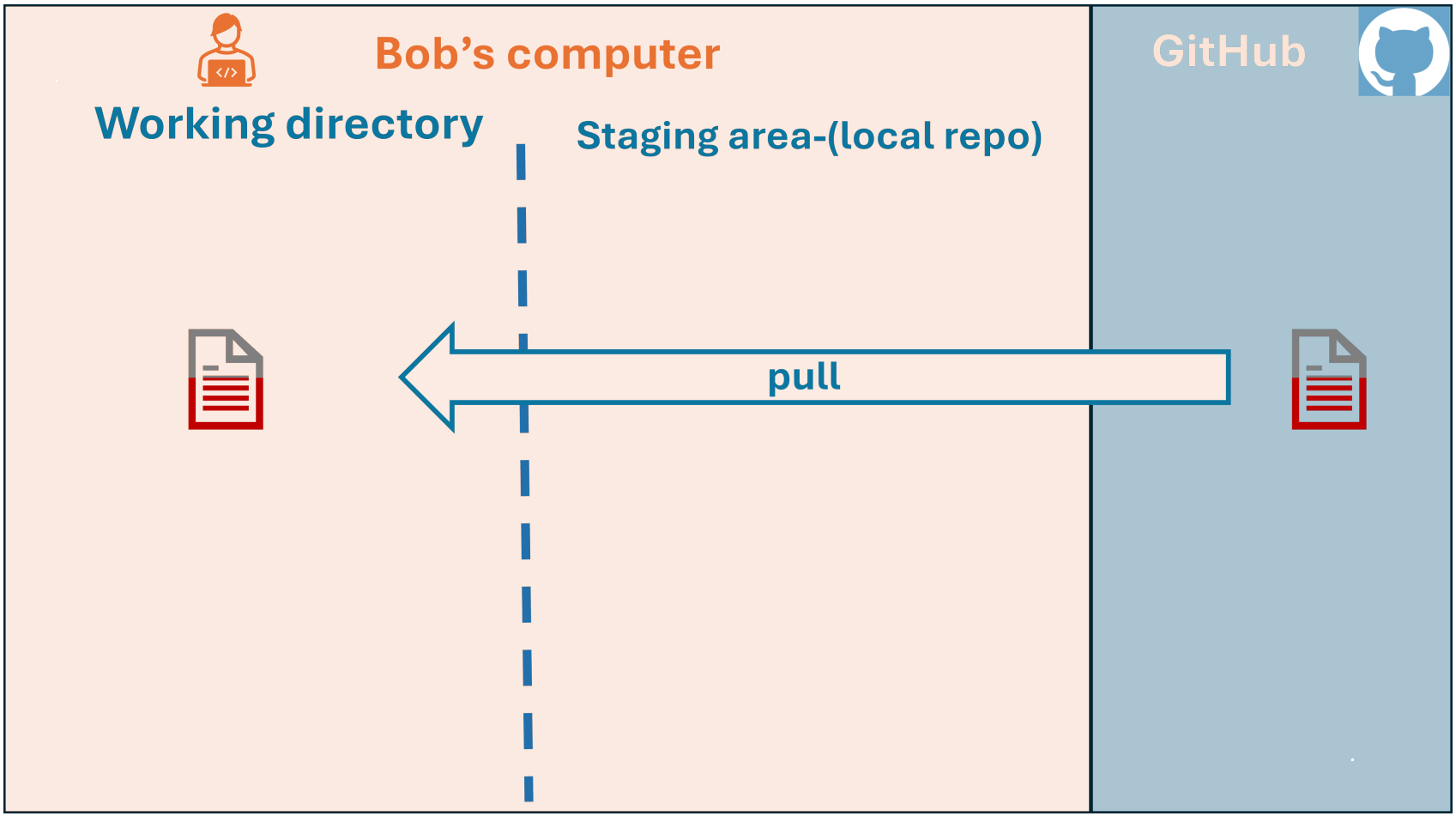
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



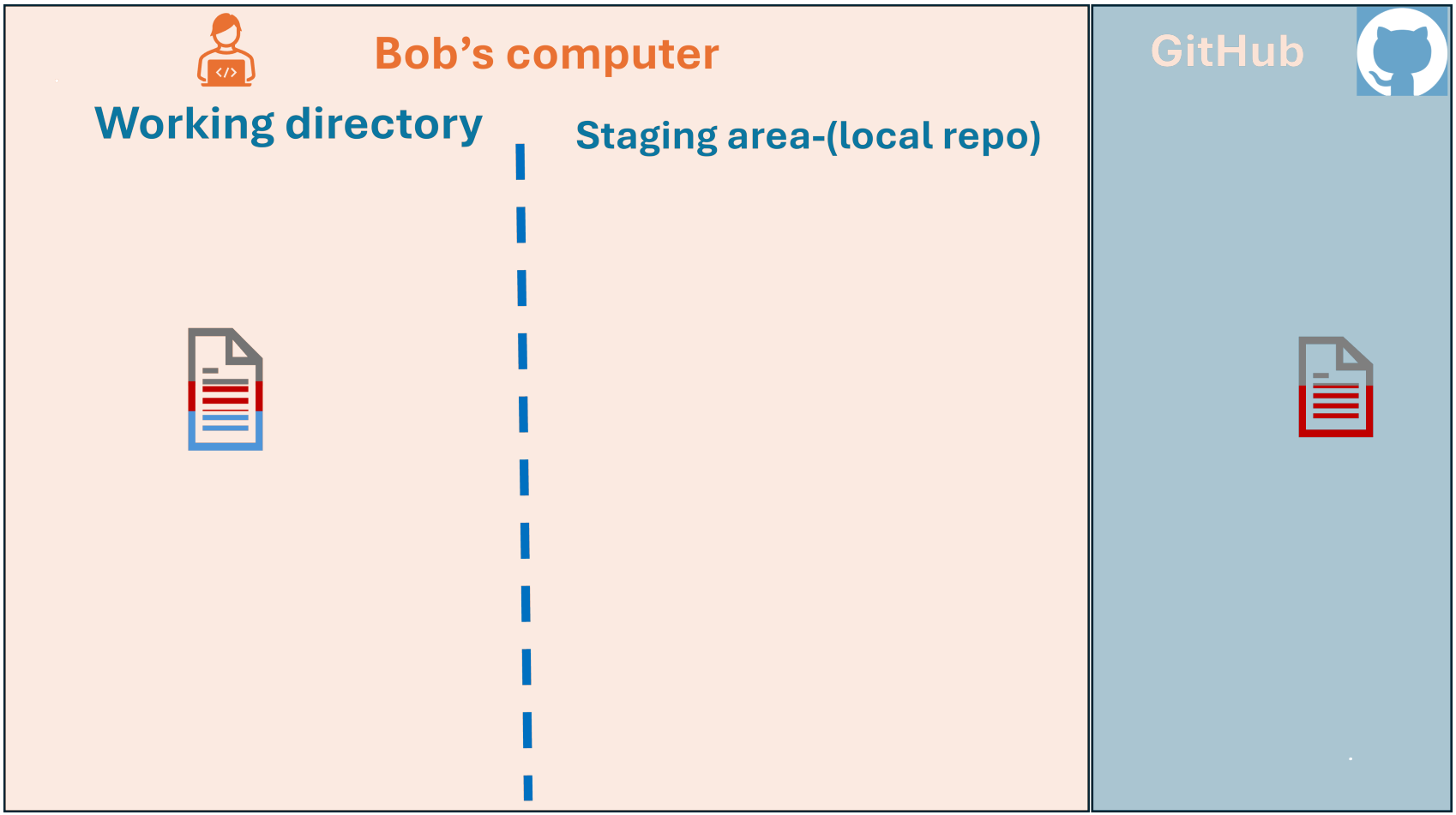
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



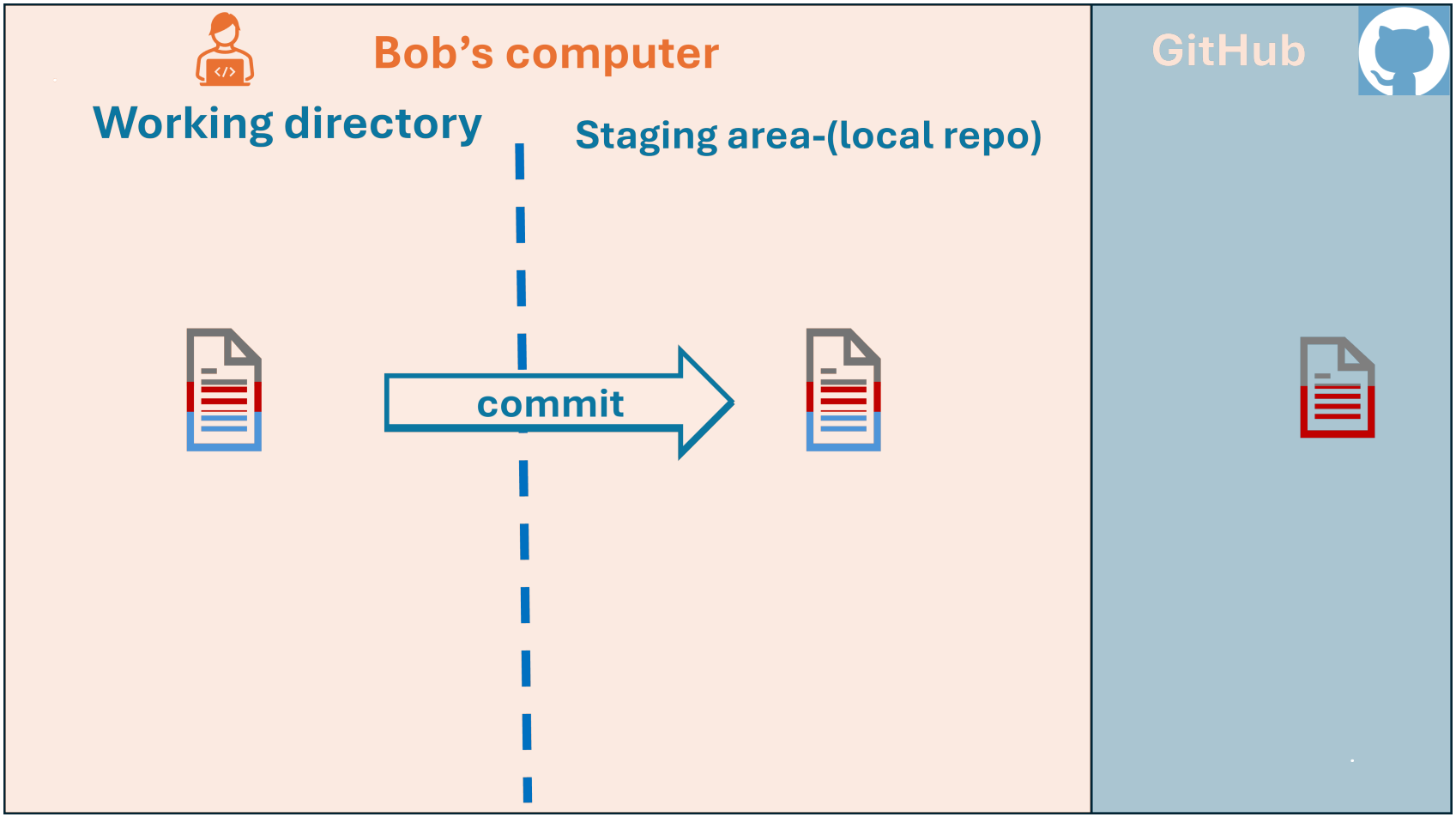
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



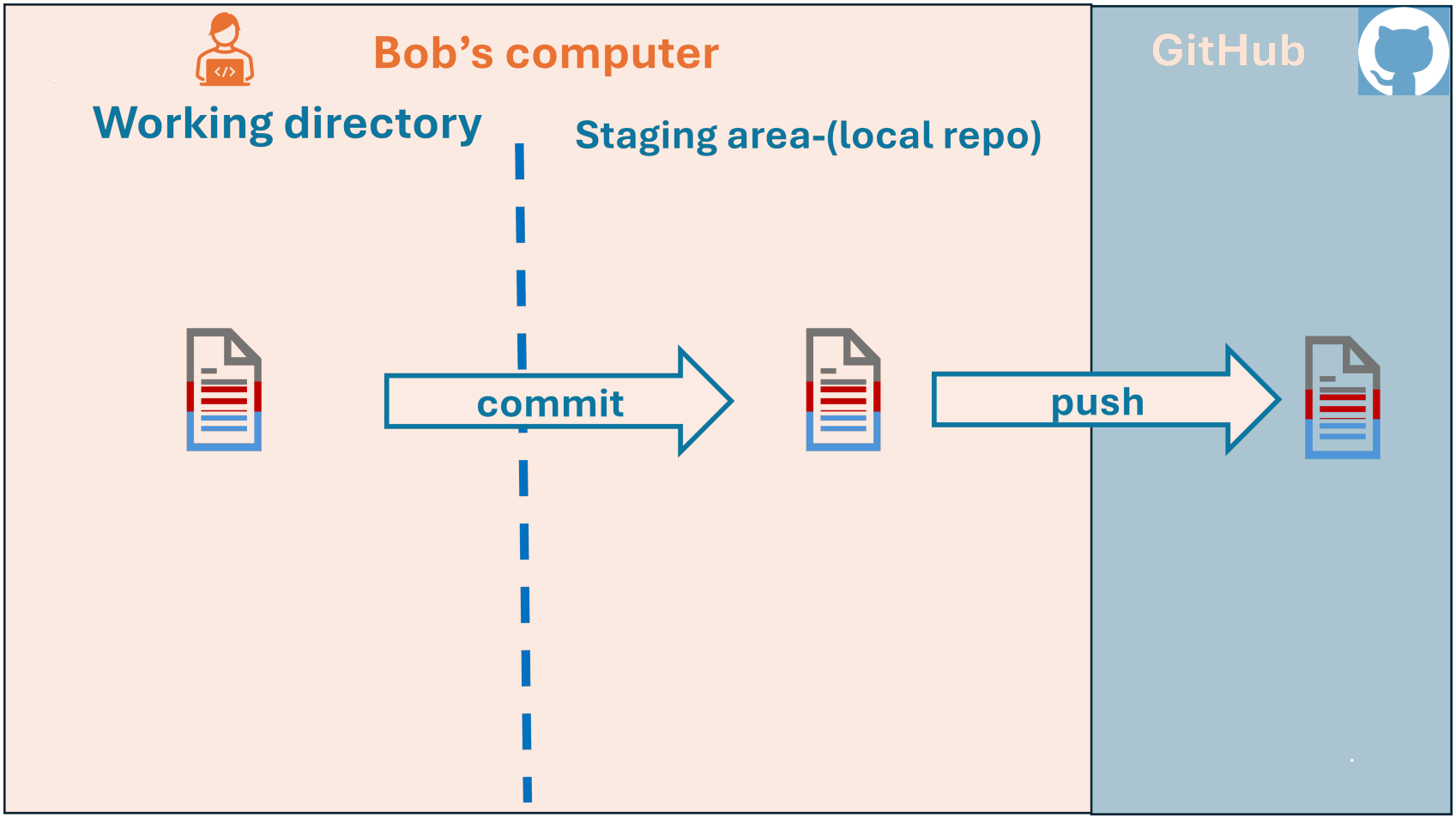
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



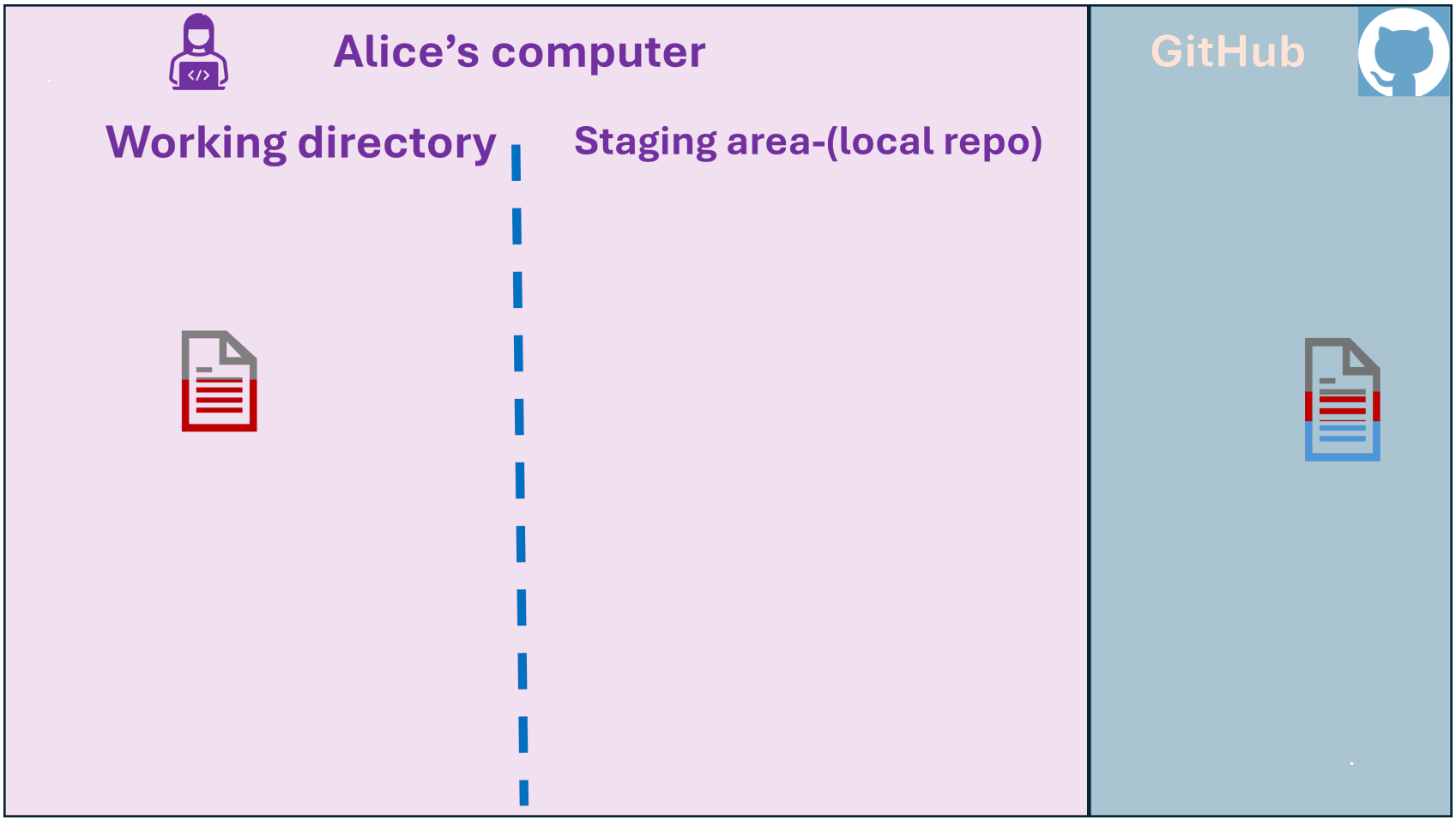
Git for collaborating: Details

Alice and **Bob** work on the same file (Details)



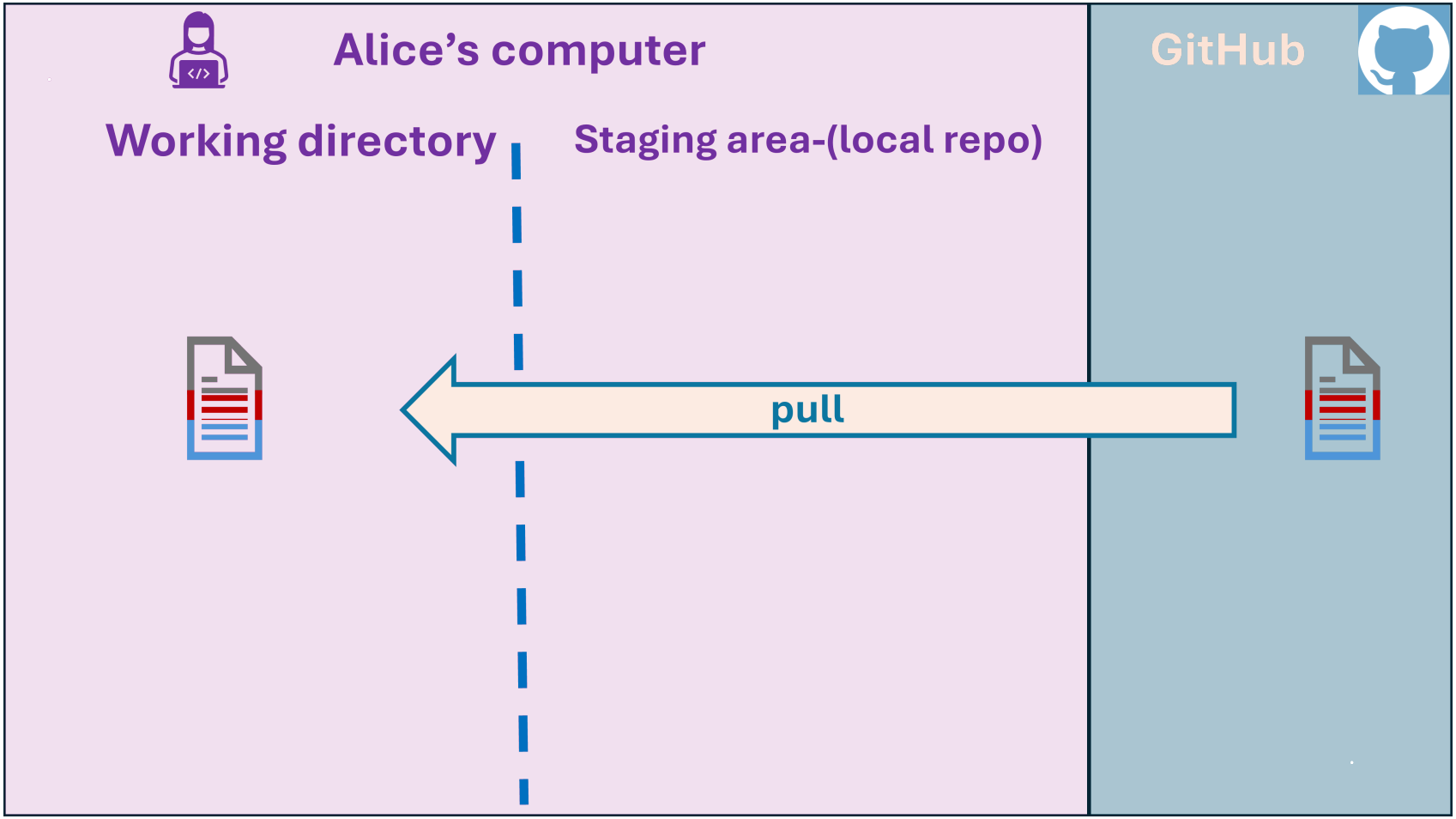
GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



GIT FOR COLLABORATING: DETAILS

Alice and **Bob** work on the same file (Details)



Remarks and issues

- A commit can include changes from multiple files simultaneously

Remarks and issues

- A commit can include changes from multiple files simultaneously
- One can do several commits before pushing (to GitHub)

Remarks and issues

- A commit can include changes from multiple files simultaneously
- One can do several commits before pushing (to GitHub)
- Commit messages can be edited (`amend`)

Remarks and issues

- A commit can include changes from multiple files simultaneously
- One can do several commits before pushing (to GitHub)
- Commit messages can be edited (`amend`)
- Every commit has an identifier (`hash` or `SHA`)

Remarks and issues

- A commit can include changes from multiple files simultaneously
- One can do several commits before pushing (to GitHub)
- Commit messages can be edited (`amend`)
- Every commit has an identifier (`hash` or `SHA`)
- Git manage complex situations

Remarks and issues

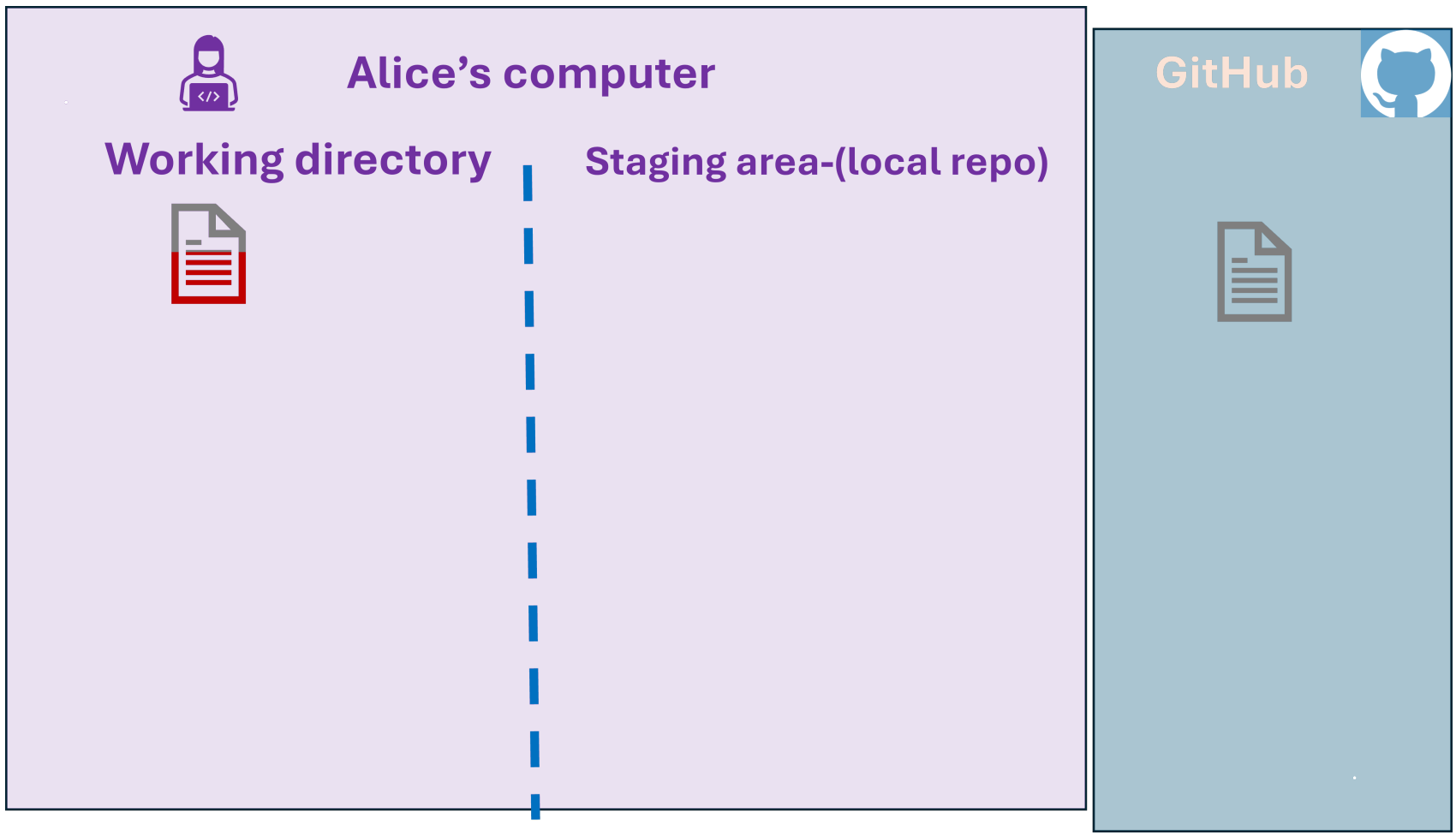
- A commit can include changes from multiple files simultaneously
- One can do several commits before pushing (to GitHub)
- Commit messages can be edited (`amend`)
- Every commit has an identifier (`hash` or `SHA`)
- Git manage complex situations
- Many actions available directly in Visual Studio

GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**

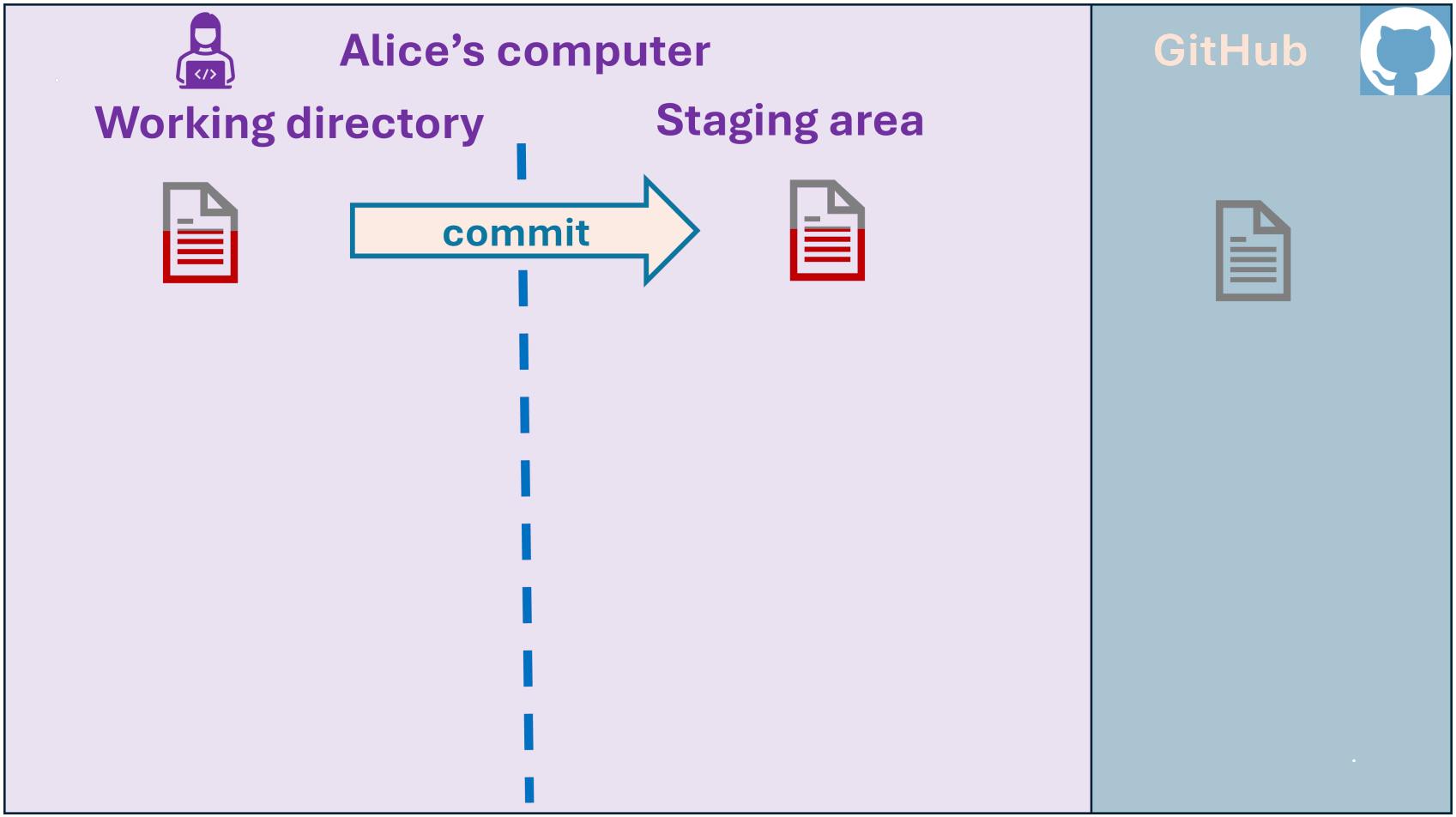
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



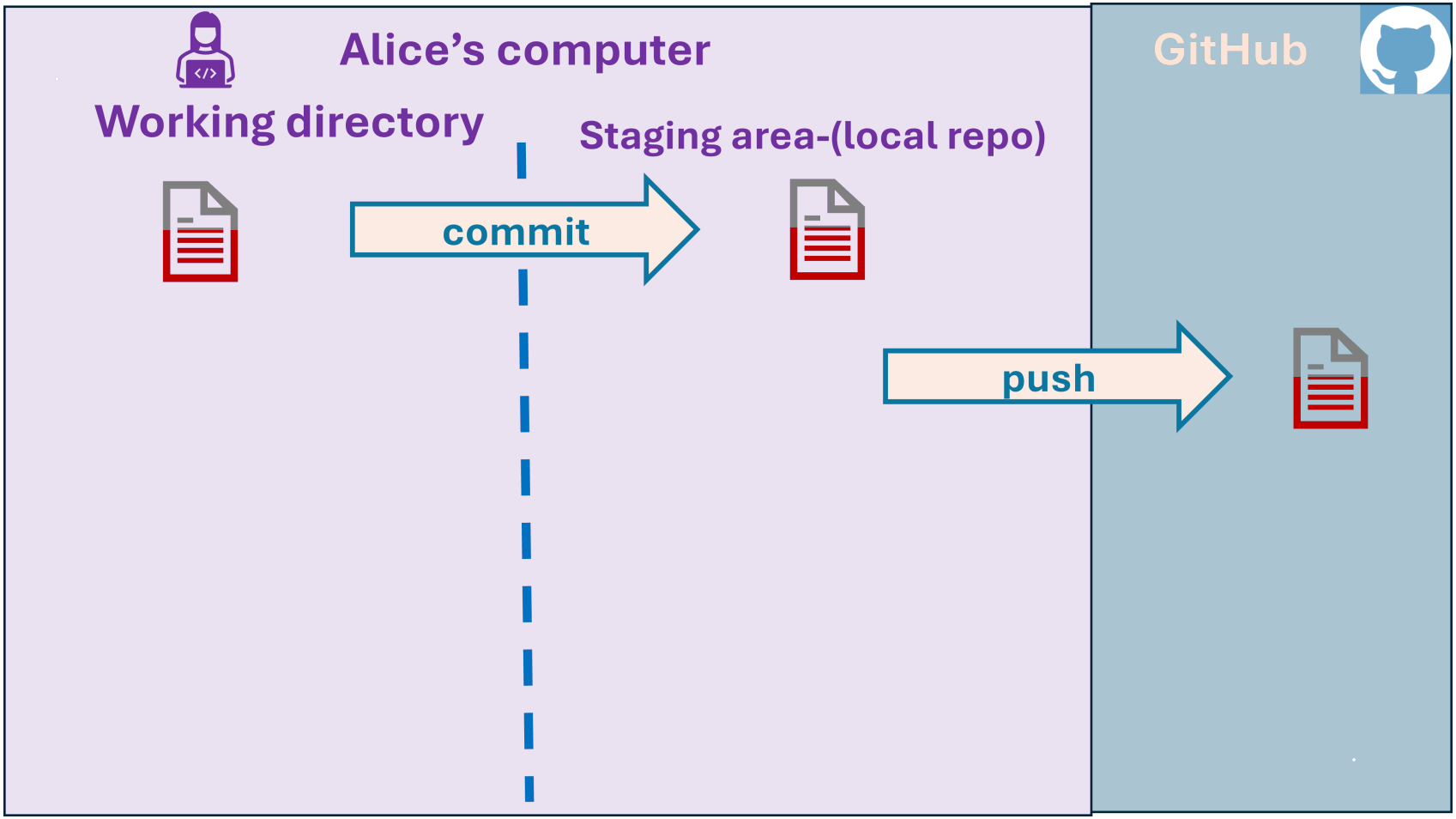
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



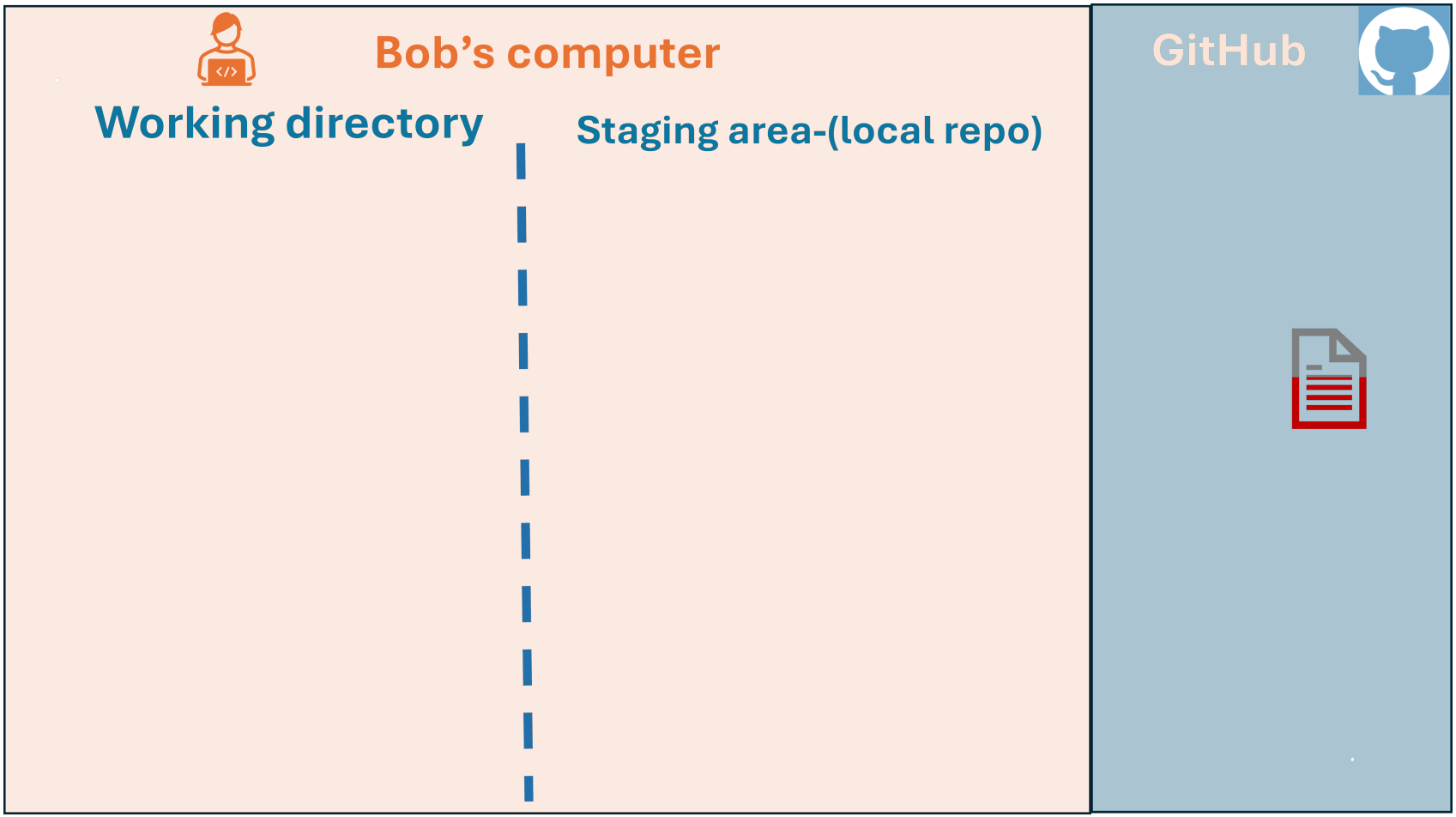
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



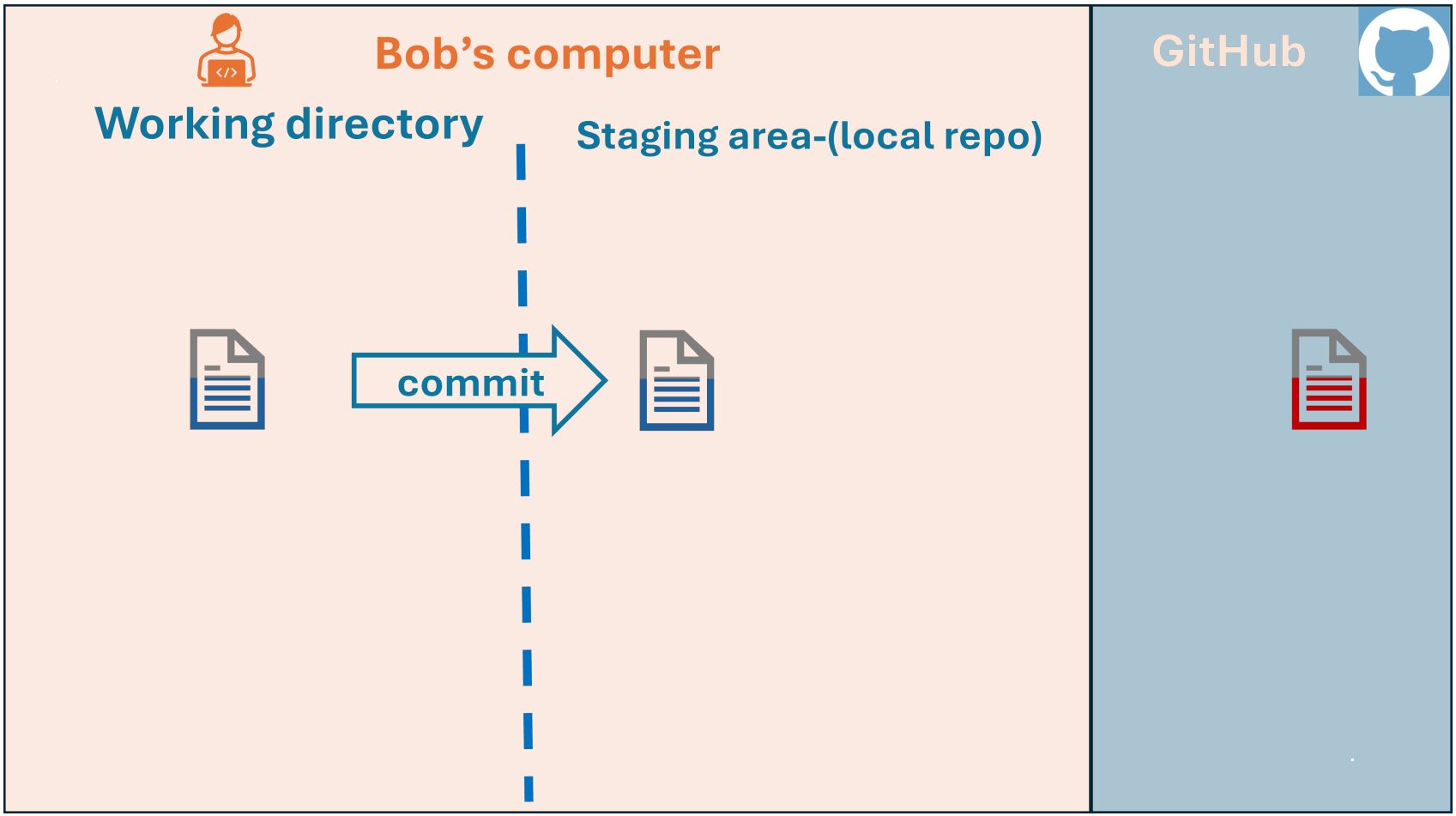
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



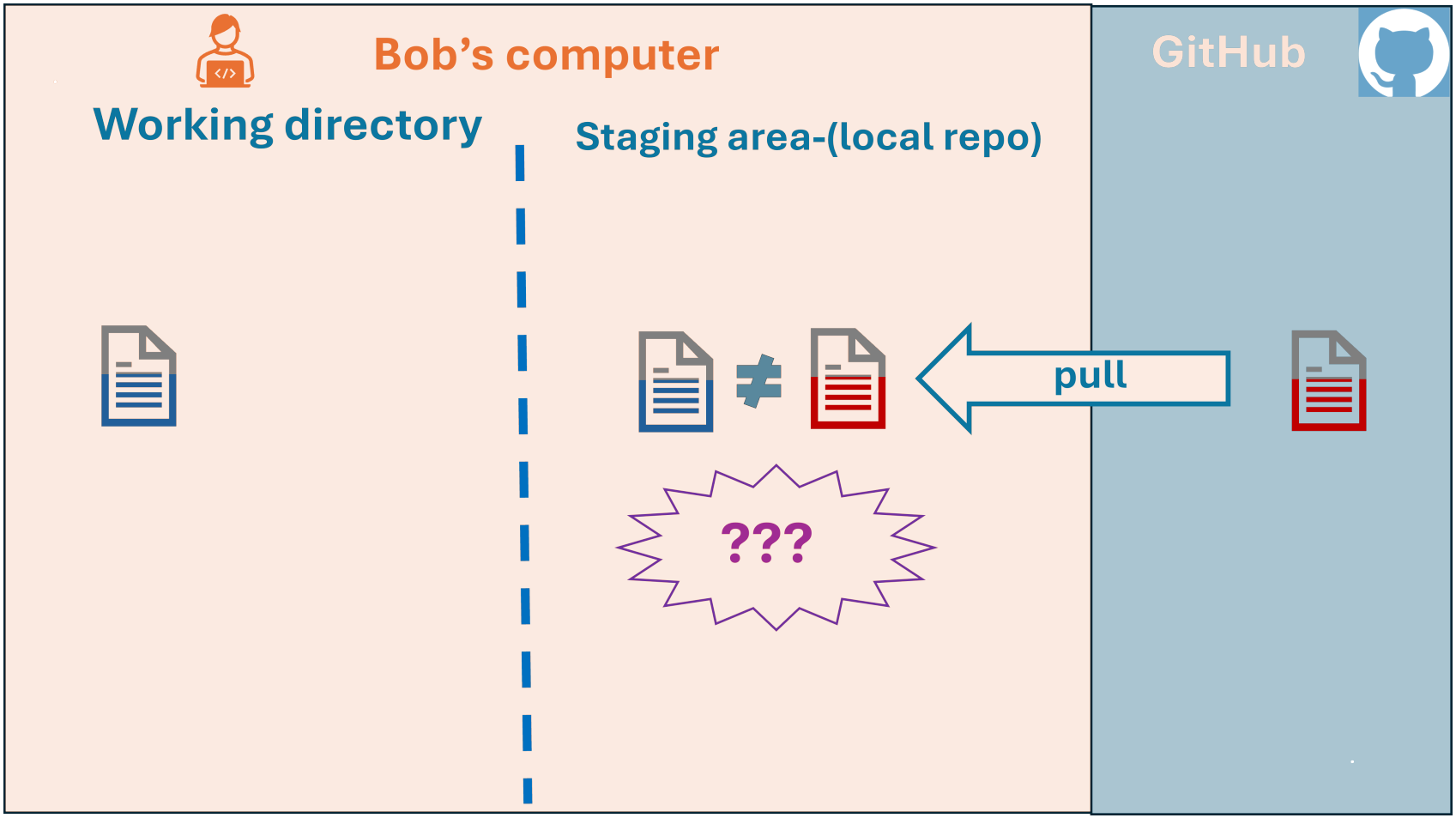
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



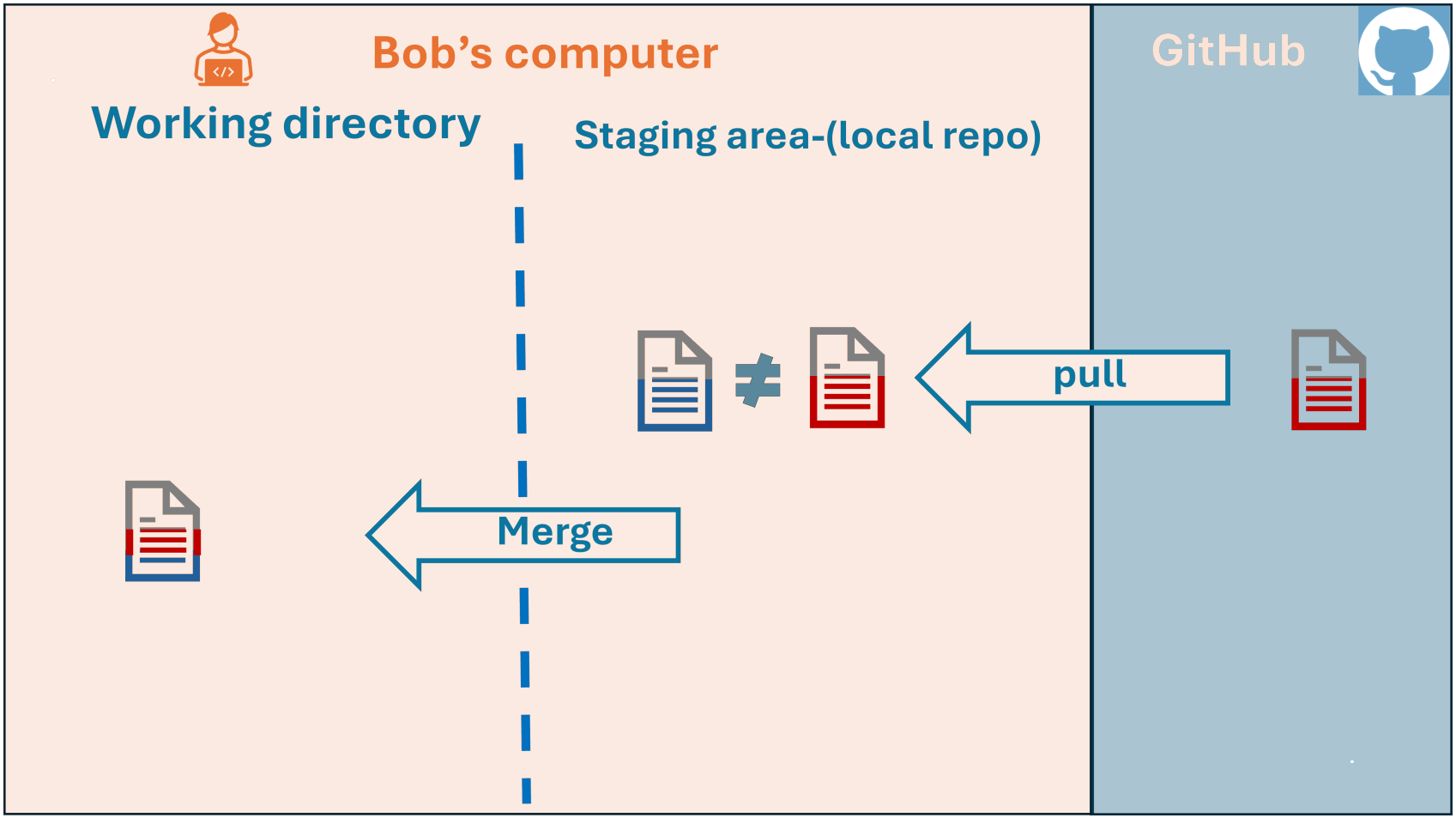
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



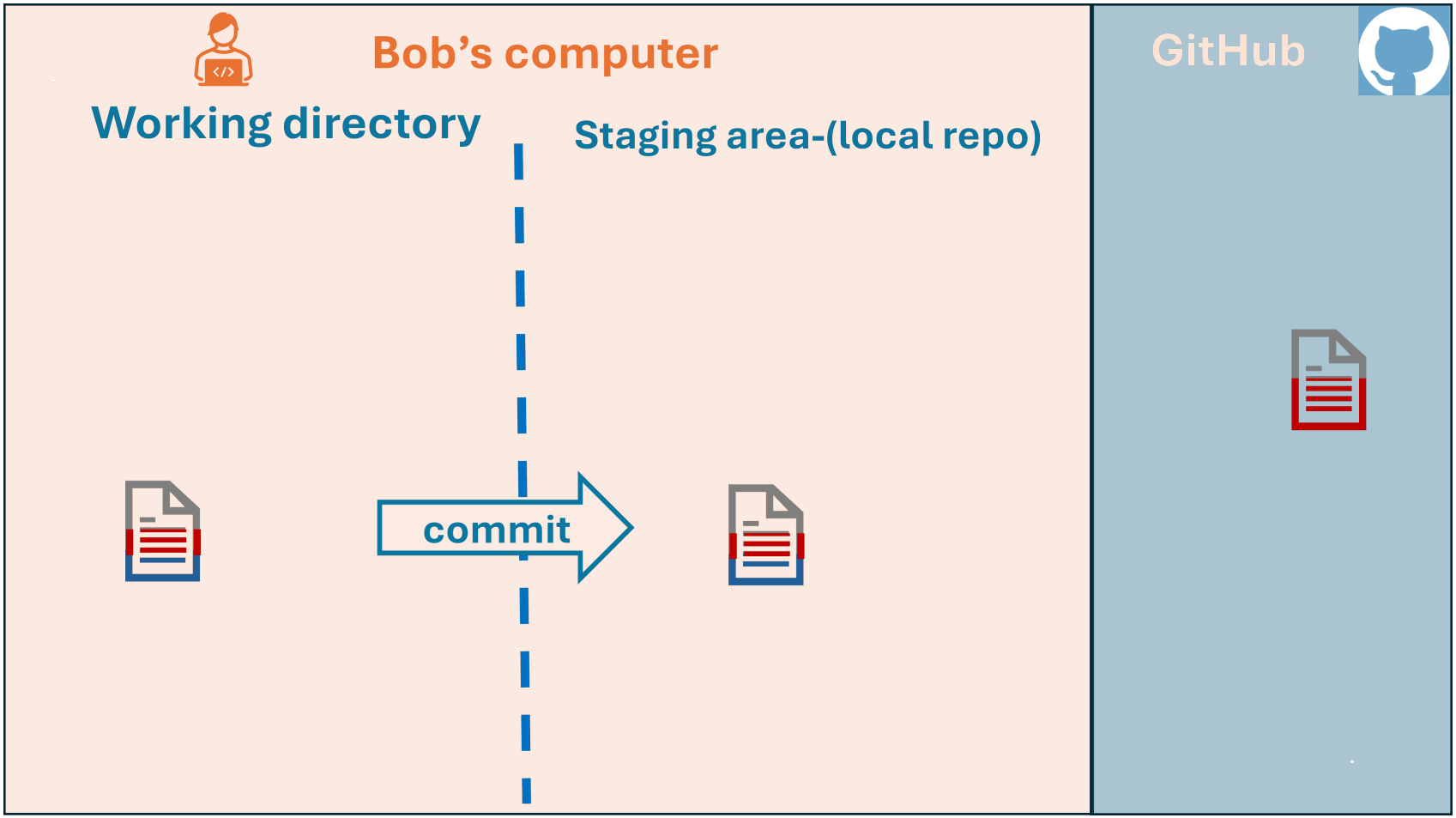
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



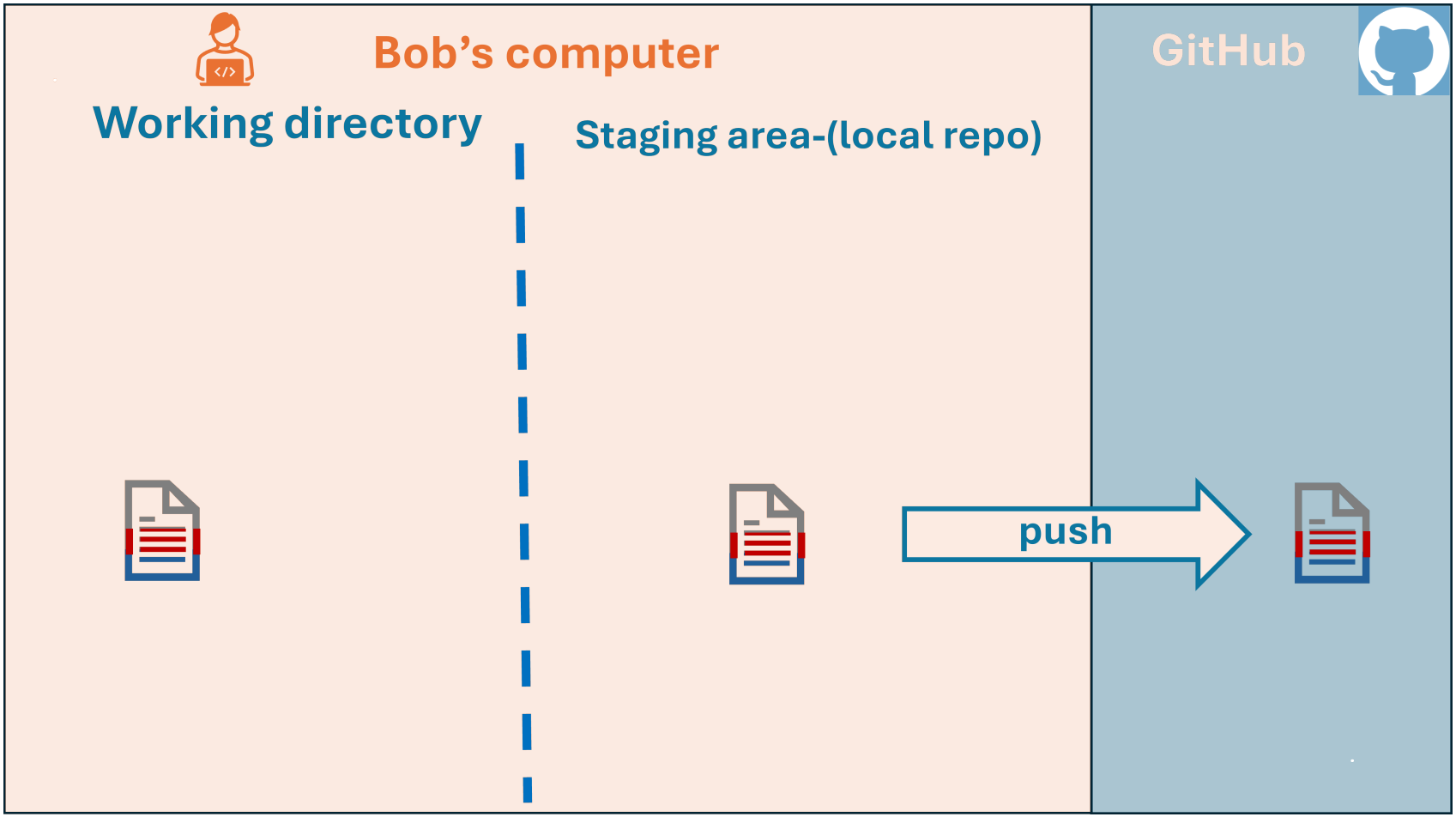
GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



GIT FOR COLLABORATING: COMPLEX SITUATIONS

Alice and **Bob** work on the same file **at the same time**



Version Control in a Nutshell

Version control system:

- Allows to travel back in time

GitHub logo

Version Control in a Nutshell

Version control system:

- Allows to travel back in time
- Keeps track of all changes

GitHub logo

Version Control in a Nutshell

Version control system:

- Allows to travel back in time
- Keeps track of all changes
- Allows to "undo" at any point

GitHub logo

Version Control in a Nutshell

Version control system:

- Allows to travel back in time
- Keeps track of all changes
- Allows to "undo" at any point
- Allows reviewing stages of development

GitHub logo

Version Control in a Nutshell

Version control system:

- Allows to travel back in time
- Keeps track of all changes
- Allows to "undo" at any point
- Allows reviewing stages of development
- Allow collaborating on projects

GitHub logo

Version Control in a Nutshell

Version control system:

- Allows to travel back in time
- Keeps track of all changes
- Allows to "undo" at any point
- Allows reviewing stages of development
- Allow collaborating on projects
- Backups your work

GitHub logo

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training
- New tools, new challenges, new problems...

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training
- New tools, new challenges, new problems...

↪ Requires time, patience and training

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training
- New tools, new challenges, new problems...

↪ Requires time, patience and training

- Git is powerful, but complex

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training
- New tools, new challenges, new problems...

↪ Requires time, patience and training

- Git is powerful, but complex

↪ It works with 

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training
- New tools, new challenges, new problems...

↪ Requires time, patience and training

- Git is powerful, but complex

↪ It works with 

- Automation is hard

Towards a full RAP

- In short,

RAP

=

Reproducible documents

+

Version Control

+

Automation

- Each of these notion require training
- New tools, new challenges, new problems...

↪ Requires time, patience and training

- Git is powerful, but complex

↪ It works with 

- Automation is hard
- Building a RAP is a collective process

Useful resources

- This course website (created by Serge Goussev)
- Vanuatu Bureau of Statistics implementation of RAP
- SIAP's (free) online RAP course
- The UK government RAP website.
- UK best practice documentation.
- A free RAP course to teach you all you need to know.
- How the Data Science Campus sets its coding standards.
- A new open-source book from the Alan Turing institute setting out how to do reproducible data science.