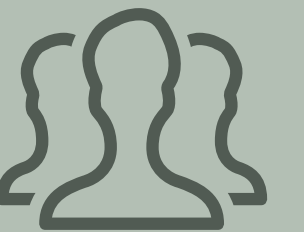
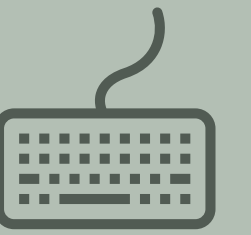
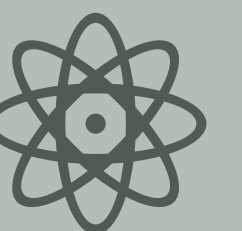


EXPERIMENTAL
PSYCHOLOGY
LAB
PRACTICE



TIDY COOPERATION



TODAY'S TOPICS

1 folder structure

2 version control

3 git

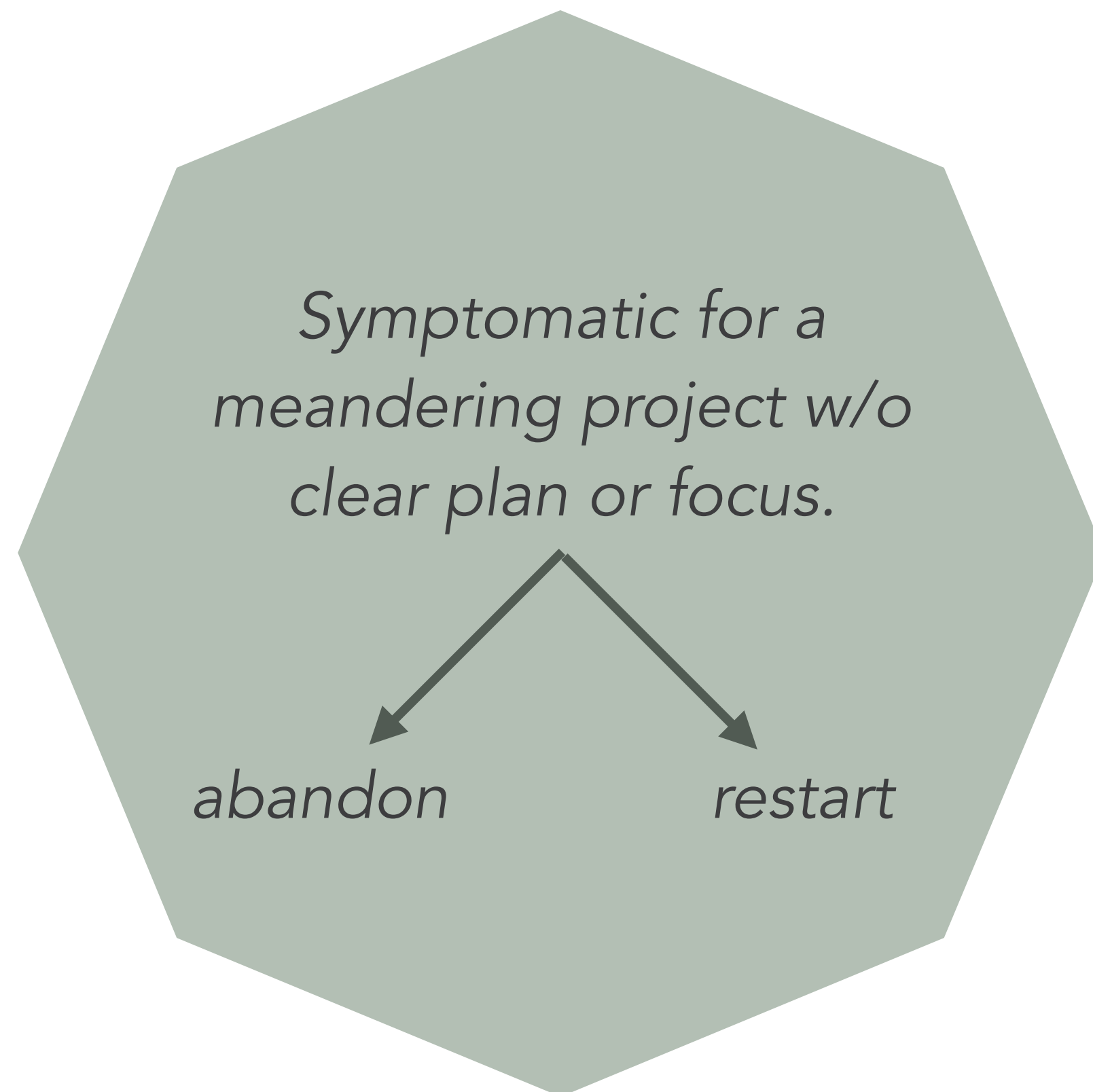
4 markdown



FOLDER STRUCTURE ::: HOW NOT TO

manually produced version history → clutter

no conceptual structure → big brown soup of paper, experiments, posters, notes, code snippets etc.



Name	Date Modified
manuscript	24. Mar 20
revision_2018	3. Apr 201
paper_MF	19. Sep 20
aktuelle_version	12. Aug 20
manuscript_11_08.docx	12. Aug 20
manuscript_10_08_2.docx	10. Aug 20
manuscript_10_08.pdf	10. Aug 20
manuscript_10_08.docx	10. Aug 20
manuscript_07_08.docx	9. Aug 201
manuscript_04_08_2.docx	4. Aug 201
manuscript_04_08.docx	4. Aug 201
manuscript_03_08.docx	3. Aug 201
manuscript_02_08.docx	2. Aug 201
manuscript_29_07.pdf	30. Jul 20
manuscript_29_07.docx	30. Jul 20
manuscript_07_03.pdf	7. Mar 201
manuscript_07_03.docx	7. Mar 201
squid_barplots	25. Feb 20
modeling_N400	14. Nov 20
amlap_2017	10. Aug 20
schedules	4. Aug 201
amlap_2016	24. May 20
abstract_Bochum	12. May 20
CoCoLab_slides	16. Apr 20
behav_all_squid3.txt	3. Apr 201
behav_all_squid4.txt	3. Apr 201
squid_3_data	23. Mar 20
SFB-Poster	17. Mar 20
fragebogenstudie	7. Mar 201
pictures_small_PDFs	1. Mar 201
pictures	28. Feb 20
data_fit_P	14. Feb 20

FOLDER STRUCTURE ::: MINIMAL TEMPLATE

usually you want to include:

analyses

[scripts to process, visualize and analyze data]

data

[raw and preprocessed data from all experiments]

experiments

[everything relevant to understanding & reproducing the experiments]

notes

[internal bookkeeping, notes from meetings, thoughts etc.]

writing

[for the final write-up, term paper, report, publication etc.]

```
~/Desktop/LabPrac/example_project_repo_minimal $ tree
```

```
.
|-- analyses
|   |-- 01_pilot
|   |   |-- 01_data_preprocessing.R
|   |   |-- 02_data_plotting.R
|   |   `-- 03_summary.Rmd
|   `-- 02_main
|       |-- 01_data_preprocessing.R
|       |-- 02_data_plotting.R
|       |-- 03_regression_models.R
|       |-- 04_visualize_models.R
|       `-- 05_summary.Rmd
|-- data
|   |-- 01_pilot
|   |   |-- 01_raw_data.csv
|   |   `-- 02_clean_data.csv
|   `-- 02_main
|       |-- 01_raw_data.csv
|       |-- 02_clean_data.csv
|       `-- 03_aggregate_data.csv
|-- experiments
|   |-- 01_pilot
|   |   |-- everything_about_the_pilot
|   |   `-- think::pictures_code_participantInfo
|   `-- 02_main
|       `-- everything_about_the_main_experiment
|-- notes
|   `-- 01_first_meeting_1978-12-06.md
`-- writing
    |-- 01_termPaper
    |   `-- termPaper.Rmd
    `-- 02_journalPaper
        `-- journalPaper.Rmd
```

FOLDER STRUCTURE ::: EXTENSIVE TEMPLATE

maybe also include:

code

[for simulations, cognitive models beyond statistical analyses]

posters

presentations

whateverelsemakesense

README .md

[in every relevant folder to describe what is in which files]

```
[~/Desktop/LabPrac/example_project_repository $ tree
```

```
.
|-- analyses
|   |-- 01_pilot
|   |   |-- 01_data_preprocessing.R
|   |   |-- 02_data_plotting.R
|   |   `-- 03_summary.Rmd
|   `-- 02_main
|       |-- 01_data_preprocessing.R
|       |-- 02_data_plotting.R
|       |-- 03_regression_models.R
|       |-- 04_visualize_models.R
|       `-- 05_summary.Rmd
|-- code
|   `-- 01_cognitive_model.R
|-- data
|   |-- 01_pilot
|   |   |-- 01_raw_data.csv
|   |   `-- 02_clean_data.csv
|   `-- 02_main
|       |-- 01_raw_data.csv
|       |-- 02_clean_data.csv
|       `-- 03_aggregate_data.csv
|-- experiments
|   |-- 01_pilot
|   |   |-- everything_about_the_pilot
|   |   `-- think::pictures_code_participantInfo
|   `-- 02_main
|       `-- everything_about_the_main_experiment
|-- notes
|   `-- 01_first_meeting_1978-12-06.md
|-- posters
|   `-- 01_CogSci_2016.key
|-- presentations
|   `-- 01_EuroCogSci_2017.key
`-- writing
    |-- 01_termPaper
    |   `-- termPaper.Rmd
    `-- 02_journalPaper
        `-- journalPaper.Rmd
```

VERSION CONTROL ::: WHY

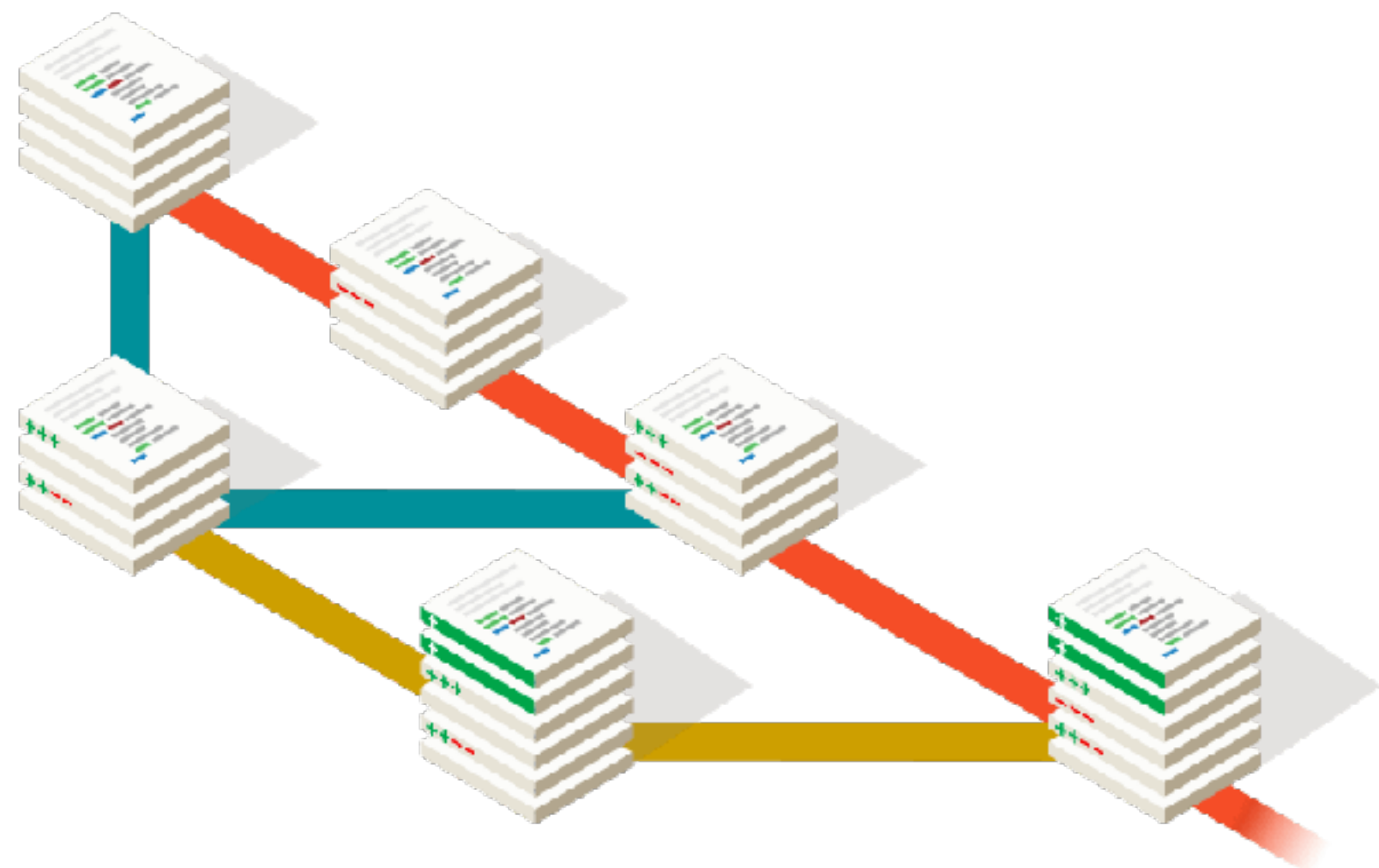
keep a record of every past state of your work

compare versions and track changes easily

back up all files in the process

try out ideas without cluttering or damaging working code

collaborate using issue tracking and automatic merges



THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

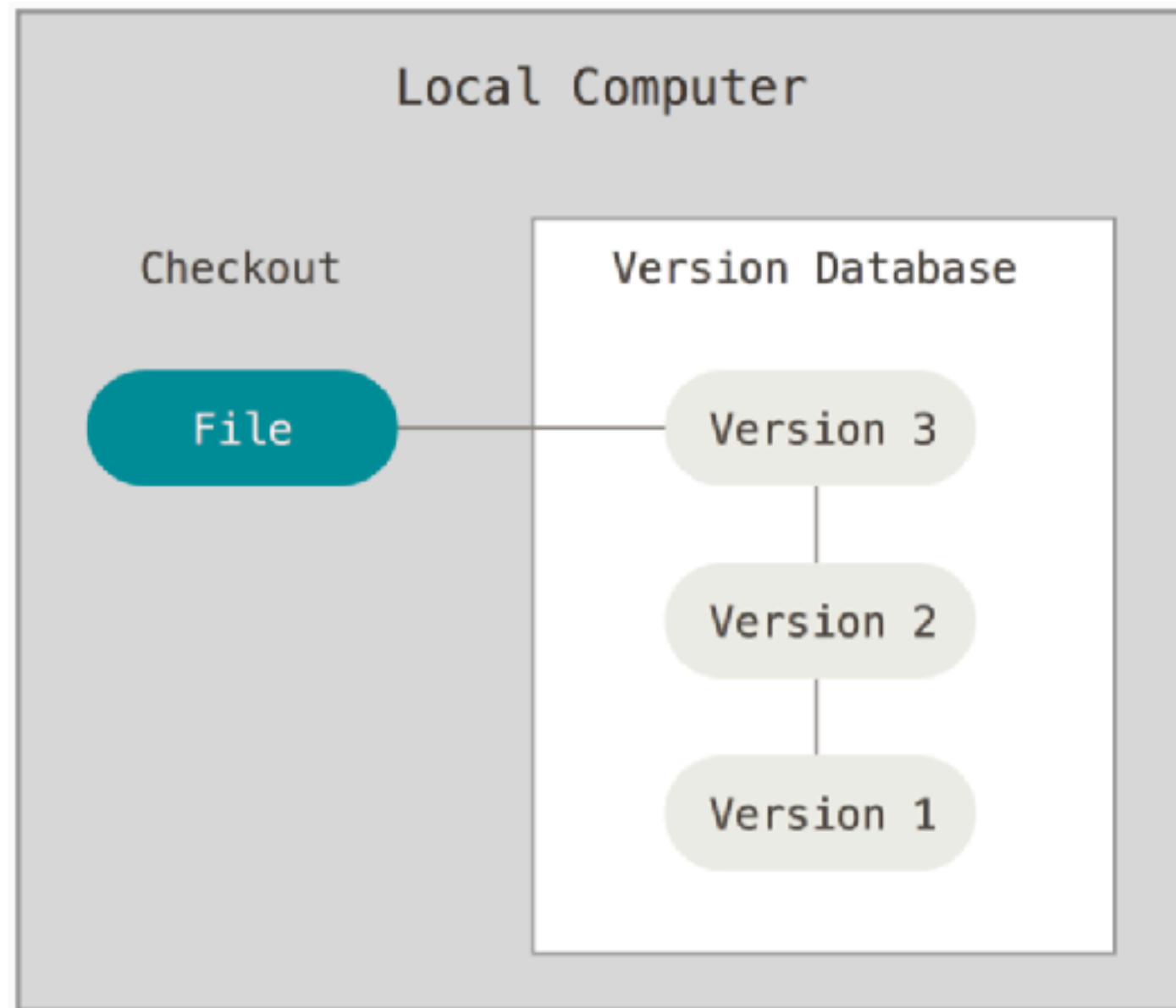
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

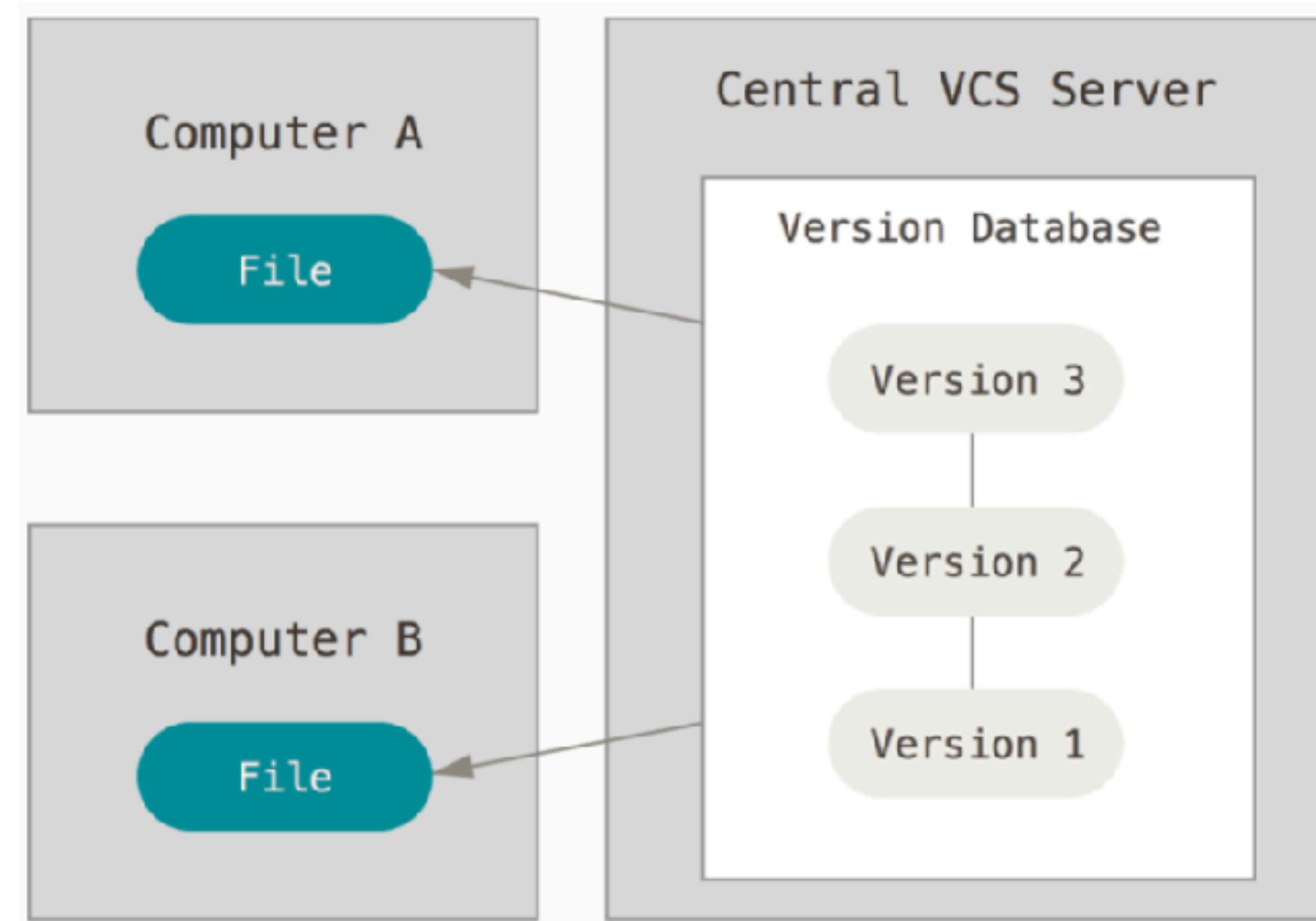


DIFFERENT TYPES OF VERSION CONTROL SYSTEMS

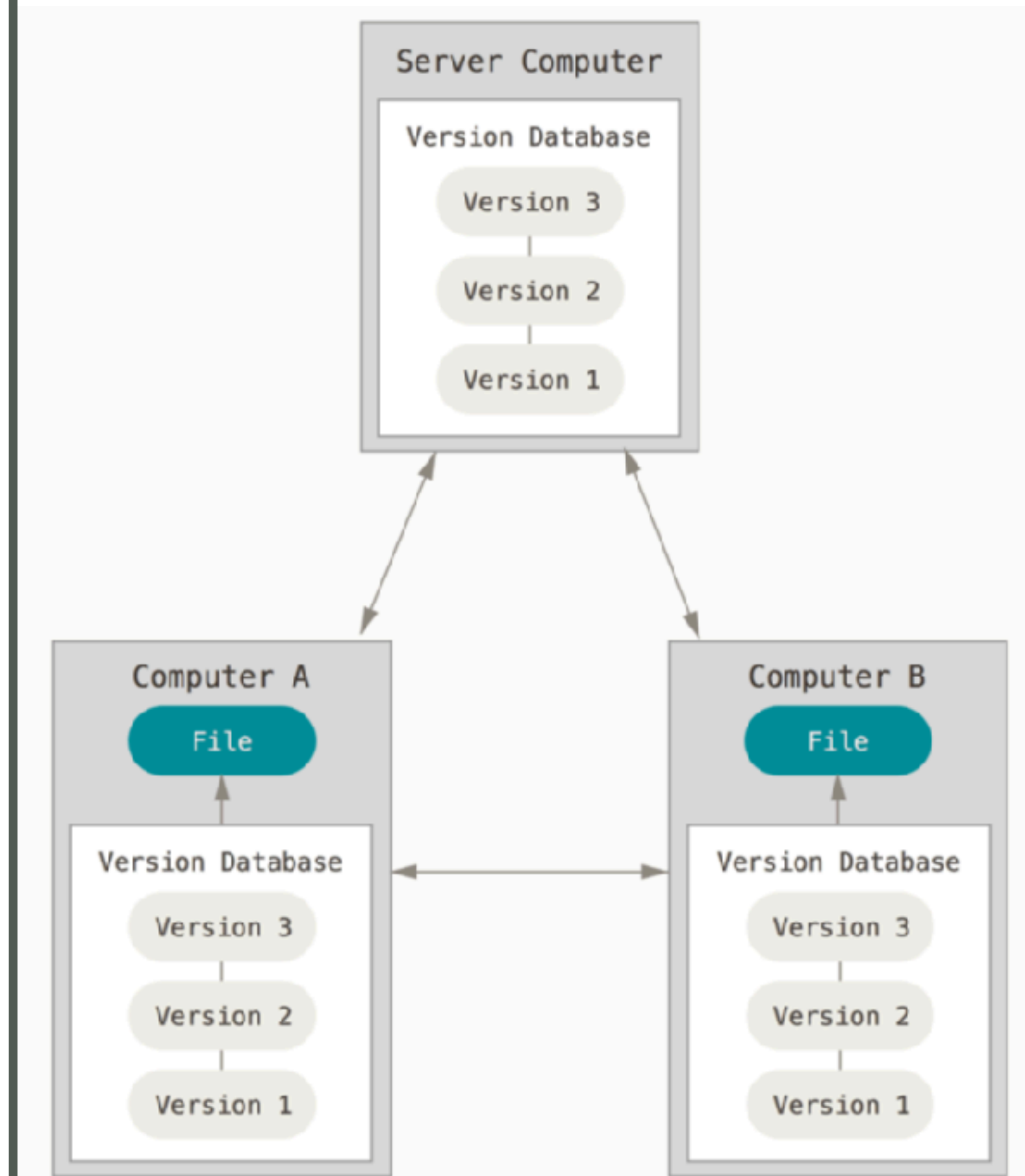
LOCAL



CENTRAL

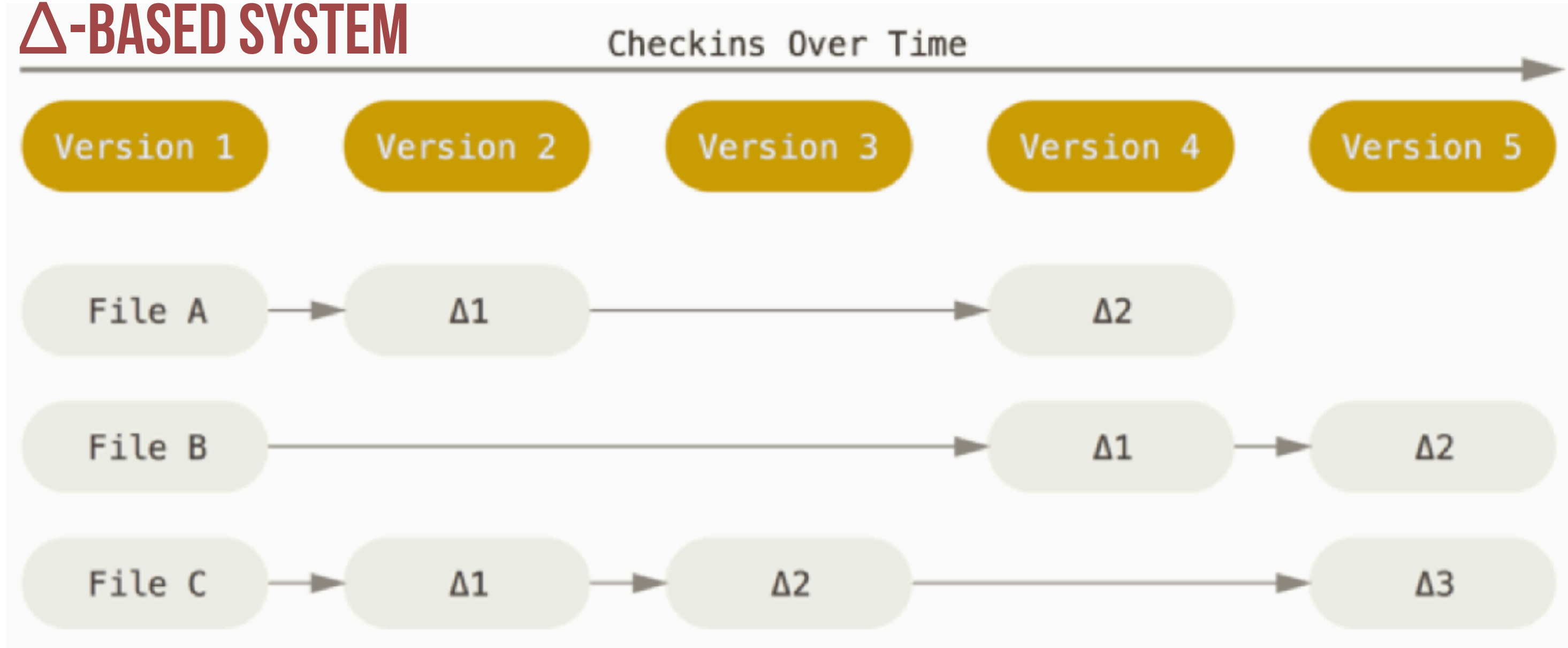


DISTRIBUTED

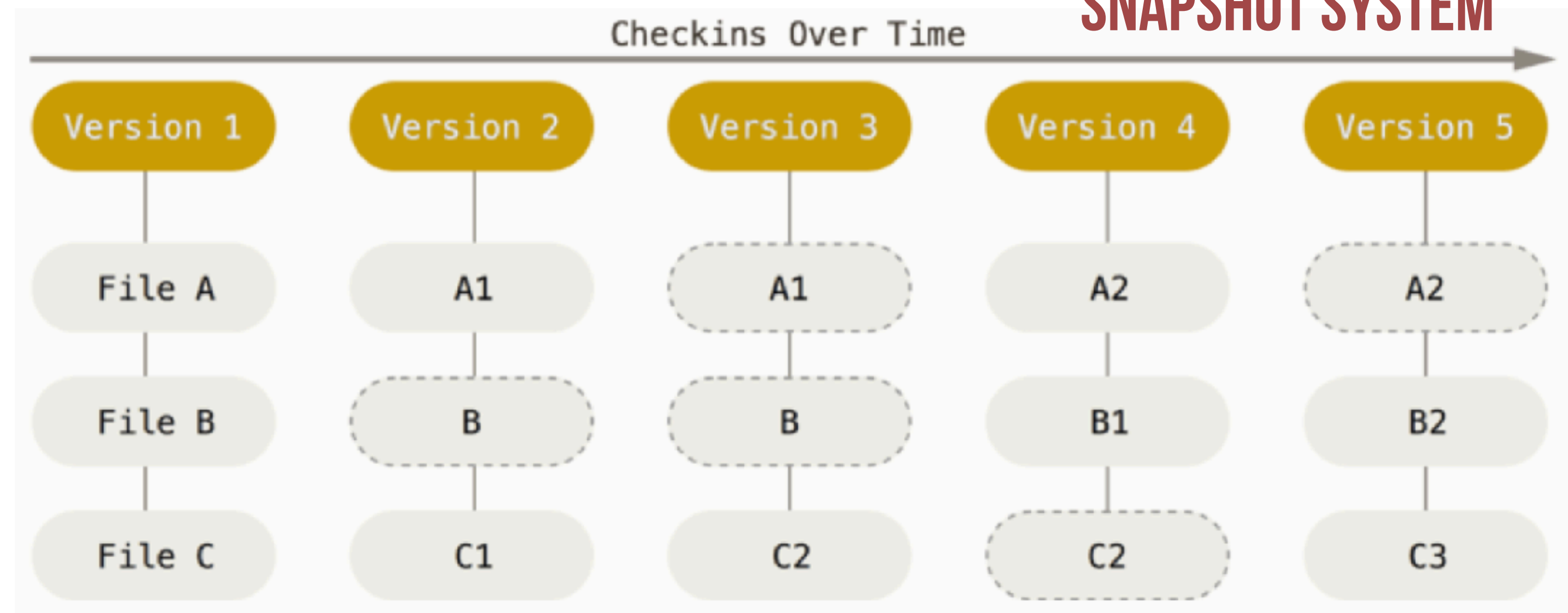


DIFFERENT TYPES OF VERSION CONTROL SYSTEMS

Δ-BASED SYSTEM



SNAPSHOT SYSTEM



BASIC CONCEPTS & TERMINOLOGY

pulling: download changes from server

tracked files: marked to be under VC

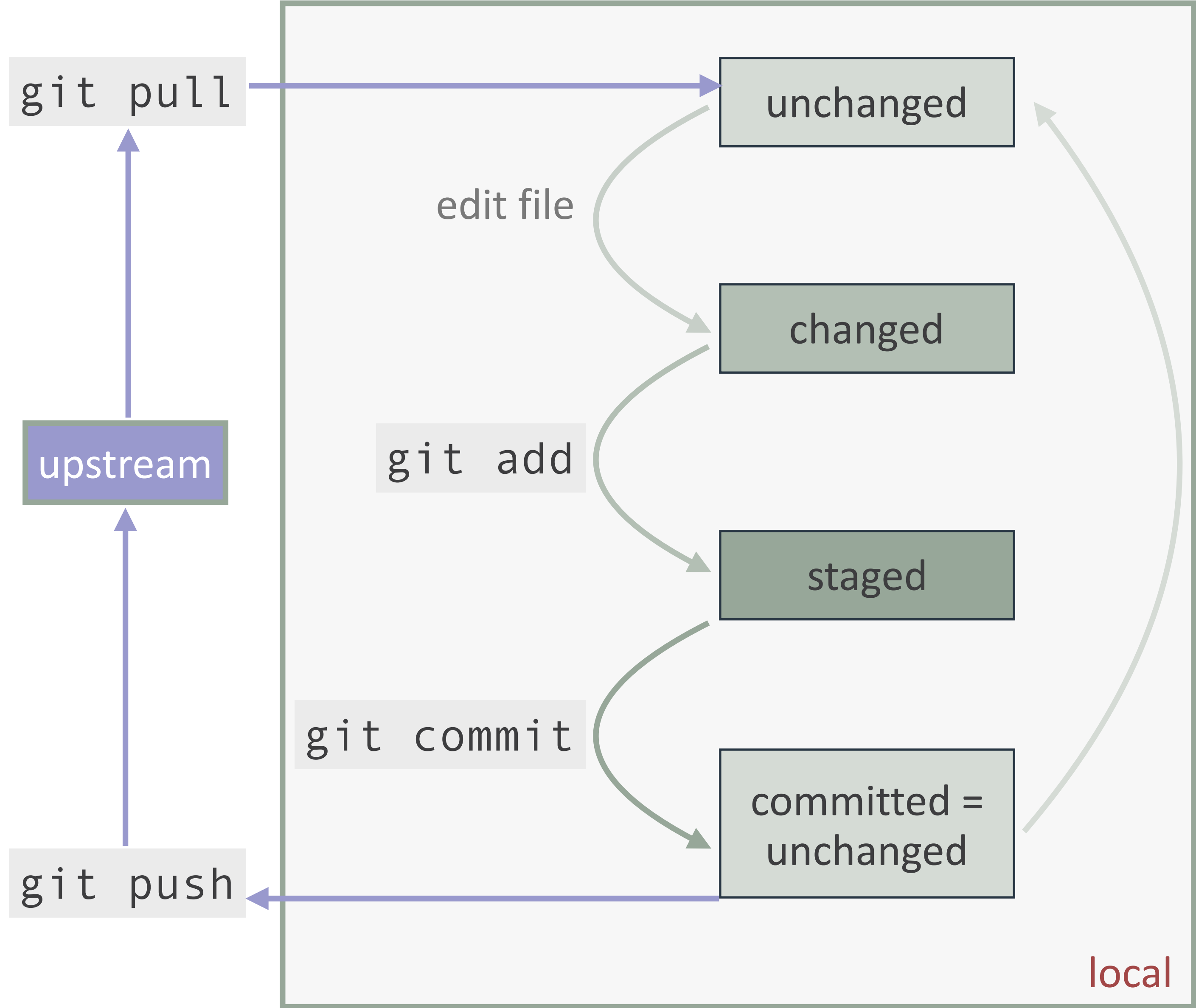
ignores: file types excluded from VC

adding/staging: mark local changes as to be committed

commit: save local changes locally

stage area (aka index): everything that is locally committed but not yet pushed

pushing: upload local changes to server



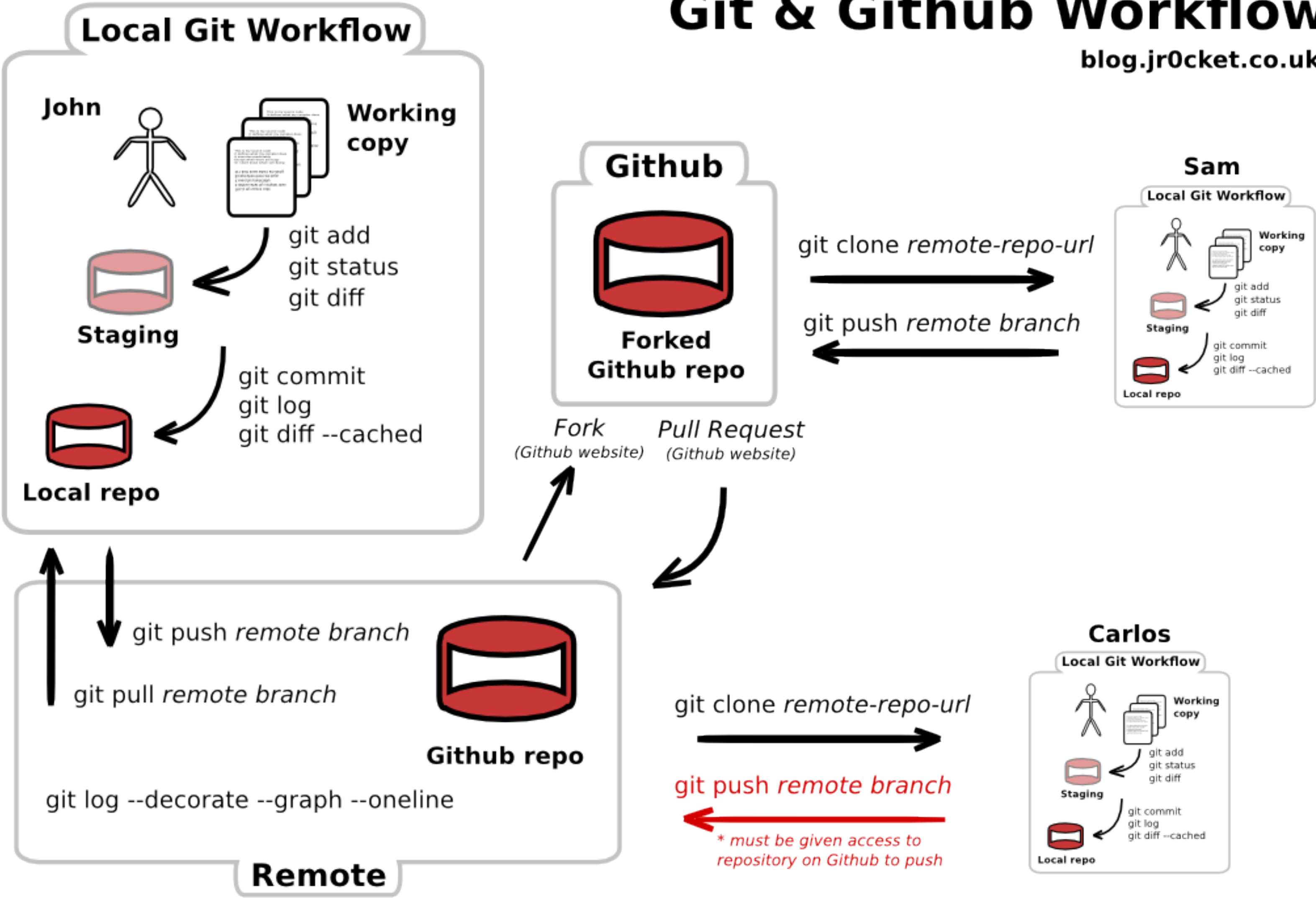
COOPERATION WITH GIT

Git & Github Workflow

blog.jr0cket.co.uk

use **log** & **diff** to know what others did

forks are unofficial copies, can be **merged** after a **pull request**



EVERYTHING'S NOT LOST

unless forced violently, git does not forget any information that it was given

you do not lose information

never upload sensitive information to a public repository

BASIC WORKFLOW

CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```

Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```

Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```

Enables helpful colorization of command line output

BASIC WORKFLOW

SUPPRESS TRACKING

Exclude temporary files and paths

```
*.log  
build/  
temp-*
```

A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

```
$ git ls-files --other --ignored --exclude-standard
```

Lists all ignored files in this project

BASIC WORKFLOW

MAKE CHANGES

Review edits and craft a commit transaction

```
$ git status
```

Lists all new or modified files to be committed

```
$ git diff
```

Shows file differences not yet staged

```
$ git add [file]
```

Snapshots the file in preparation for versioning

```
$ git diff --staged
```

Shows file differences between staging and the last file version

```
$ git reset [file]
```

Unstages the file, but preserve its contents

```
$ git commit -m "[descriptive message]"
```

Records file snapshots permanently in version history

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

BASIC WORKFLOW

SYNCHRONIZE CHANGES

Register a repository bookmark and exchange version history

```
$ git fetch [bookmark]
```

Downloads all history from the repository bookmark

```
$ git merge [bookmark]/[branch]
```

Combines bookmark's branch into current local branch

```
$ git push [alias] [branch]
```

Uploads all local branch commits to GitHub

```
$ git pull
```

Downloads bookmark history and incorporates changes

BASIC WORKFLOW

REVIEW HISTORY

Browse and inspect the evolution of project files

```
$ git log
```

Lists version history for the current branch

```
$ git log --follow [file]
```

Lists version history for a file, including renames

```
$ git diff [first-branch]...[second-branch]
```

Shows content differences between two branches

```
$ git show [commit]
```

Outputs metadata and content changes of the specified commit

BASIC WORKFLOW

REDO COMMITS

Erase mistakes and craft replacement history

```
$ git reset [commit]
```

Undoes all commits after [commit], preserving changes locally

```
$ git reset --hard [commit]
```

Discards all history and changes back to the specified commit

BASIC WORKFLOW

REFACTOR FILENAMES

Relocate and remove versioned files

```
$ git rm [file]
```

Deletes the file from the working directory and stages the deletion

```
$ git rm --cached [file]
```

Removes the file from version control but preserves the file locally

```
$ git mv [file-original] [file-renamed]
```

Changes the file name and prepares it for commit

BASIC WORKFLOW

GROUP CHANGES

Name a series of commits and combine completed efforts

```
$ git branch
```

Lists all local branches in the current repository

```
$ git branch [branch-name]
```

Creates a new branch

```
$ git checkout [branch-name]
```

Switches to the specified branch and updates the working directory

```
$ git merge [branch]
```

Combines the specified branch's history into the current branch

```
$ git branch -d [branch-name]
```

Deletes the specified branch

BASIC WORKFLOW

SAVE FRAGMENTS

Shelve and restore incomplete changes

```
$ git stash
```

Temporarily stores all modified tracked files

```
$ git stash pop
```

Restores the most recently stashed files

```
$ git stash list
```

Lists all stashed changesets




```
$ git stash drop
```

Discards the most recently stashed changeset

SAFETY REGULATIONS

In case of fire



-  **1. git commit**
-  **2. git push**
-  **3. leave building**

HOMEWORK FOR NEXT CLASS

1. work in pairs (your next neighbor in class); last team may be a triple
2. create a new repository at GitHub (name it in whatever way makes sense to you) and make all of your team a collaborator
3. change the README.md to truthfully represent what this repository is (your first class project)
4. familiarize yourself with HTML/CSS/JS, e.g., using the suggestions provided at https://babe-project.github.io/babe_site/getting-started/introduction.html
5. create a website that uses all of HTML/CSS/JS to do/show whatever meets your fancy (e.g., display “Hello, _____!” on screen, with a text box to insert a string, which then shows up instead of _____)
6. publish your website using GitHub Pages