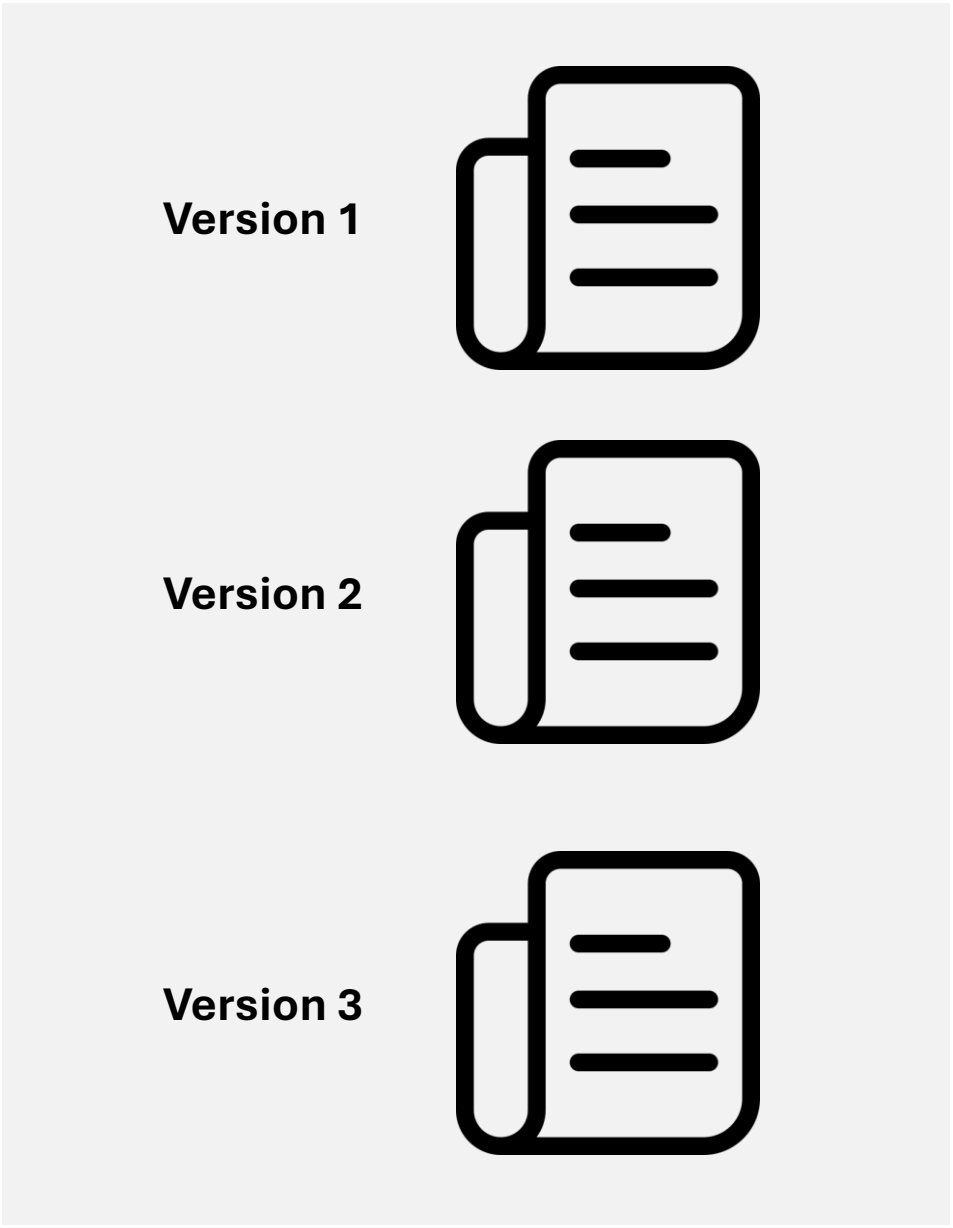
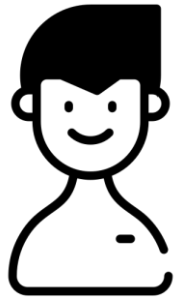
Introduction to Git

Pr. Imane Fouad

Why ?

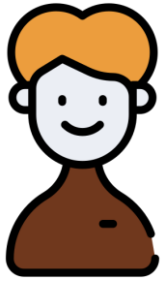


Why ?

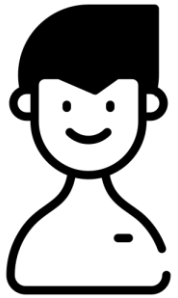


Task 1

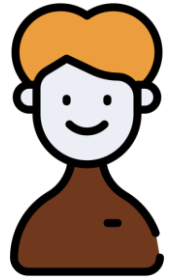
Task 2



Task 3



Task 1
+
Task 2
+
Task 3



Why ?

Sent to Alex

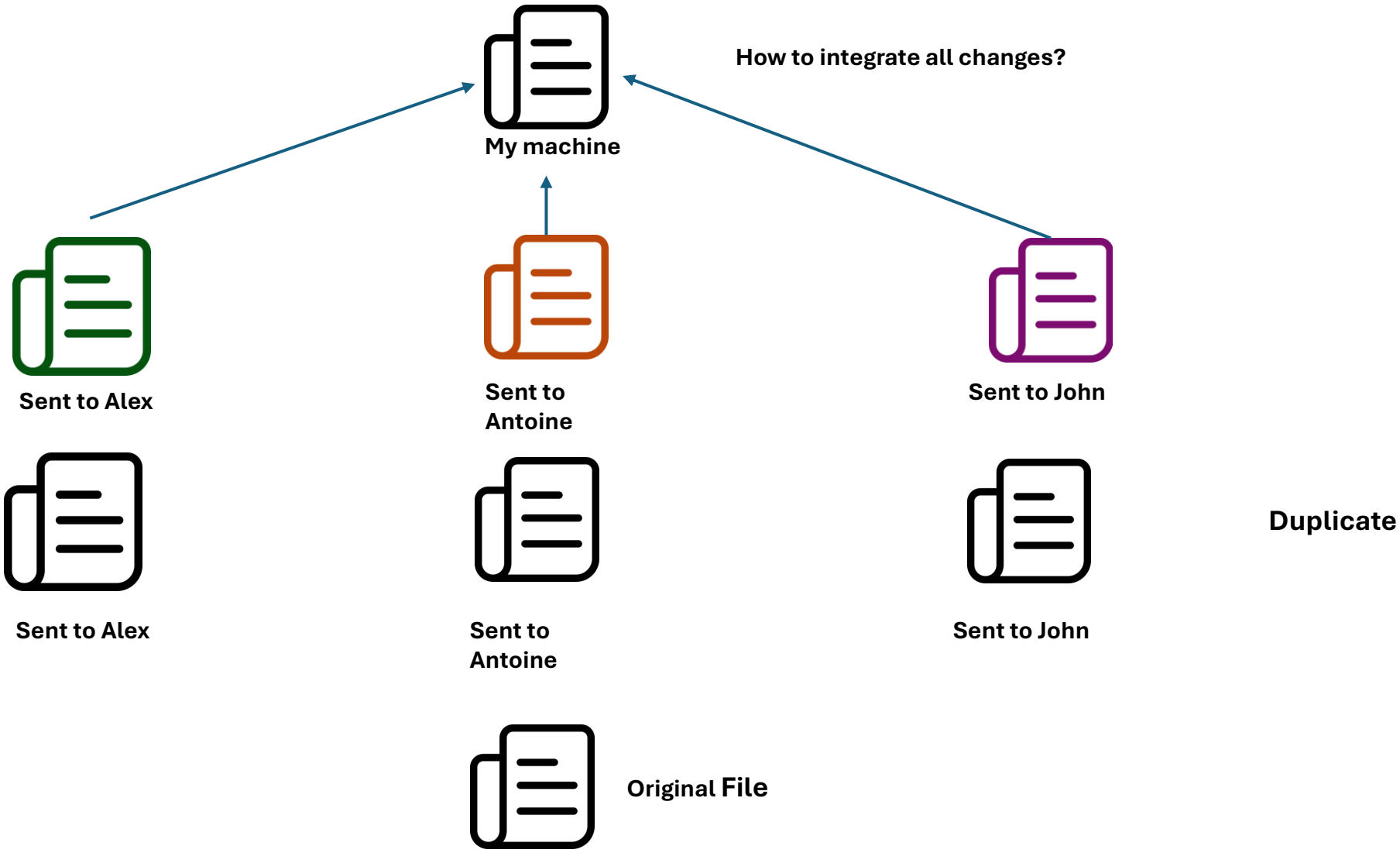
Sent to Antoine

Sent to John

Duplicate

Original File

Why ?


How to integrate all changes?
My machine
Sent to Alex
Sent to Antoine
Sent to John
Sent to Alex
Sent to Antoine
Sent to John
Duplicate
Original File

Challenges - Software Development

- A software project is a **long and complex activity**
- Involves **many files** (sometimes thousands)
- Requires **multiple iterations**
- At times, we need to mark **versions or variants** of the software
- **Mistakes happen** → rolling back may be necessary
- Projects are often done by **teams**, with developers working on the same files (→ conflicts)

Version Control Systems (VCSs)

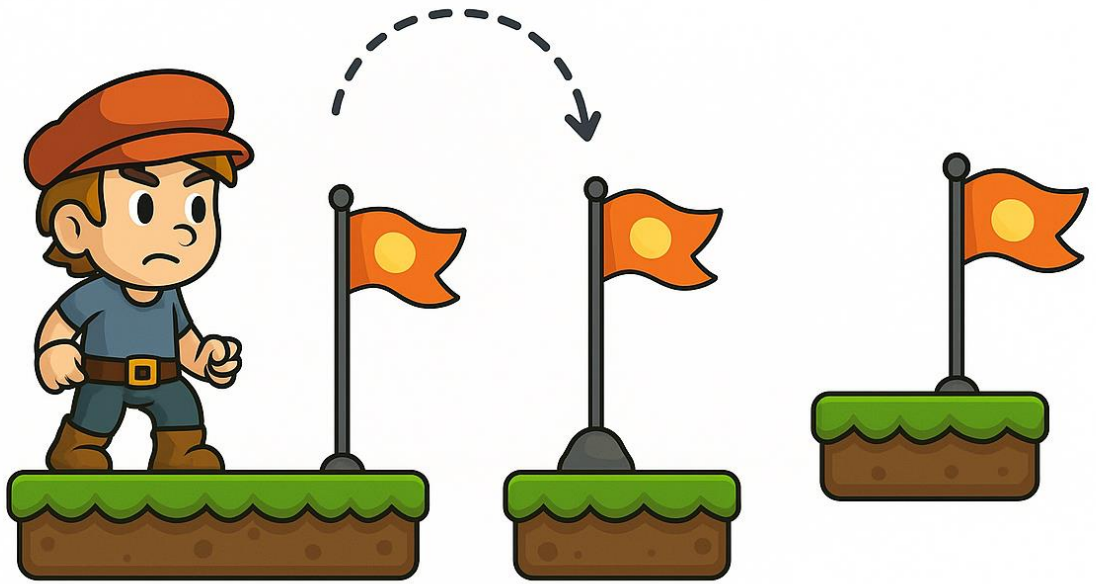
- Also known as reversion or source code control
- Revision control systems (RCS) maintain …
  - a history of changes
  - from multiple persons
  - to a set of documents.

Version Control Systems (VCSs)

- **Backup**: Undo or refer to old stuff
- **Branch**: Maintain old release while working on new
- **Collaborate**: Work in parallel with teammates

Version Control System

Advantages of Version Control



What to Put in a Version Control System?

All project sources

- Source code files (.c, .cpp, .java, .py, etc.)
- Build scripts (Makefile, pom.xml, etc.)
- Documentation (.txt, .tex, README, etc.)
- Resources (images, media files, etc.)
- Various scripts (deployment scripts, .sql, .sh, etc.)

What to Put in a Version Control System?

All project sources

- Source code files (.c, .cpp, .java, .py, etc.)
- Build scripts (Makefile, pom.xml, etc.)
- Documentation (.txt, .tex, README, etc.)
- Resources (images, media files, etc.)
- Various scripts (deployment scripts, .sql, .sh, etc.)

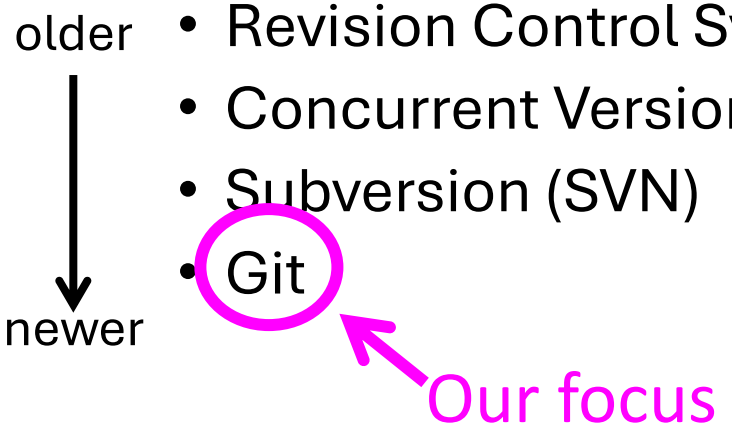
What NOT to Include:

Generated files

- Compilation results (.class, .o, .exe, .jar, etc.)

Version Control Systems (VCSs)

- Help you track/manage/distribute revisions
- Standard in modern development
- Examples:
  - Revision Control System (RCS)
  - Concurrent Versions System (CVS)
  - Subversion (SVN)
  - Git



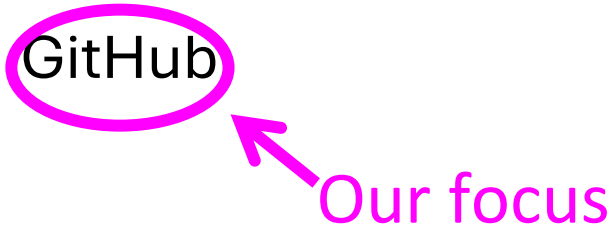
Why the git ?

It’s The Standard for Version Control

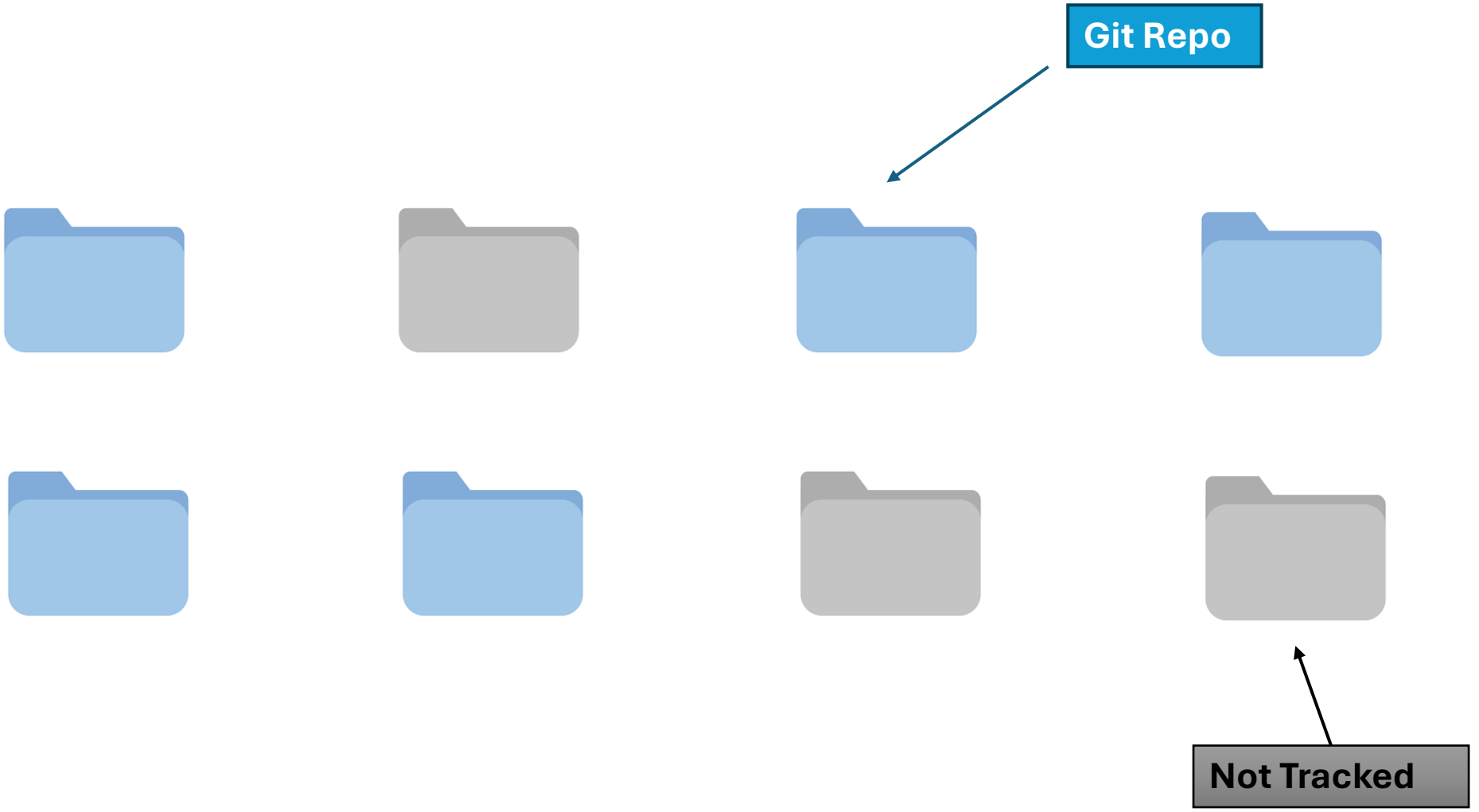
- Git - “the stupid content tracker”
- Created by Linus Torvalds in 2005
- Professional-grade tool:
  - Fast
  - Cross-platform
  - Powerful
  - Fully distributed

Version Control Hosting Services

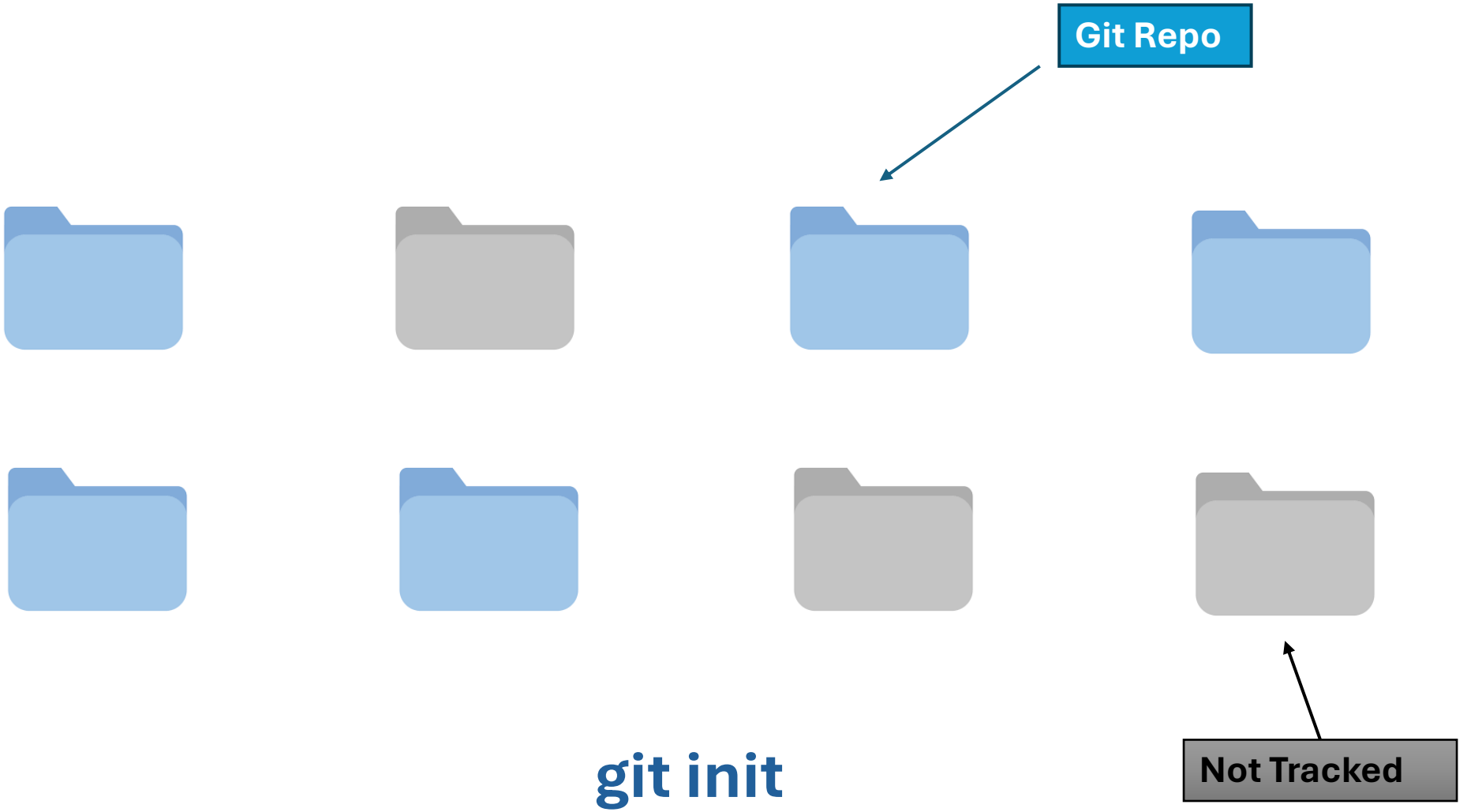
- Enable sharing version control repos
- Internet/Web based
- Examples:
  - SourceForge
  - Bitbucket
  - GitLab
  - GitHub



How Git Works?



How Git Works?



git init

How Git Works?

/localWorkingDir

Staging Area

Local Repository

.git

How Git Works?

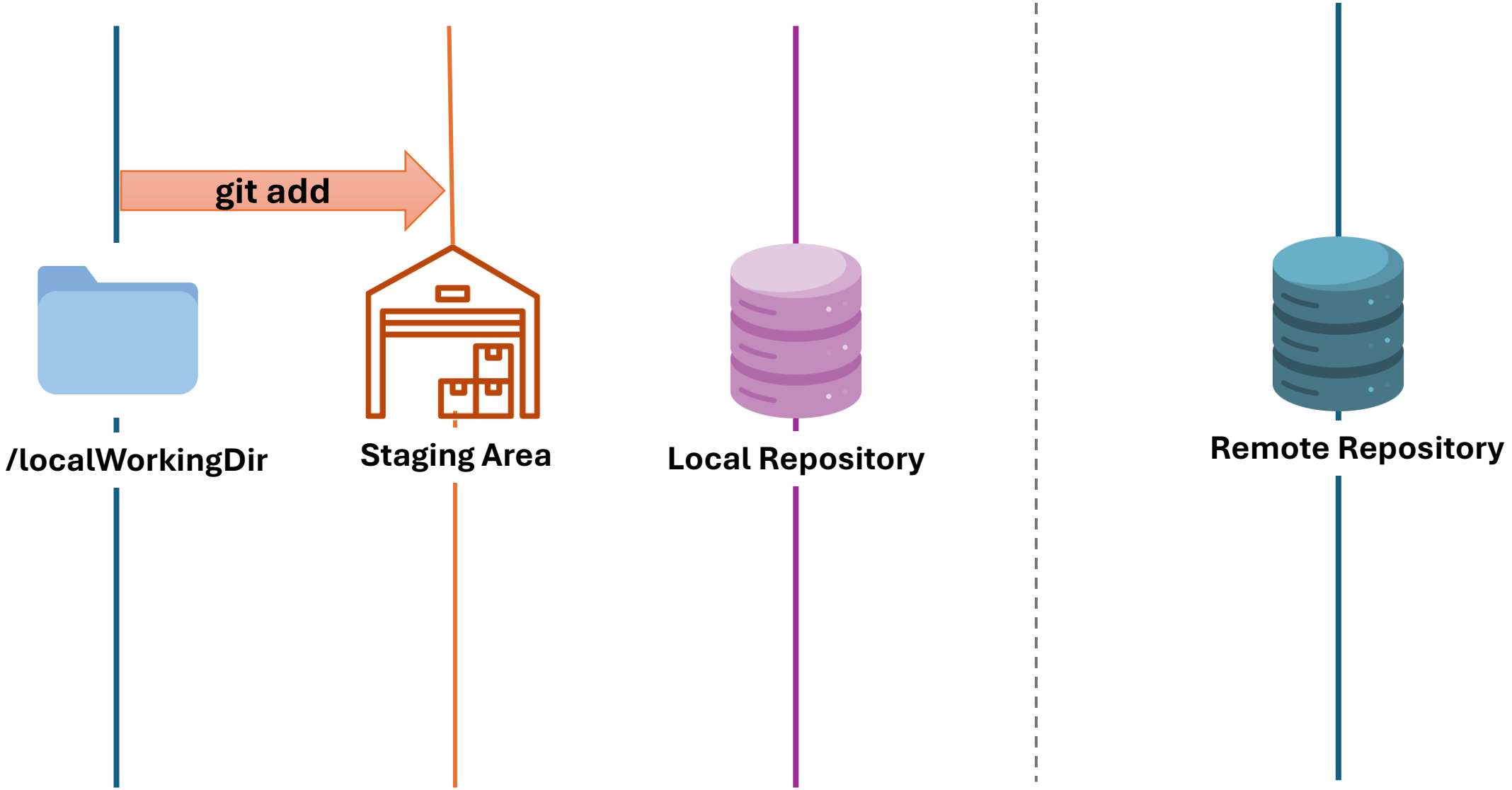
/localWorkingDir

Staging Area

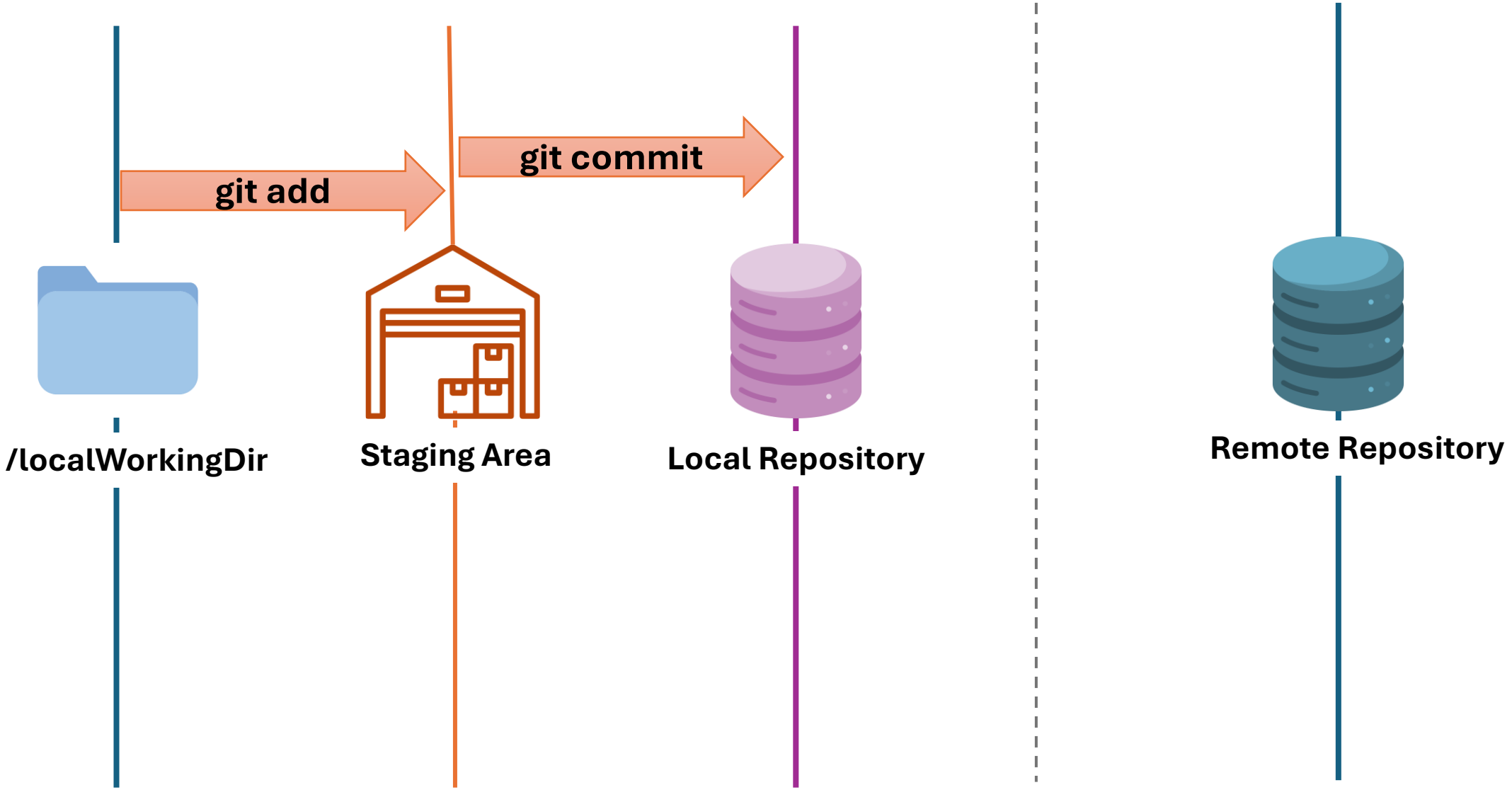
Local Repository

Remote Repository

How Git Works?



How Git Works?

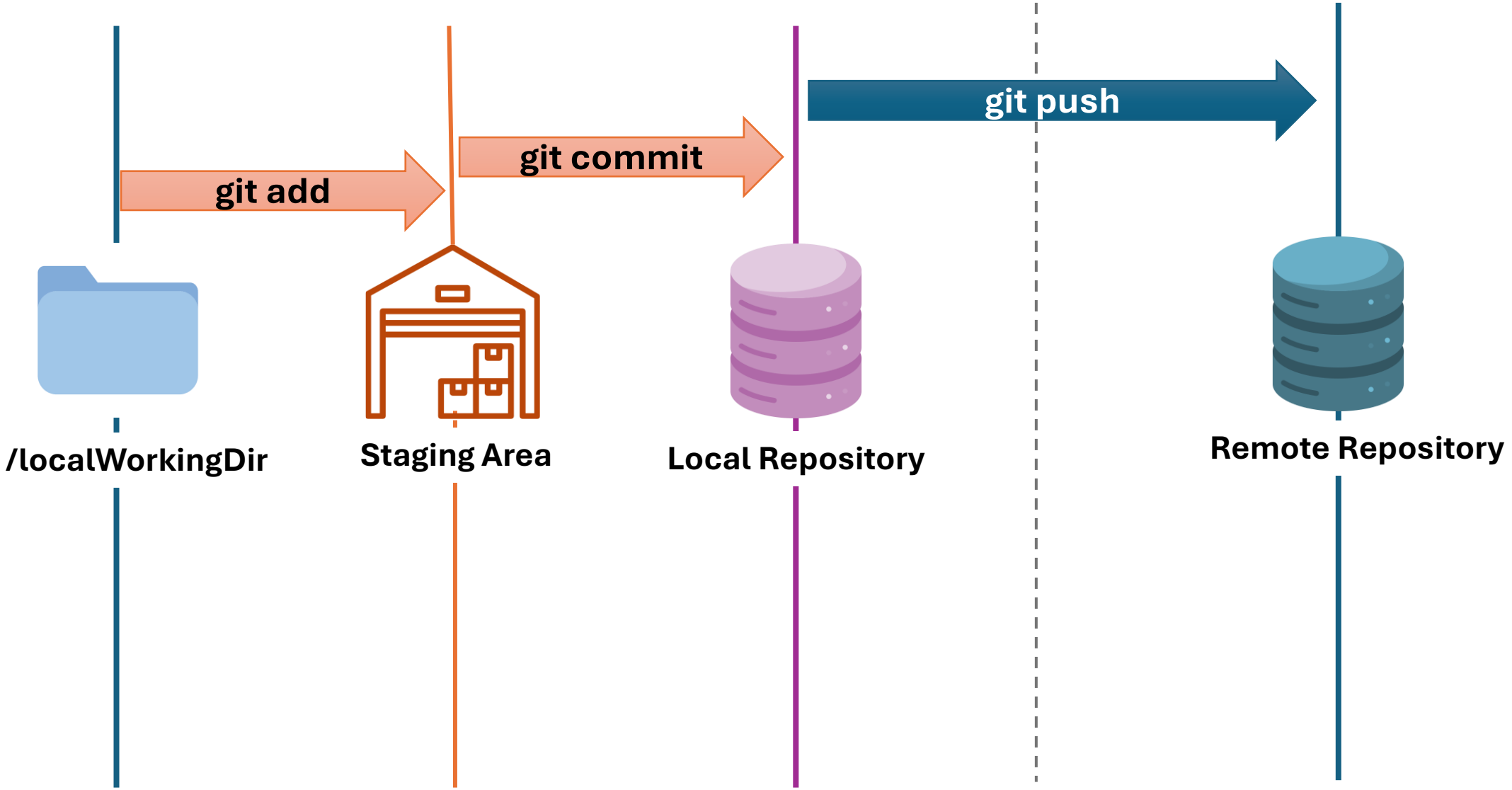


Atomic Commits

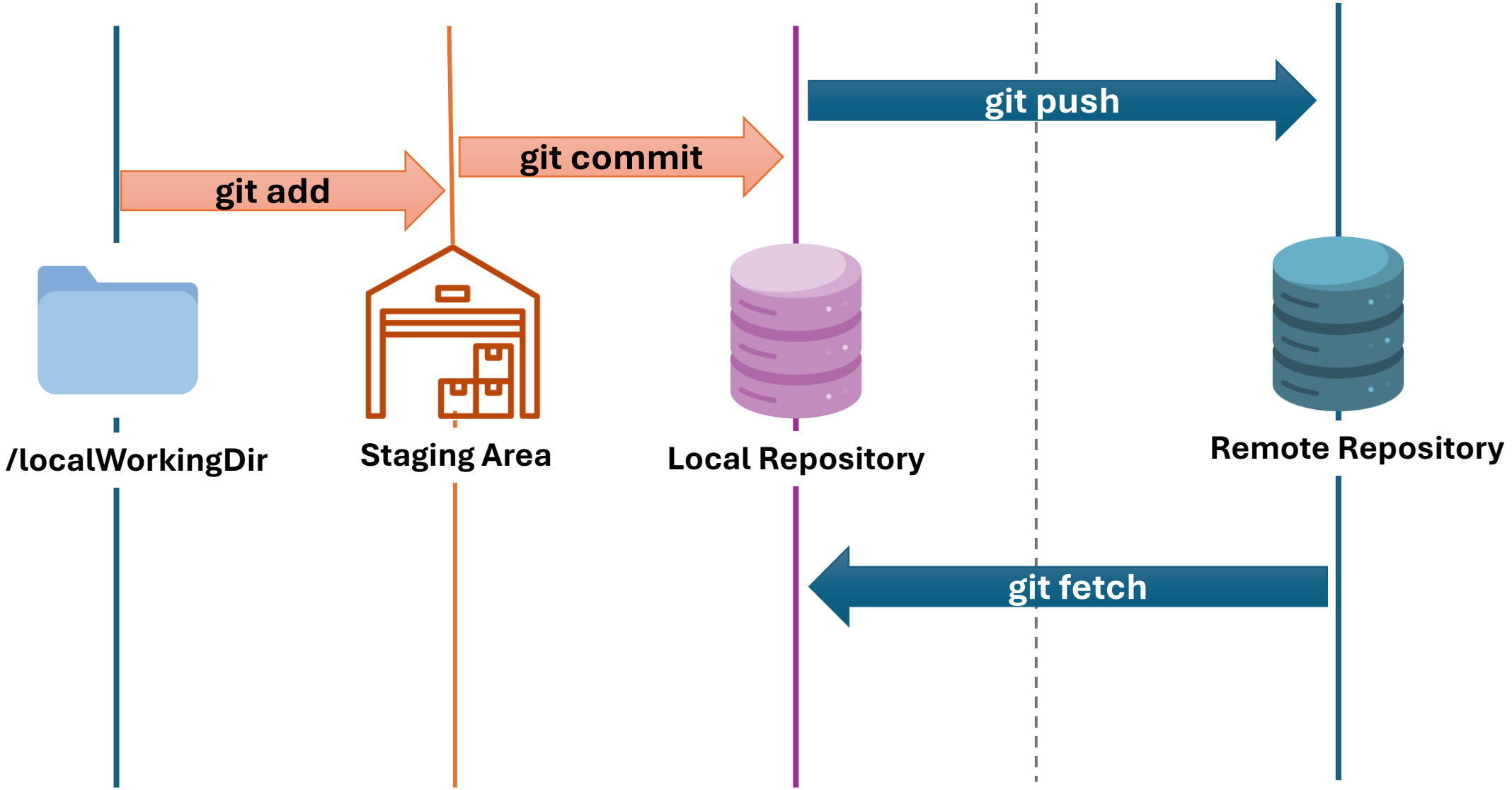
- **Atomic Commit = One logical change per commit**
- Each commit should do **one thing only**.
- Keeps history **clear, readable**, and **reversible**.
- Helps with **debugging** and **code reviews**.
- Example of atomic: "Fix typo in README" ✅

  Non-atomic: "Update README and fix login bug" ❌

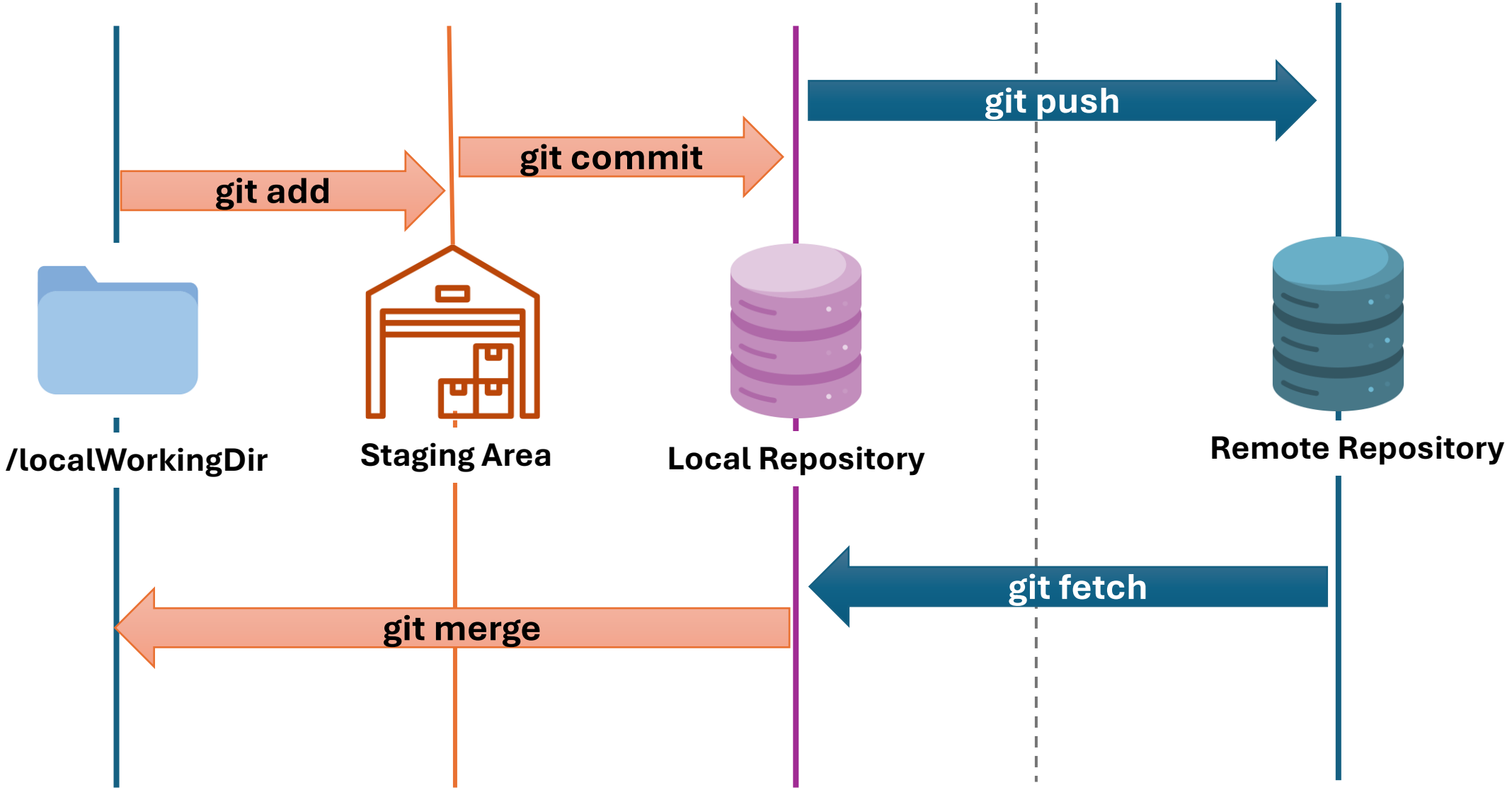
How Git Works?



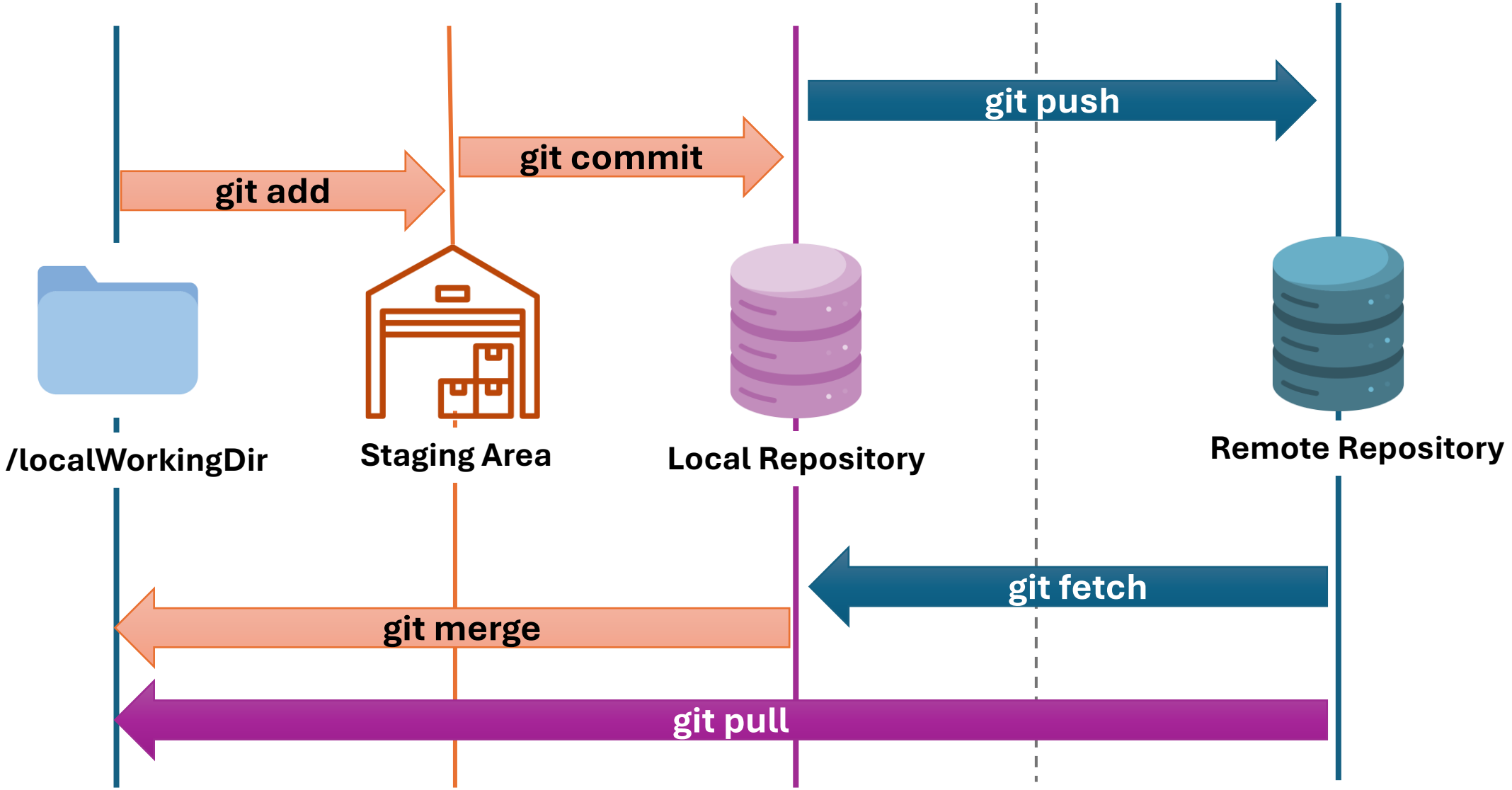
How Git Works?



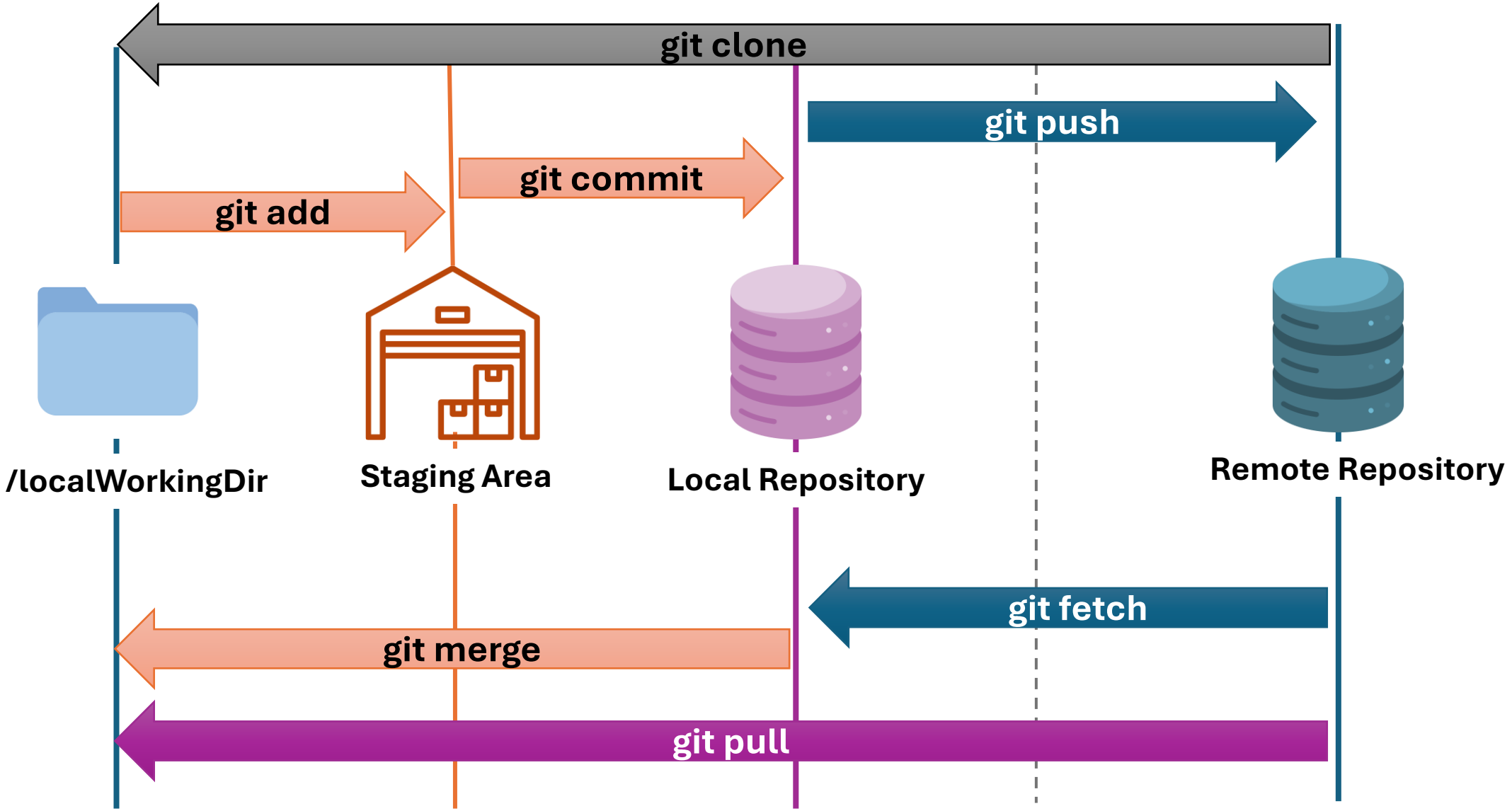
How Git Works?



How Git Works?



How Git Works?



Install Git

1 Download Git

- Go to: https://git-scm.com/downloads
- Choose your operating system (Windows, macOS, Linux)

2 Run the Installer

- Open the downloaded file
- Follow the setup wizard

3 Verify Installation

- Open a terminal (Command Prompt, PowerShell, or Terminal):
  Run: git --version

Example – Creating a Git Repository

(a) Empty repo

In a terminal...

```
mkdir my_repo
cd my_repo
git init .
```

Example – Creating a Git Repository

(a) Empty repo

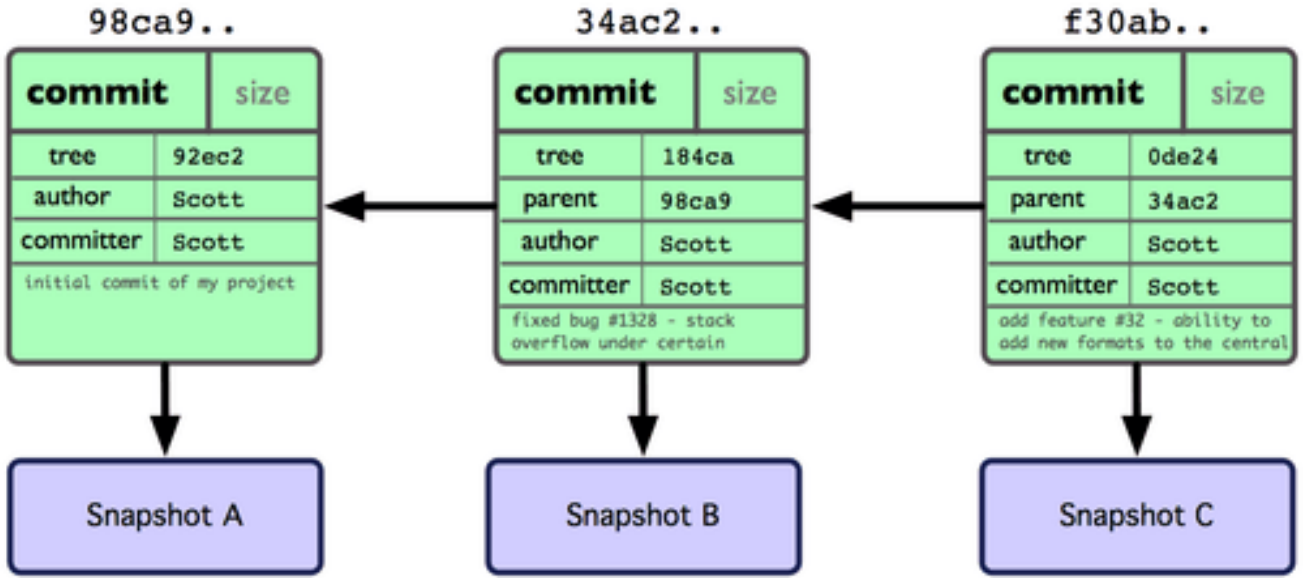
In a terminal...

```
mkdir my_repo
cd my_repo
git init .

echo apple >> fruits.txt
git add fruits.txt
git commit -m "Added apple to fruit list"
```

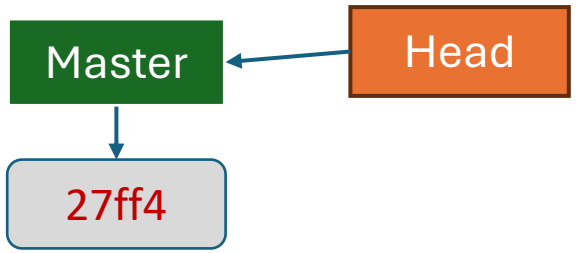
=> Commit ID generated: 27ff4

What Is a Commit ID?



- A **unique identifier** for every commit.
- Looks like a long string of letters and numbers (e.g. 27ff4d8e...).
- Created automatically by Git using **SHA-1 hashing**.
- Helps you **track**, **compare**, or **revert** specific changes.

Example – Creating a Git Repository



(a) First commit

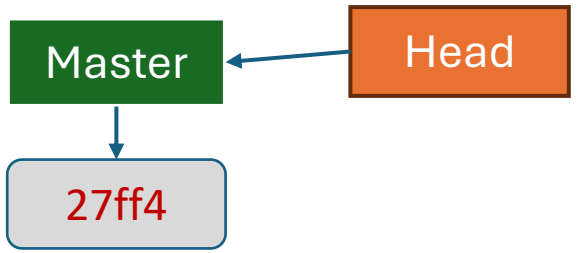
In a terminal...

```
mkdir my_repo
cd my_repo
git init .

echo apple >> fruits.txt
git add fruits.txt
git commit -m "Added apple to fruit list"
```

=> Commit ID generated: 27ff4

Example – Creating a Git Repository



(a) First commit

In a terminal…

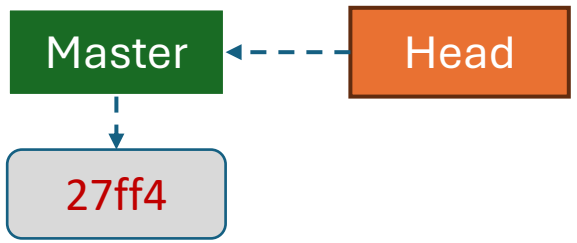
```
mkdir my_repo
cd my_repo
git init .

echo apple >> fruits.txt
git add fruits.txt
git commit -m "Added apple to fruit list"
```

Always use git status!

=> Commit ID generated: 27ff4

The graph: Commit 2



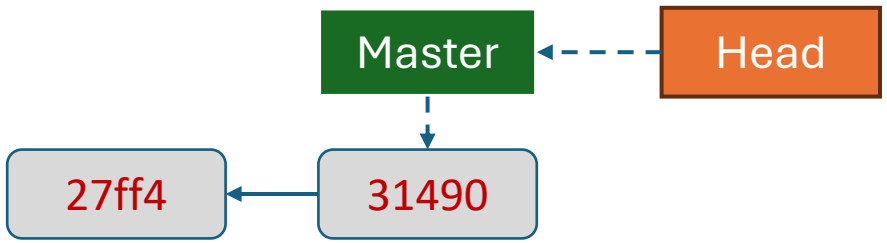
(a) Before 2nd commit

In a terminal...

```
echo banana >> fruits.txt
git add fruits.txt
git commit -m "Added banana to fruits.txt"
```

=> Commit ID generated: 31490

The graph: Commit 2



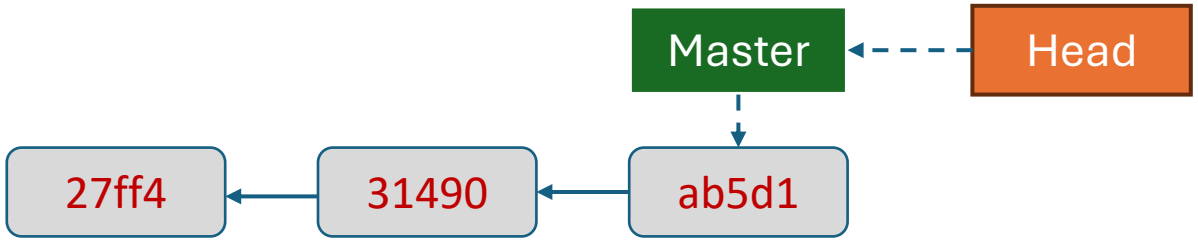
(a) 2nd commit

In a terminal...

```
echo banana >> fruits.txt
git add fruits.txt
git commit -m "Added banana to fruits.txt "
```

=> Commit ID generated: 31490

The graph: Commit 2



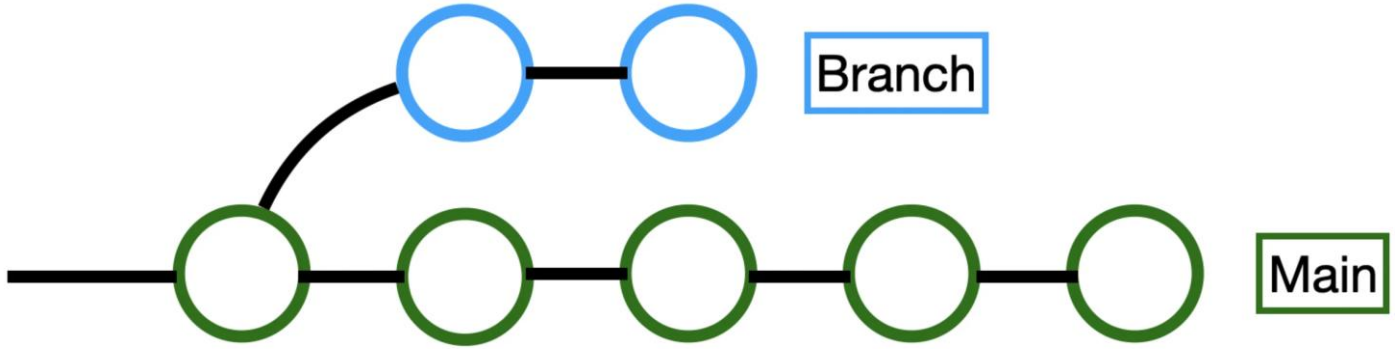
(a) Third commit

In a terminal...

```
echo orange >> fruits.txt
git add fruits.txt
git commit -m "Added orange to fruits.txt"
```

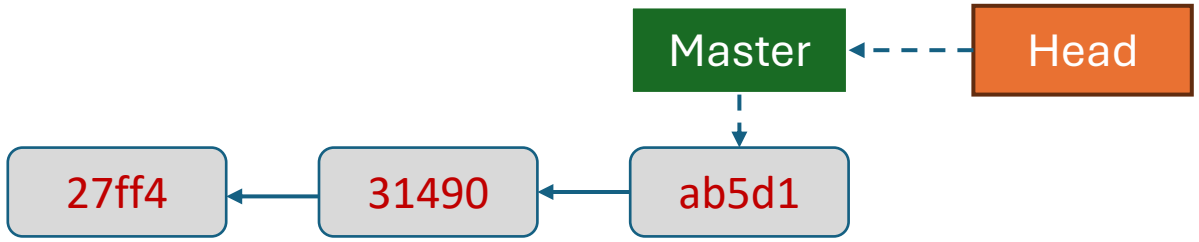
=> Commit ID generated: ab5d1

What is a Branch?



- A **branch** is a separate line of development in Git.
- It lets you work on features or fixes **independently** without affecting the main project.
- Branches are lightweight pointers to commits.
- You can switch between branches to work on different tasks.

The graph: Vegetables branch



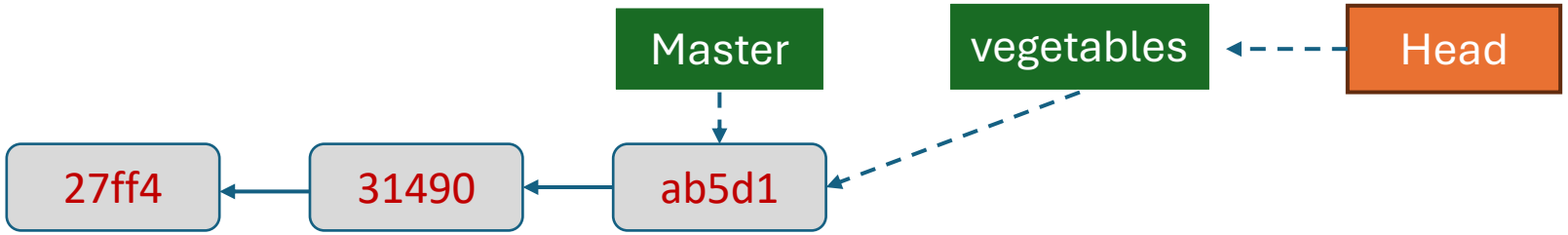
(a) Before branch

In a terminal...

git branch vegetables

git checkout vegetables

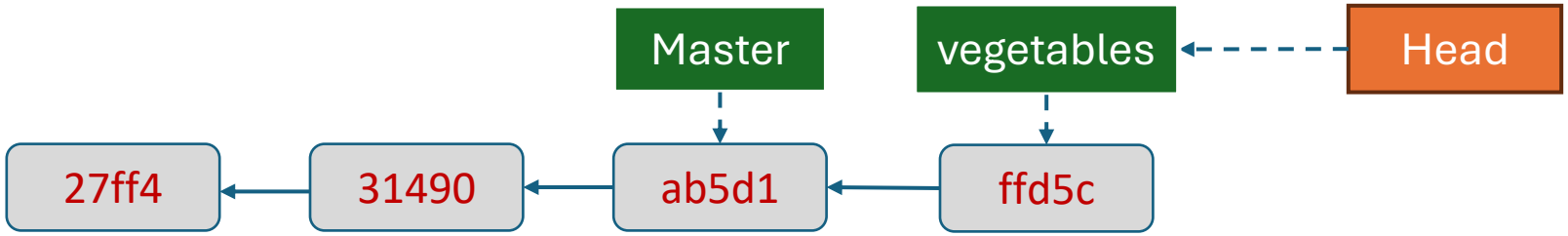
The graph: Vegetables branch



(a) After branch

A new label (branch) named **vegetables** appears.
This label points to the current commit (e.g., ab5d1).
The git checkout vegetables command moves HEAD to point to the **vegetables** branch.

The graph: Vegetables branch



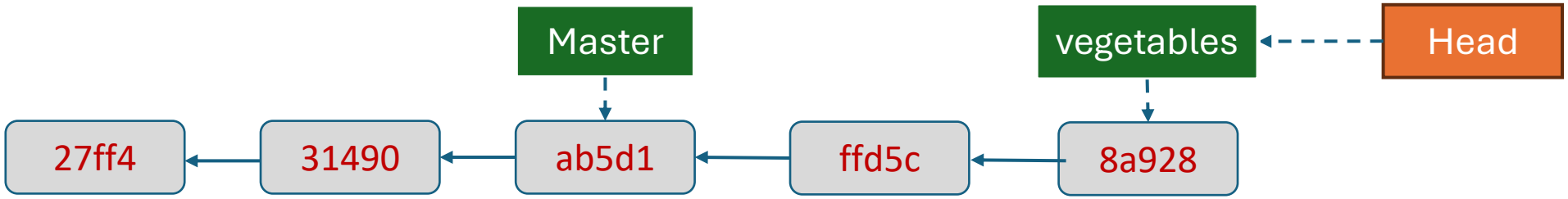
(a) After first commit in vegetables branch

In a terminal...

```
echo eggplant >> vegetables.txt
git add vegetables.txt
git commit -m "Add eggplant to vegetables"
```

=> Commit ID generated: ffd5c

The graph: Vegetables branch



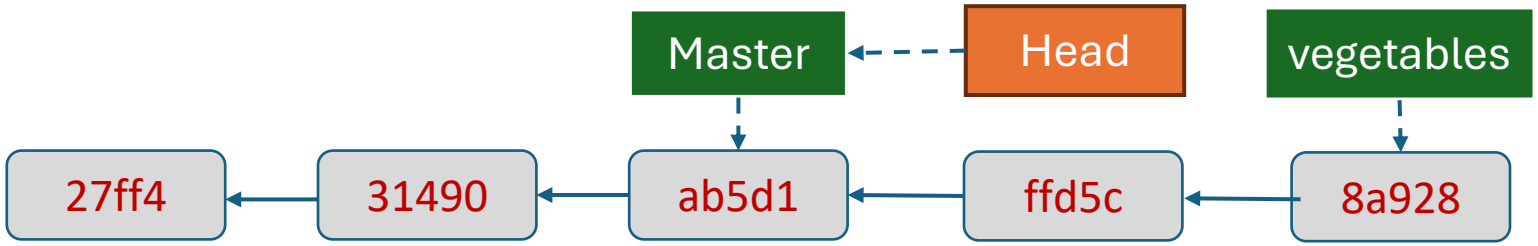
(a) After 2nd commit in vegetables branch

In a terminal...

```
echo zucchini >> vegetables.txt
git add vegetables.txt
git commit -m "Add zucchini to vegetables"
```

=> Commit ID generated: 8a928

The graph: Master branch

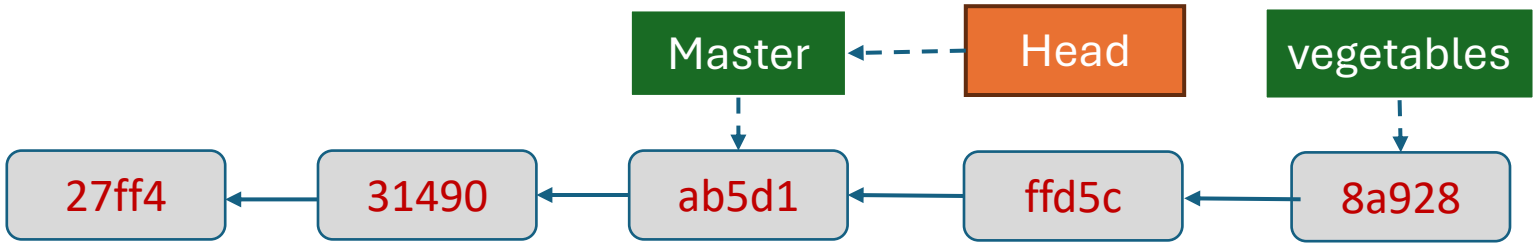


(a) Lets move to the Master branch

In a terminal...

```
git checkout master
```

The graph: Master branch



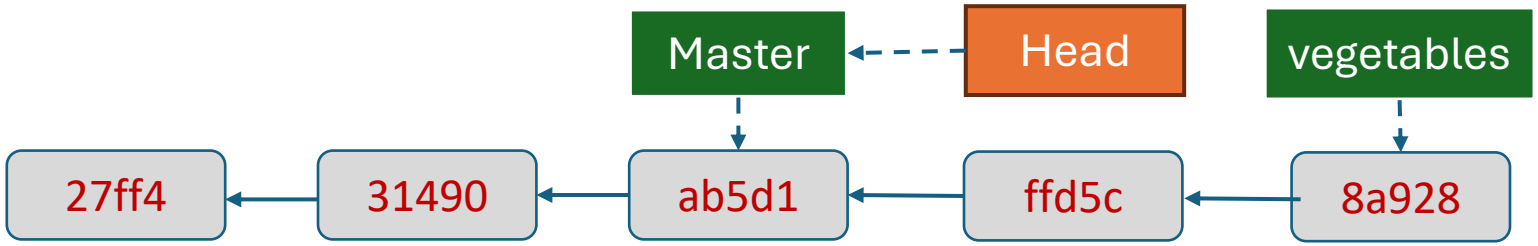
(a) Lets move to the Master branch

In a terminal...

```
git checkout master
```

The file vegetables.txt no longer exists in the Working Directory.

The graph: Master branch



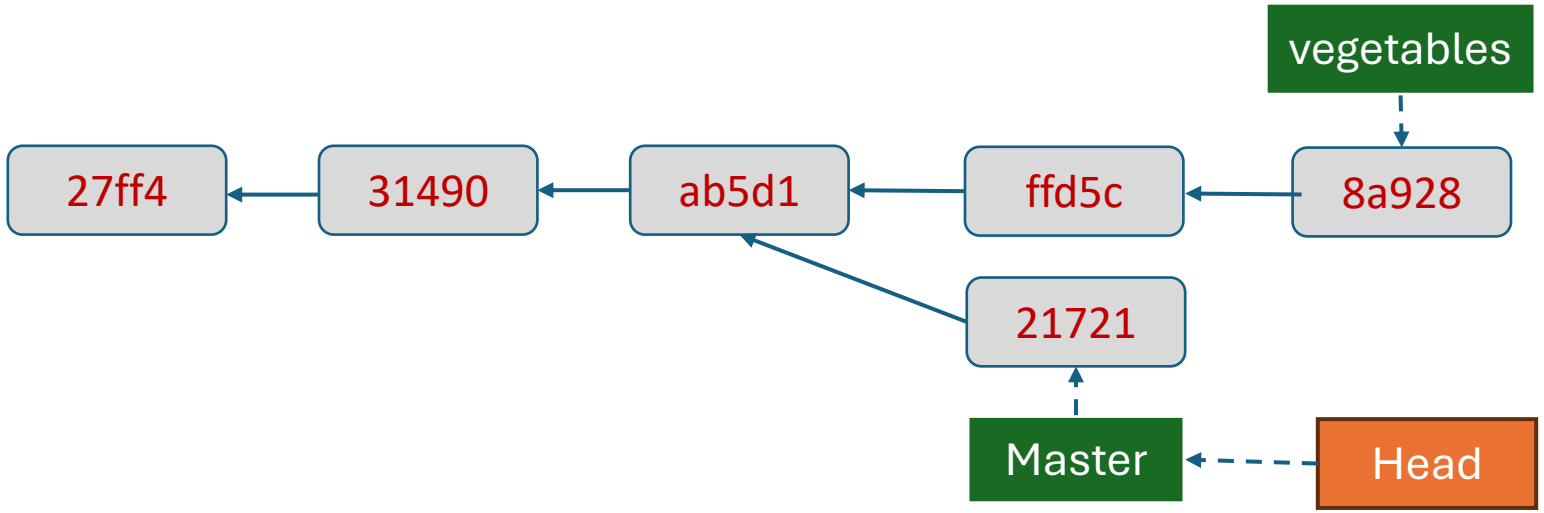
(a) Commit Changes on the Master Branch

In a terminal...

```
echo pear >> fruits.txt
git add fruits.txt
git commit -m "Add pear to fruits.txt“
```

=> Commit ID generated: 21721

The graph: Master branch



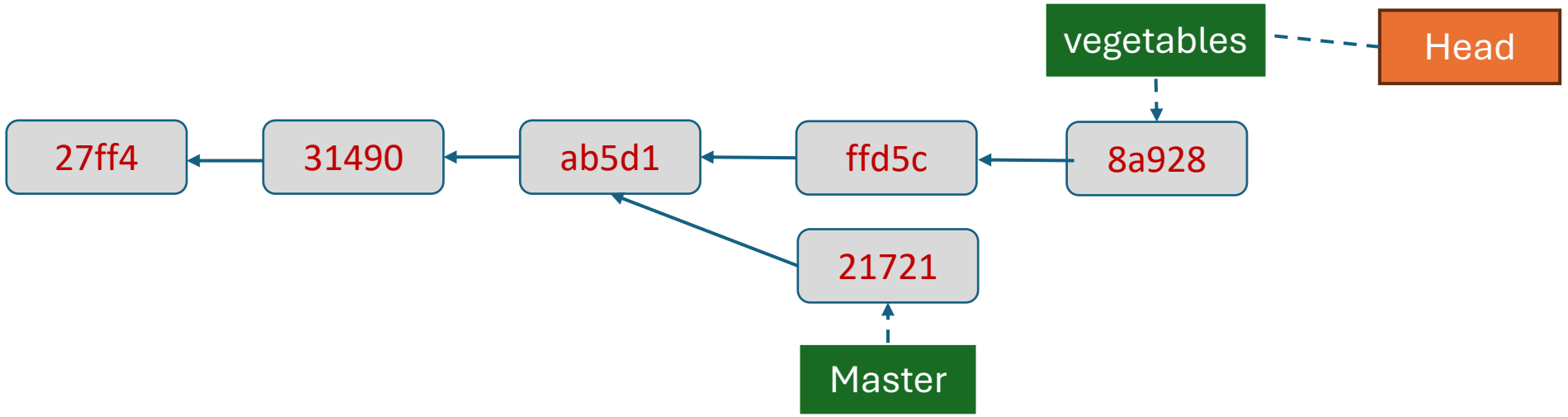
(a) After Commit Changes on the Master Branch

In a terminal...

```
echo pear >> fruits.txt
git add fruits.txt
git commit -m "Add pear to fruits.txt“
```

=> Commit ID generated: 21721

The Graph: Merging master and vegetables

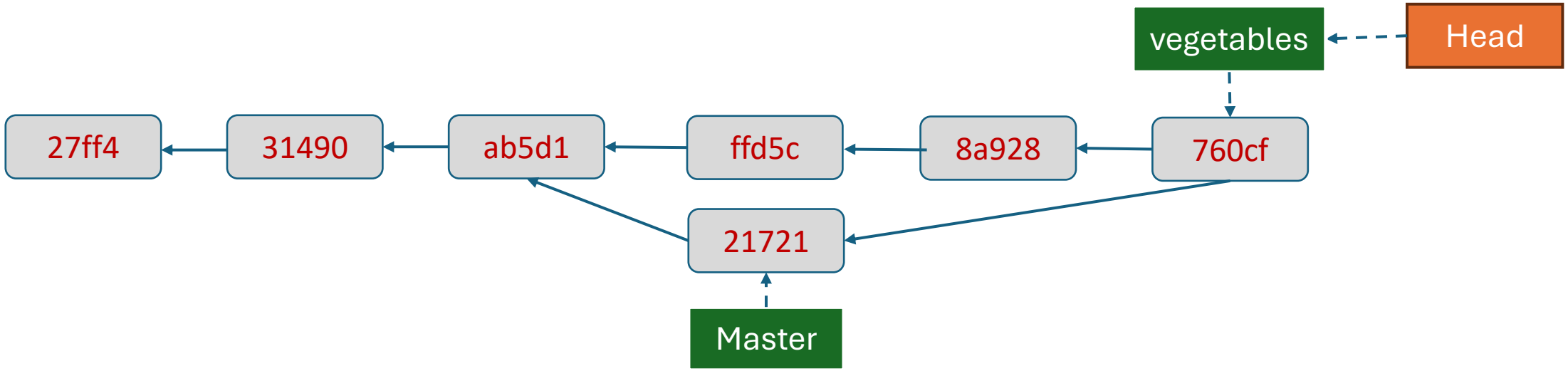


(a) Checking Differences with diff

In a terminal...

```
git checkout vegetables
git diff master
```

The Graph: Merging master and vegetables



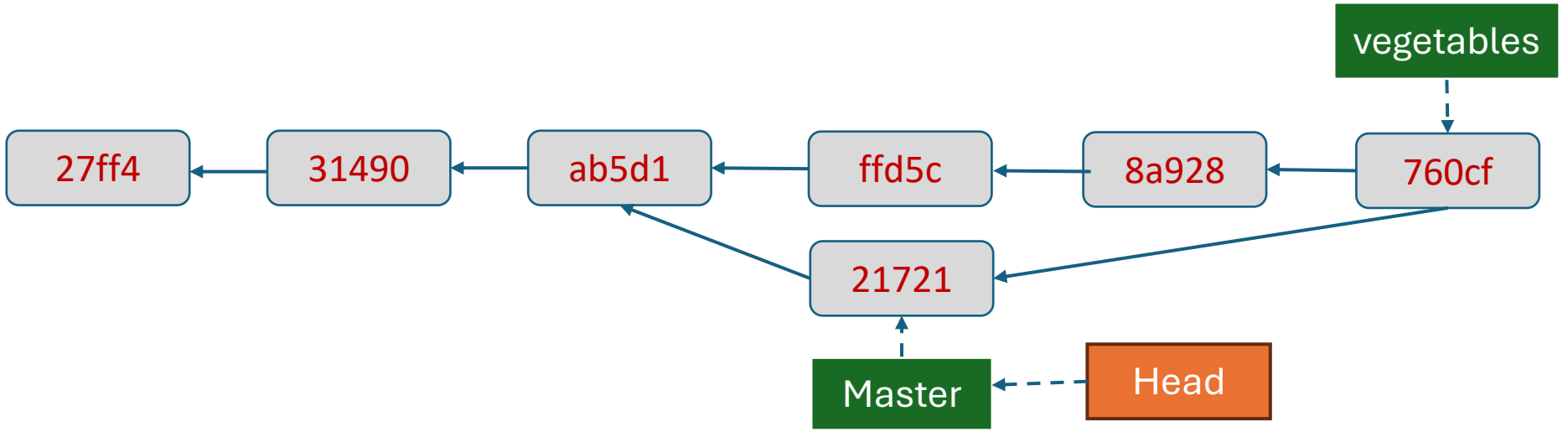
(a) Merge vegetables and Master

In a terminal...

```
git checkout vegetables
git diff master
git merge master
```

Merging master into vegetables Creates a New Commit (760cf)

The Graph: Merging master and vegetables

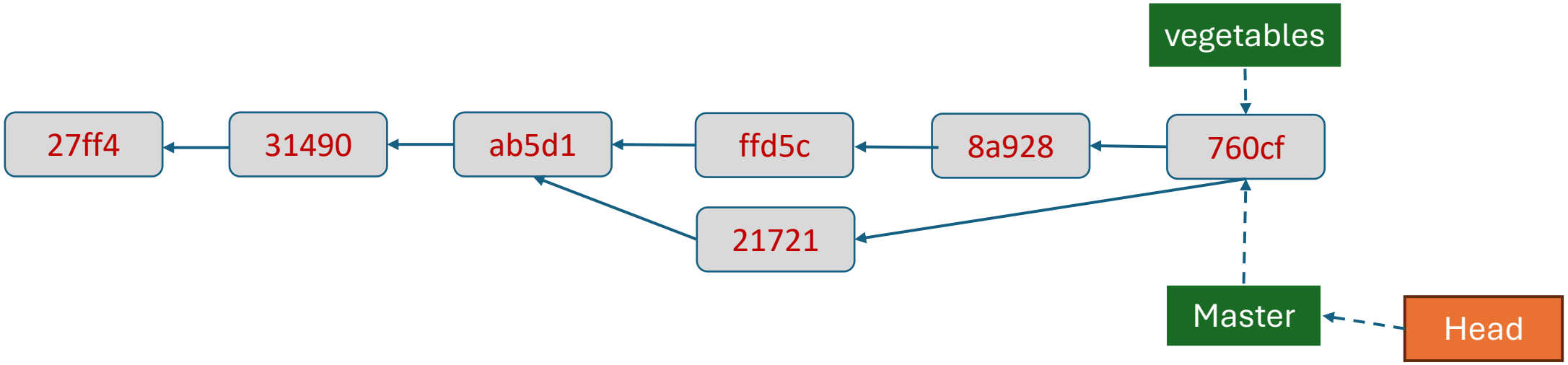


(a) Check Master branch

In a terminal...

```
git checkout master
git diff vegetables
```

The Graph: Merging master and vegetables



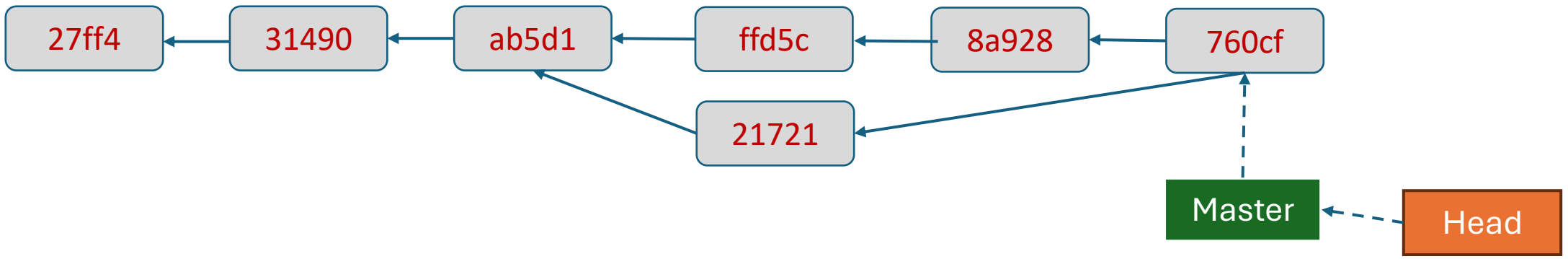
(a) Merge Master and vegetables

In a terminal...

```
git checkout master
git diff vegetables
git merge master
```

No New Commit Created

The Graph: Merging master and vegetables



(a) Merge Master and vegetables

In a terminal...

```
git checkout master
git diff vegetables
git merge master
git branch –d vegetables
```

Remove vegetables