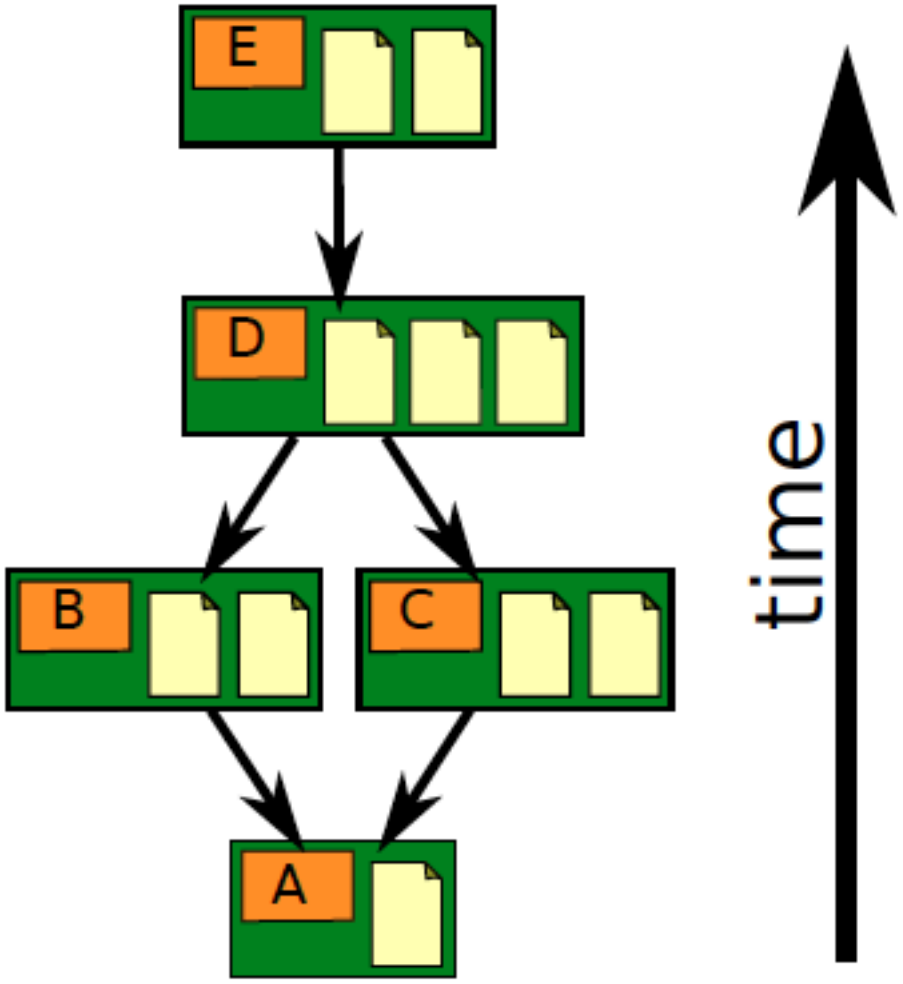
Some Git Basics

Git is a Version Control System

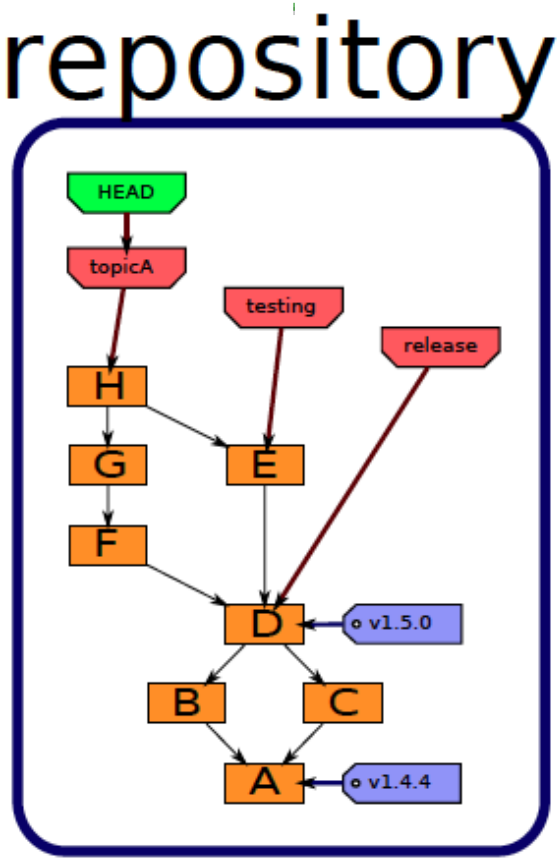
A VCS stores files in a repository.

Its keeps a history of all additions & changes, and prevents "old" versions from overwriting newer versions.

Adding new files or updating files is called a "commit".



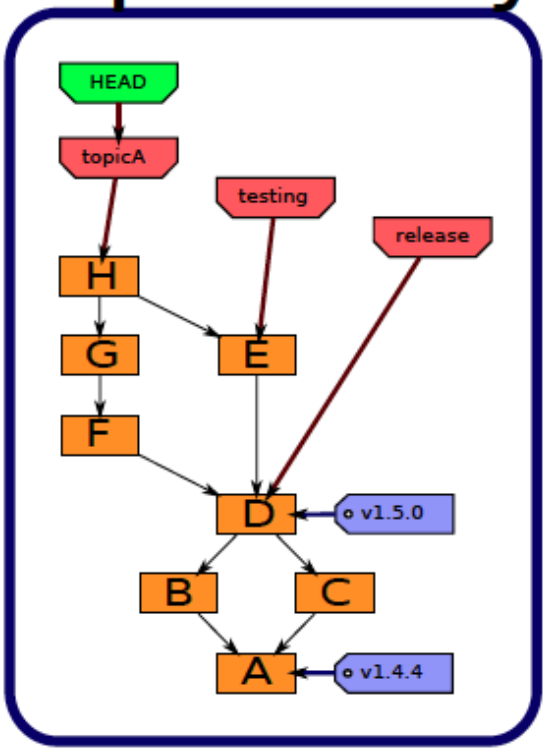
Git repository is stored in a directory



Git repository is usually stored in a subdirectory named **`.git`**

Staging area

repository



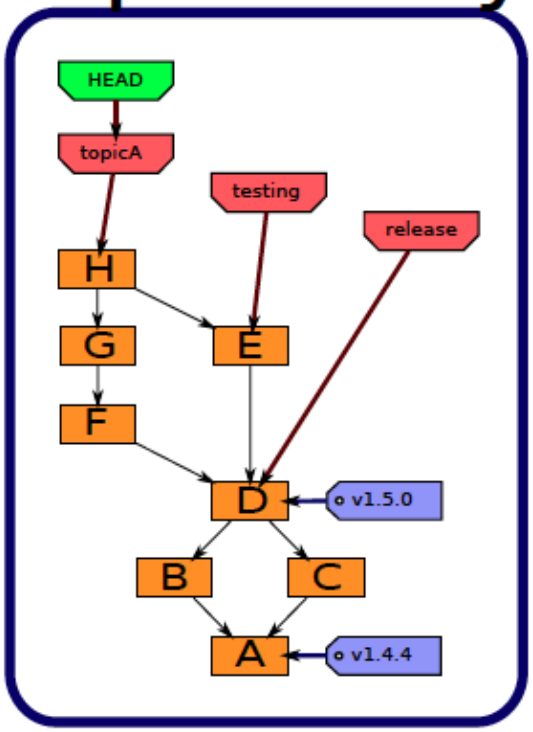
index

```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

"Staging area" or "index" is for files and other transactions waiting to be added/deleted/updated in the repository.

Working Copy

repository



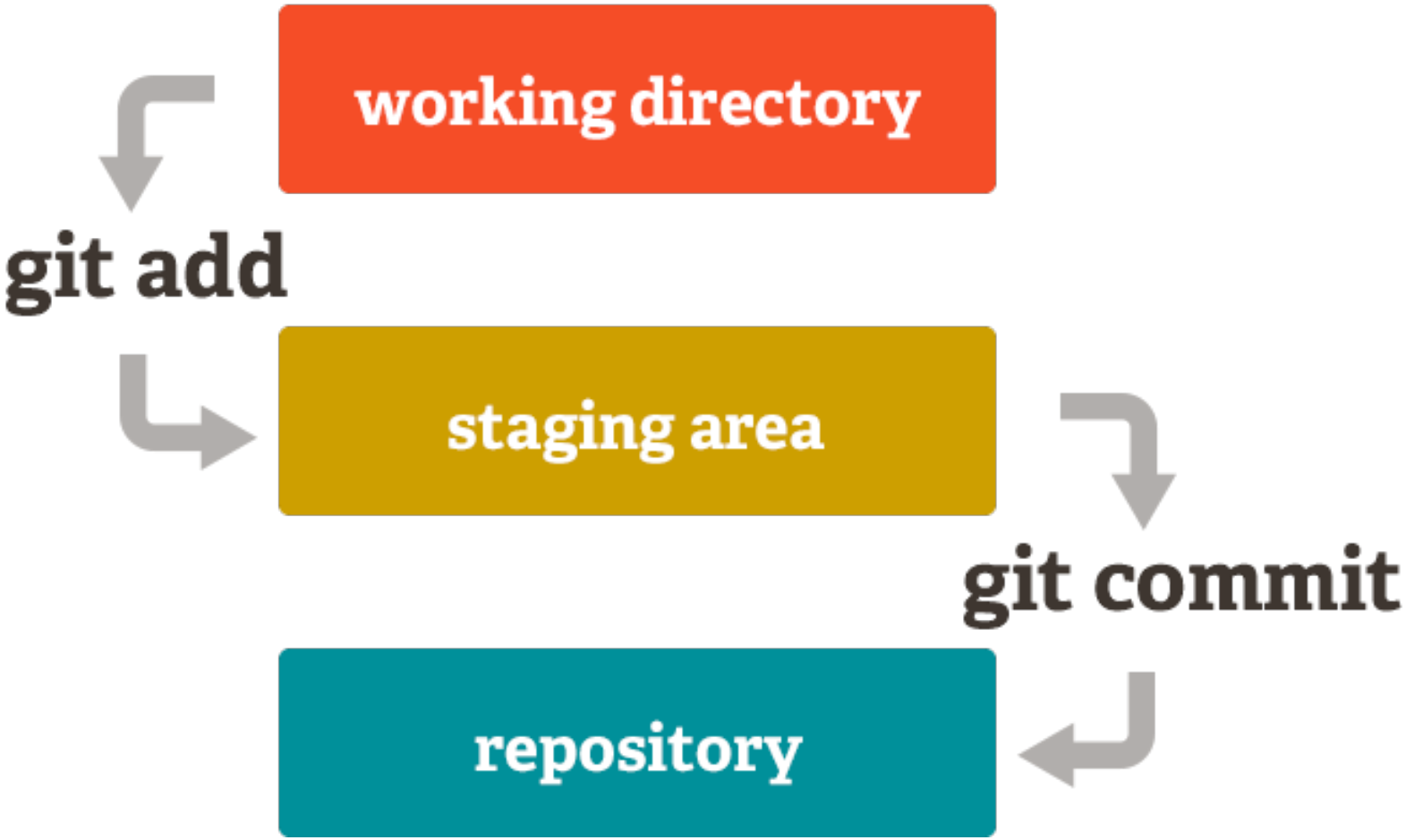
index

```
100644 20b024 0   bar
100644 1d52a6 0   baz
100644 20b024 0   sub/fi
100644 43dbe0 0   sub/foo
```

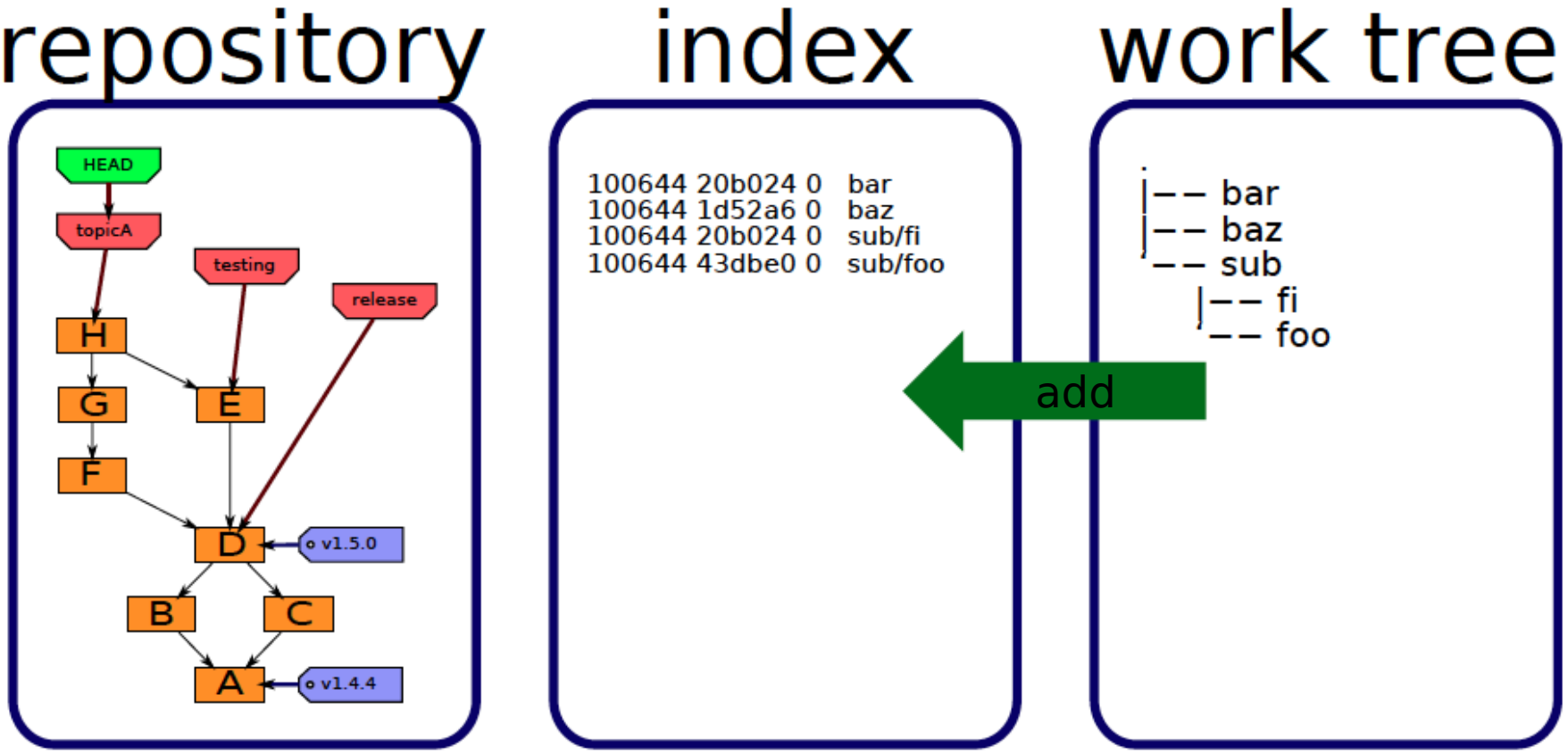
work tree

```
.
|-- bar
|-- baz
`-- sub
    |-- fi
    `-- foo
```

Your working copy contains both "tracked" files (files in repo) and "untracked" files (not included in repository).


working directory
git add
staging area
git commit
repository

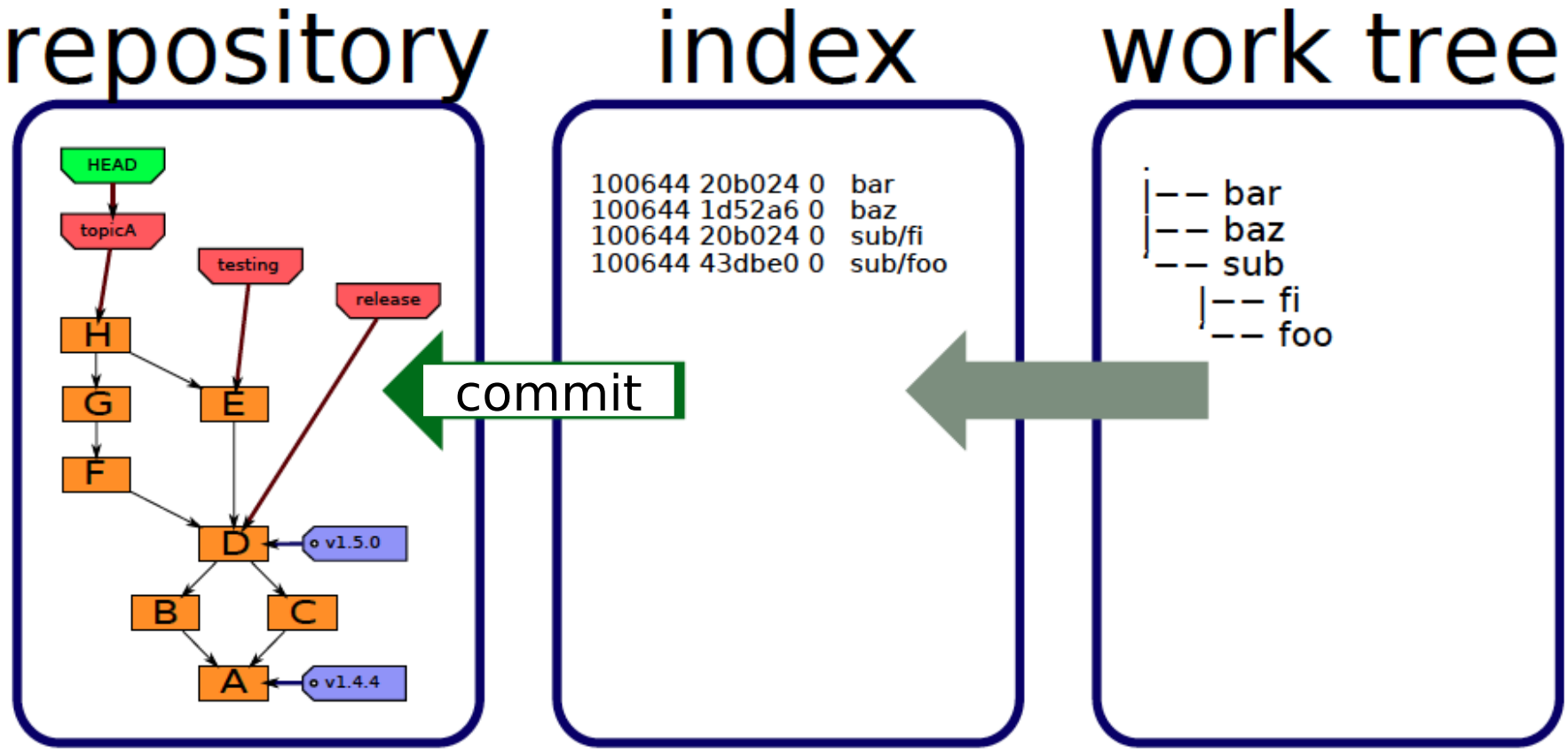
Adding files & updates to staging



There are 2 steps to storing files in the repository:

1) **mark** the files you want to add/update/move/delete
2) "**commit**" the changes

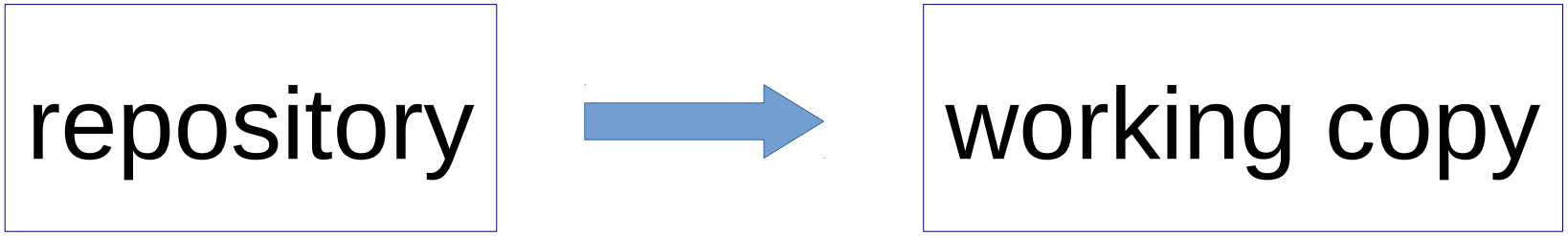
Commit: update work in the repo



"**commit**" updates the repository using the index (staging area). Each "commit" stores a snapshot of some work as a **new node** in the repository.

Checking out files

git checkout [commit-id]



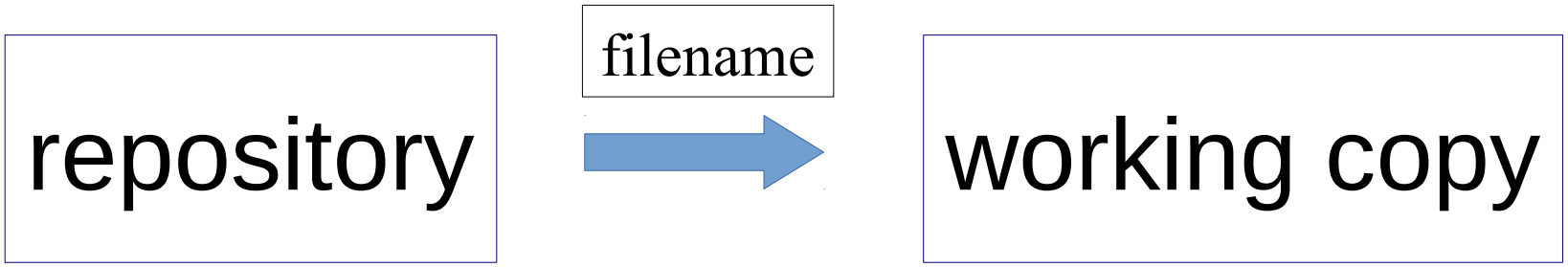
Update your working copy to match the current version, or any specified commit.

Untracked files are not affected.

Tracked files are only "checked out" if working copy is "clean", meaning no uncommitted changes.

Checking out a file

git checkout -- filename



You can checkout an individual file or files.

Useful if you accidentally delete or screw-up a file.

What to Include in a Repo?

Files you should include are:

1. source code
2. configuration and data files (but not sensitive data)
3. documentation
4. (optional) project plans and project documents
    - a wiki may be better for these

What NOT to save in repo

1. compiler output
2. anything you can recreate with a build tool.
   - For executable "releases" use Github "releases"
3. virtual environments, such as Python virtualenv.

   can be recreated using a `requirements.txt` file
4. IDE or editor project files, such as .vscode/ or .idea/
5. temporary files and log files

Essential Git

Essential commands to know are:

`git init`	create a new repository
`git status`	check status of your working copy
`git add`	add files to staging area
`git commit`	commit staged work to repo
`get checkout`	update working copy
`git log`	view history of commits
`git help` *`cmd`*	show help on any git command

Create a repo for existing code

Verify that git is installed and on your search PATH:

cmd> **`git --version`**

Change to your project directory

cmd> **`cd workspace/myproject`**

Create a new (empty) repository

cmd> **`git init`**

Adding Files

You need some files to add.

Create a file README.md containing description of your project. You can use Markdown mark-up.

cmd> **`edit README.md`** (any editor you like)

Add the file to "staging area"

cmd> **`git add README.md`**

Check Status & Commit

Always check status before commiting, to avoid mistakes.

cmd> **`git status`**

```
On branch master
Changes to be committed:
   (use "git rm --cached <file>..." to unstage)
new file:   README.md
```

Commit the files, -m "message" is a commit msg.

cmd> **`git commit -m "Initial code checkin"`**

```
 [master (root-commit) 51a8d5a] initial code checkin
 1 file changed, 5 insertions(+)
 create mode 100644 README.md
```

View History

There are several command for this, depending on system:

cmd> **`git log --oneline`**

cmd> **`git log1`**

cmd> **`gitk`** (a graphical tool)

```
51a8d5a (HEAD -> master) initial code checkin
```

Normal Workflow

1. Do some work and save files
2. a) Add files to staging area

   ```
   git add file1 file2 ...
   ```

   b) or rename existing files

   ```
   git mv oldname newname
   ```

   c) or delete files previously added to repo

   ```
   git rm unwanted_file
   ```

3. Check status: `git status`
4. Commit changes: `git commit -m "helpful msg"`

Commit a Unit of Work

Commit a group of files that represent a unit of work.

Avoid committing individual files.

OK to frequently commit files as protection against loss of work, then *squash the commits* later.

- squash is a git command to combine multiple commits

Write Descriptive Commit Msgs!

Good:

"add interface for money and update implementing classes"

"fix issue #3 in recursive withdraw"

Bad:

"finish"

"add interface"

"commit"

.gitignore

To avoid accidentally adding unwanted files a repository, add a file named `.gitignore` to the top directory of your repo. This is common practice.

`.gitignore` contains names or *patterns* for files and directories that git should ignore -- that is, never add to a repository (unless you force it to).

See next slide for example.

Github can create a `.gitignore` for you when you create a repository! This is a good way to see an example of what you might put in .gitignore.

Examples what not to save

```
# Example .gitignore for a Python project
__pycache__
*.py[cod]    this means *.pyo or *.pyc or *.pyd
.coverage    output of code coverage app
*.log
# Intellij files
.idea/
*.iml
# Eclipse, Pydev, and VSCode files
.settings
.project/
.vscode/
# Virtual env files (common directory names)
venv/
env/
```

References

Tutorial from the Git home

`https://git-scm.com/docs/gittutorial`

Pro Git Book - free on git-scm.com

`https://git-scm.com/book/en/v2`

Git Intro (PDF) from U. of Washington

https://courses.cs.washington.edu/courses/cse391/18su/lectures/9/391Lecture09-Git-18su.pdf

Git Tutorial on freecodecamp

https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/

Command Line Essentials

Developers need to know how to use a terminal or "shell" window. Essential commands are:

- move from one directory to another
- list files in a directory
- find your home directory
- find files in the file system
- rename, move, and delete files
- edit a text file
- how to use and modify the PATH