LECTURE 39

VERSION CONTROL

MCS 275 Spring 2023

Emily Dumas

LECTURE 39: VERSION CONTROL

Reminders and announcements:

- Please **complete your course evaluations**.
- Homework 14 (the last!) due Tuesday
- Project 4 autograder is open
- Projects requiring subdirectories should upload as a ZIP file

VERSION CONTROL

A system to:

- Track changes
- Document changes
- Archive previous versions
- Allow concurrent work

Version control systems (VCS) are also known as "source code management" (SCM).

DO YOU HAVE THIS?

```
project4.py
project4draft.py
project4-new.py
project4-fixed.py
project4-fixed-debug.py
project4final.py
project4final2.py
project4final3.py
project4final3 (1).py
project4final_fixed-new2_revised\ (1).2022-04-27.py
```

A version control system (VCS) can help.

VCS

Some version control systems:

- Historically important
  - 1970s: VAX/VMS filesystem has versioning
  - 1980s: Revision Control System (RCS)
  - 1990s: Concurrent Versions Systems (CVS)
  - 2000s: Subversion (SVN)
- Widely used today
  - **git**
  - fossil, mercurial, ...

GIT

A VCS created by **Linus Torvalds** in 2005.

Key properties:

-
-
-
-

GIT

A VCS created by **Linus Torvalds*** in 2005.

Key properties:

-
-
-
-

* Finnish software developer and creator of Linux (1993).

GIT

A VCS created by **Linus Torvalds*** in 2005.

Key properties:

- **Open source**
-
-
-

Free to use; multiple implementations available.

GIT

A VCS created by **Linus Torvalds*** in 2005.

Key properties:

- **Open source**
- **Distributed**
-
-

Everyone has a copy of full history.

GIT

A VCS created by **Linus Torvalds*** in 2005.

Key properties:

- **Open source**
- **Distributed**
- **Nonlinear**
-

Supports parallel branches of development; no concept of a single "latest" version.

GIT

A VCS created by **Linus Torvalds*** in 2005.

Key properties:

- **Open source**
- **Distributed**
- **Nonlinear**
- **Offline-friendly**

Many commands operate only on local files. Sync with others when ready.

ONLINE SERVICES

There are some popular online services that will keep a copy of your repository on a server and/or let you interact with it in a browser.

- gitlab
- github
- AWS CodeCommit

These let you voluntarily centralize a purposely decentralized system.

PROJECT

```
/myflask/main.py
/myflask/dbutil.py
/myflask/templates/front.html
/myflask/static/myflask.css
```

REPOSITORY

```
/myflask/.git
/myflask/main.py
/myflask/dbutil.py
/myflask/templates/front.html
/myflask/static/myflask.css
```

Hidden database of previous versions, comments, etc.

`git init`

Creates a git repository in the current directory.

Initially has empty history and doesn't track any files.

DATA LIFECYCLE

DATA LIFECYCLE

Untracked | Tracked | Staged | Repository | Remote

Files that git ignores

DATA LIFECYCLE

Untracked | Tracked | Staged | Repository | Remote

Files that git monitors for changes

DATA LIFECYCLE

Untracked | Tracked | Staged | Repository | Remote

Files ready to commit to repository

DATA LIFECYCLE

Untracked | Tracked | Staged | Repository | Remote

Database of commits (content versions)

DATA LIFECYCLE

Untracked | Tracked | Staged | Repository | Remote

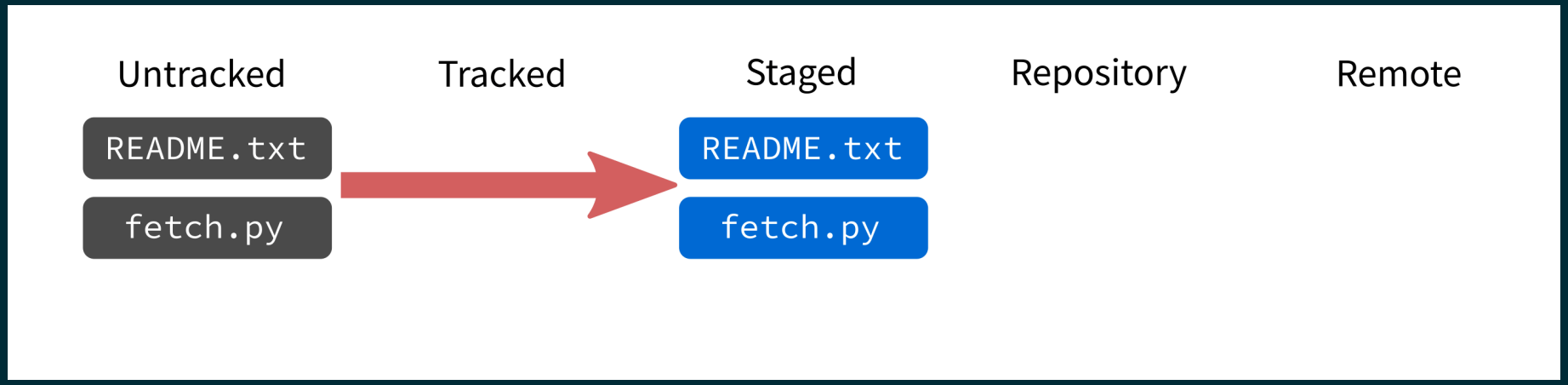
Repository stored elsewhere (e.g. GitHub)

DATA LIFECYCLE

git add

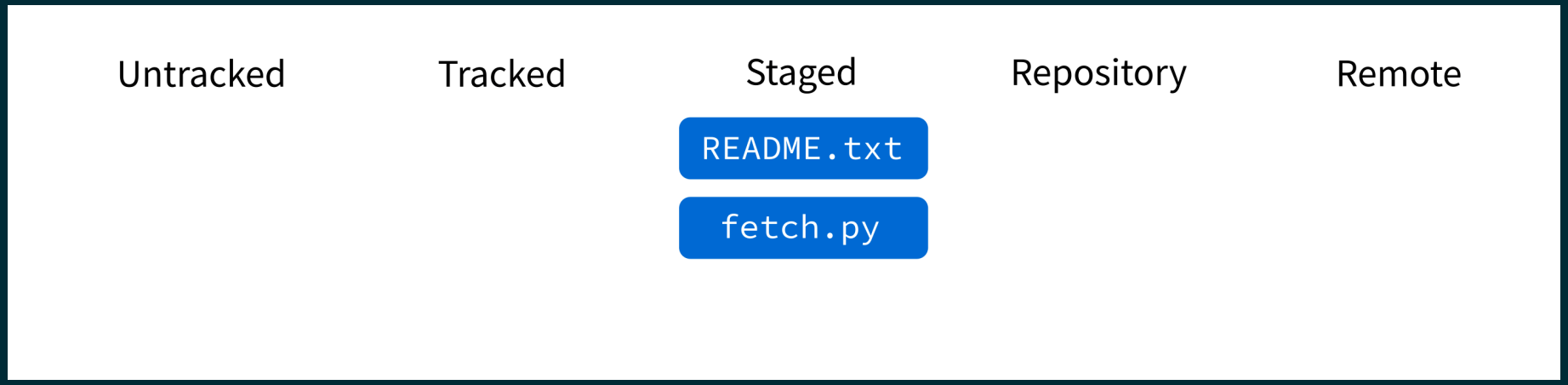
Put current version of the file in a staging area.

git add



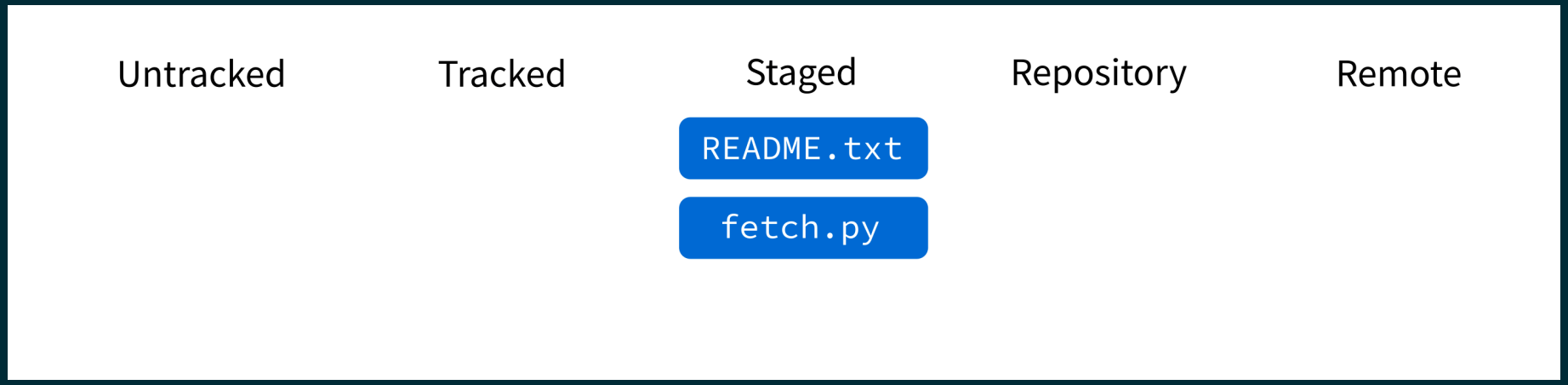
Put current version of the file in a staging area.

git add



Put current version of the file in a staging area.

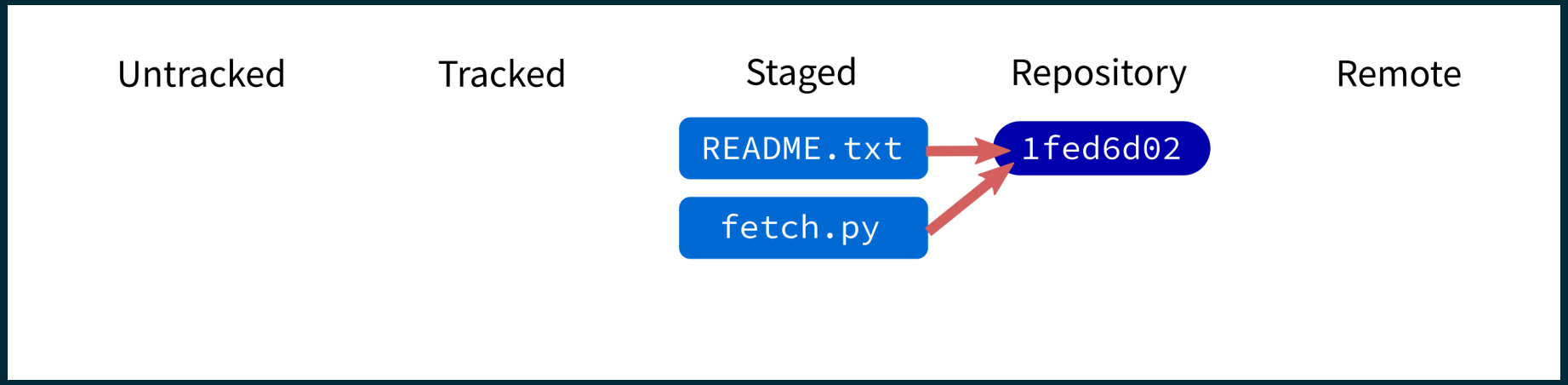
git commit



Record staged changes in the database.

(These files will be tracked from now on.)

git commit



Record staged changes in the database.

(These files will be tracked from now on.)

git commit

Record staged changes in the database.

(These files will be tracked from now on.)

`git log`

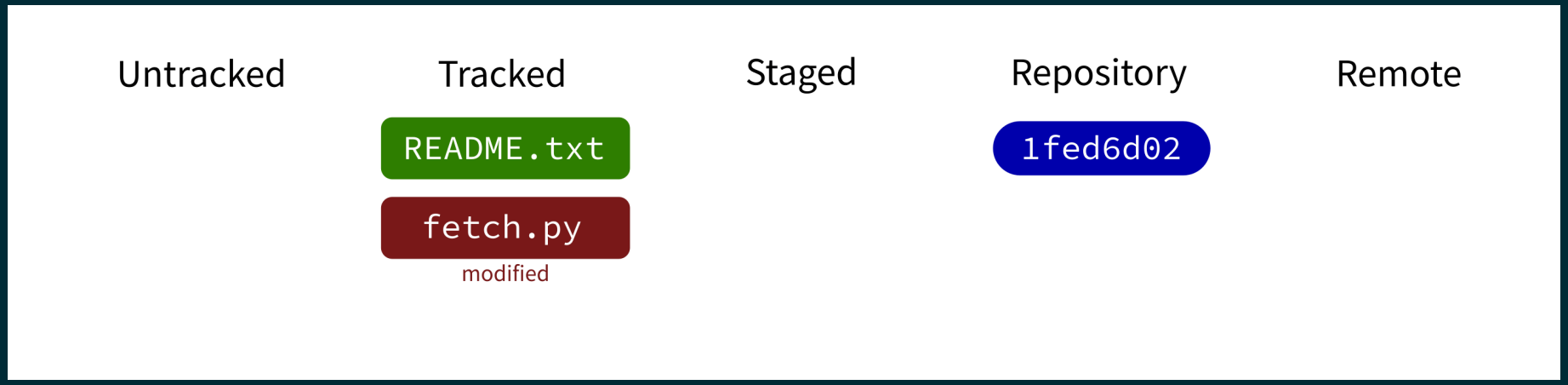
Show recent commits and descriptions.

`git status`

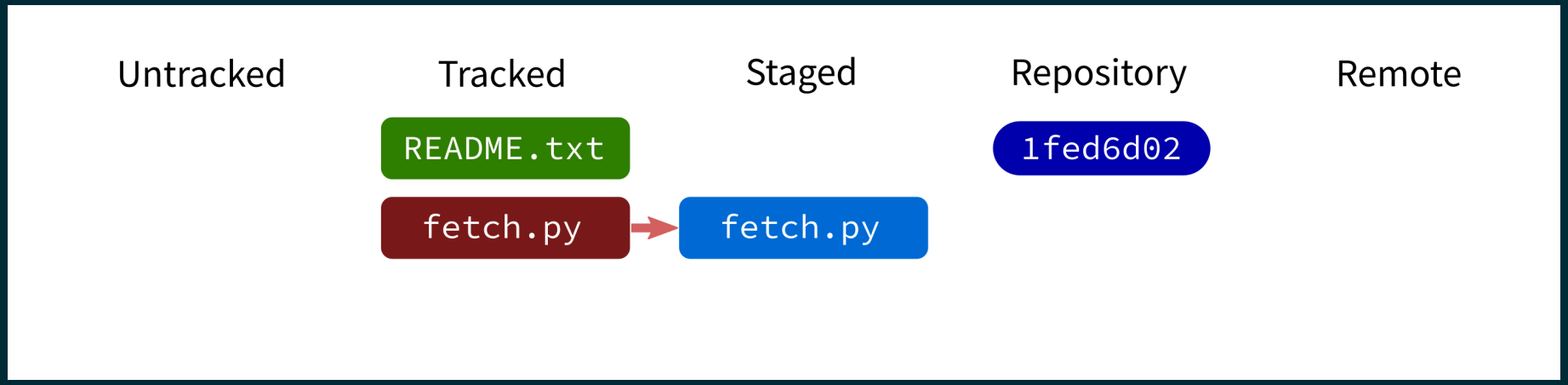
Show summary of current situation.

ANOTHER COMMIT

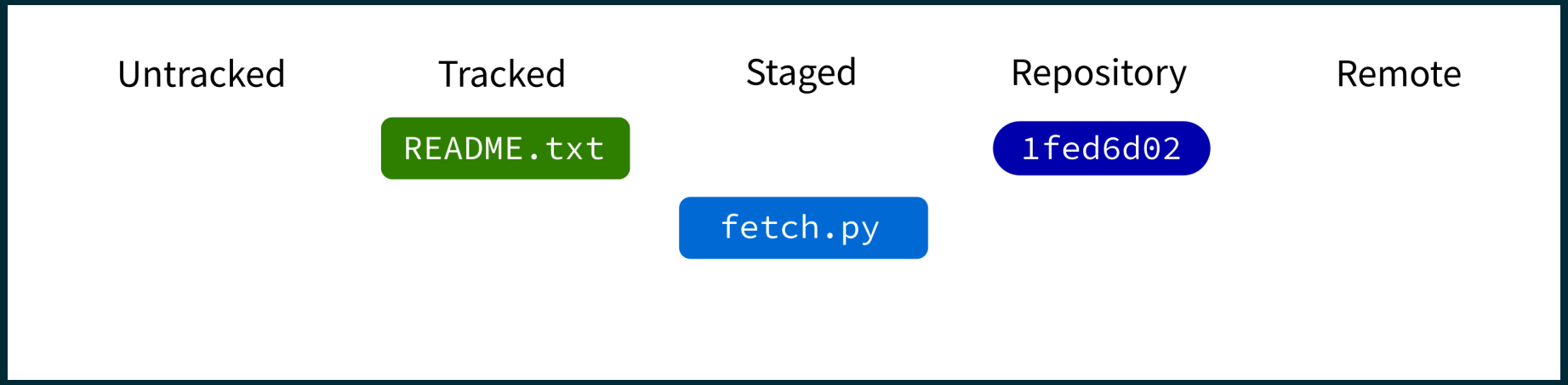
ANOTHER COMMIT



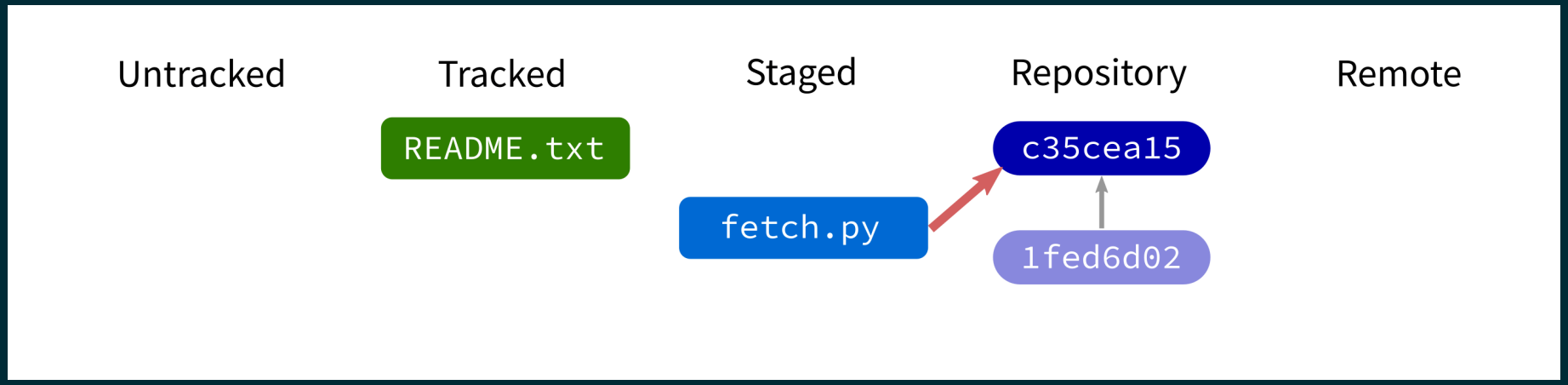
ANOTHER COMMIT



ANOTHER COMMIT



ANOTHER COMMIT



ANOTHER COMMIT

git push

Contact a remote repository and send it commits that are in our database but not theirs.

Fails if remote has changed since our last push!

git push

Contact a remote repository and send it commits that are in our database but not theirs.

Fails if remote has changed since our last push!

git push

Contact a remote repository and send it commits that are in our database but not theirs.

Fails if remote has changed since our last push!

git pull

Contact a remote repository and get commits from its database that are not yet in ours.

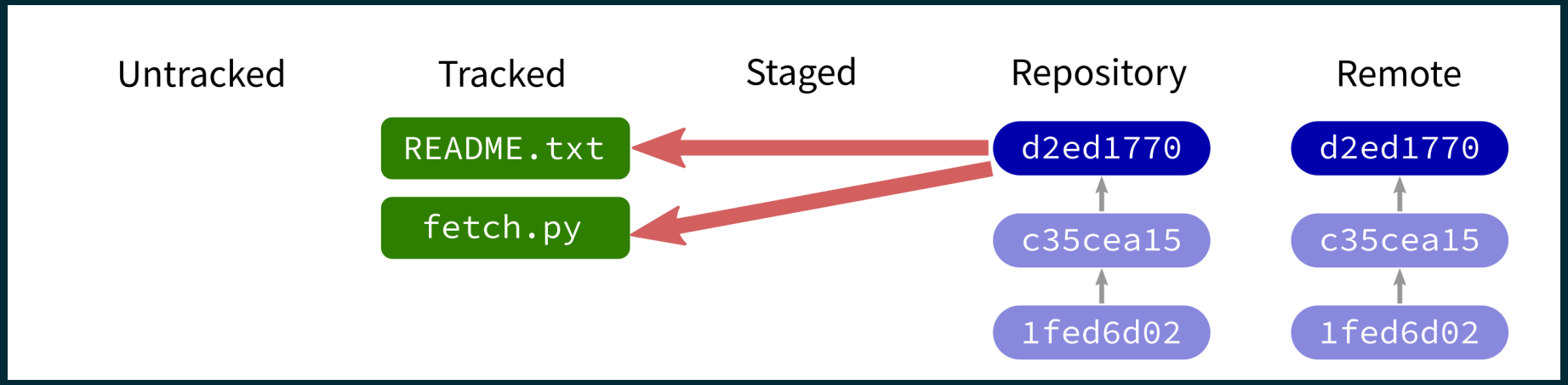
May trigger a **merge** if there have been changes to both local and remote since we last pulled.

git pull

Contact a remote repository and get commits from its database that are not yet in ours.

May trigger a **merge** if there have been changes to both local and remote since we last pulled.

git pull



Contact a remote repository and get commits from its database that are not yet in ours.

May trigger a **merge** if there have been changes to both local and remote since we last pulled.

git pull

Contact a remote repository and get commits from its database that are not yet in ours.

May trigger a **merge** if there have been changes to both local and remote since we last pulled.

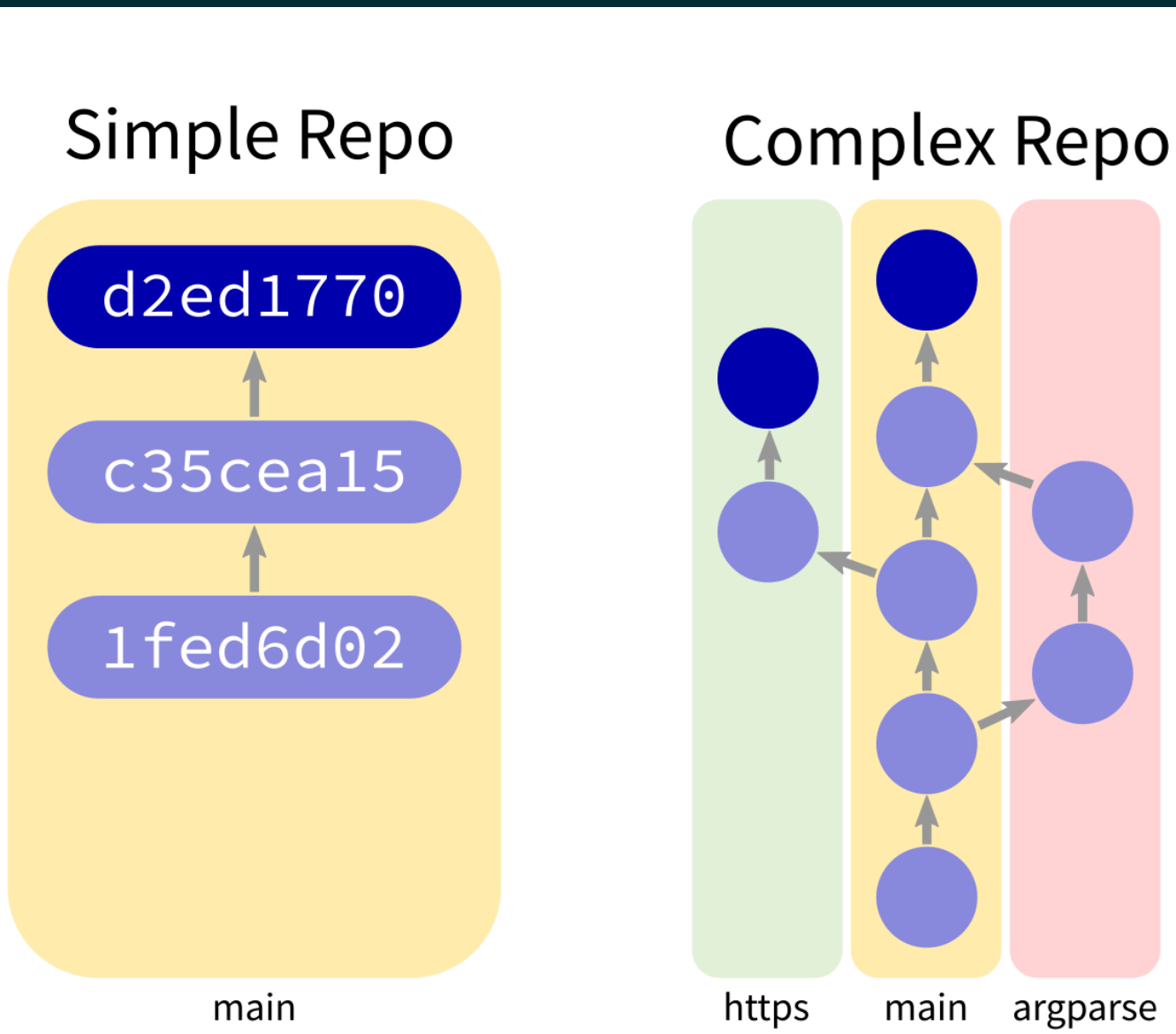
LOOKING AT HISTORY

```
git show COMMIT:FILE
```

will display file contents at any commit.

GIT CLONE

Make a local copy of an existing repository (from URL, directory, ...).


Simple Repo
d2ed1770
c35cea15
1fed6d02
main
Complex Repo
https
main
argparse

NOT COVERED TODAY

- **checkout** – change which version is seen in the filesystem
- **reset** – set files and/or DB back to a previous state
- **branch** – name a series of commits

REFERENCES

- git home page
- Official git documentation (includes tutorial videos, Pro Git book)
- git - the simple guide (nice practical introduction, with mild profanity)

REVISION HISTORY

- 2022-04-29 Last year's lecture on this topic finalized
- 2023-04-23 Updated for 2023