The Why and How of git

"The Stupid Content Tracker"

J. Rötter[1], J. Rosenzweig[1]

[1]University of Florida

2021-04-02

What you will learn:

- The **what, why, when, and how** of Git

What you will learn:

- The **what, why, when, and how** of Git
- *<u>What</u> is Git?*

What you will learn:

- The **what, why, when, and how** of Git

- *<u>What</u> is Git?*
  - According to Cambridge English Dictionary:

git

noun [C] UK informal

UK /gɪt/ US /gɪt/

a person, especially a man, who is stupid or unpleasant:

- *You stupid/lying git!*
- *He's a miserable old git.*

What you will learn:

- The **what, why, when, and how** of Git

- *<u>What</u> is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

git

noun [C] UK informal

UK /gɪt/ US /gɪt/

a person, especially a man, who is stupid or unpleasant:

- *You stupid/lying git!*
- *He's a miserable old git.*

What you will learn:

- The **what, why, when, and how** of Git

- *<u>What</u> is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

- *<u>Why</u> should you use Git?*

git

noun [C] UK informal

UK /gɪt/ US /gɪt/

a person, especially a man, who is stupid or unpleasant:

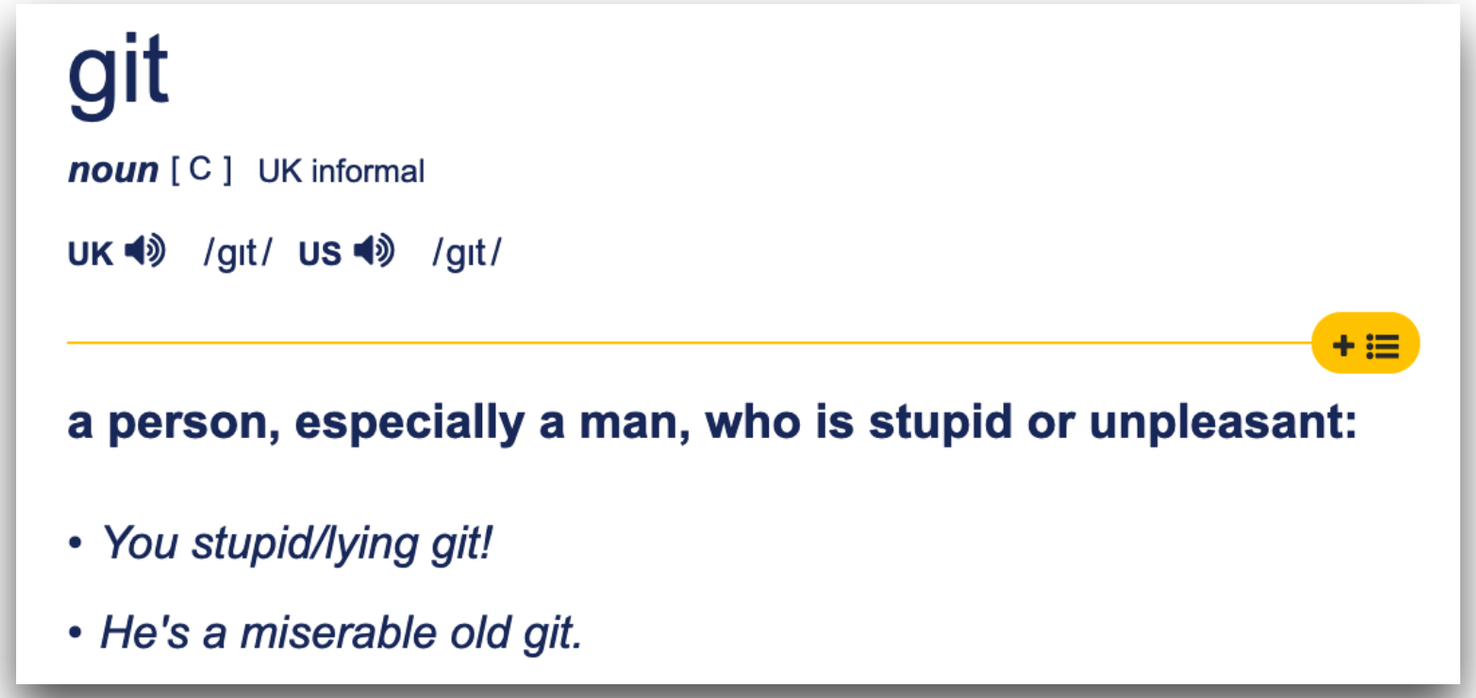
- *You stupid/lying git!*
- *He's a miserable old git.*

What you will learn:

- The **what, why, when, and how** of Git

- *<u>What</u> is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

- *<u>Why</u> should you use Git?*
  - **Save** code
  - **Share** code.
  - Modify ***other people's*** code.
  - **Revert** back to previous (working) code!
  - Public code is your **code "CV"**.
  - (You can use Git with private code)



What you will learn:

- The **what, why, when, and how** of Git

- *What is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

- *Why should you use Git?*
  - **Save** code
  - **Share** code.
  - Modify ***other people's*** code.
  - **Revert** back to previous (working) code!
  - Public code is your **code "CV"**.
  - (You can use Git with private code)

- *When should you use Git?*

git

noun [C] UK informal

UK /gɪt/ US /gɪt/

a person, especially a man, who is stupid or unpleasant:

- *You stupid/lying git!*
- *He's a miserable old git.*

What you will learn:

- The **what, why, when, and how** of Git

- *What is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

- *Why should you use Git?*
  - **Save** code
  - **Share** code.
  - Modify ***other people's*** code.
  - **Revert** back to previous (working) code!
  - Public code is your **code "CV"**.
  - (You can use Git with private code)

- *When should you use Git?*
  - **Every time you code.**

git

noun [C] UK informal

UK /gɪt/ US /gɪt/

a person, especially a man, who is stupid or unpleasant:

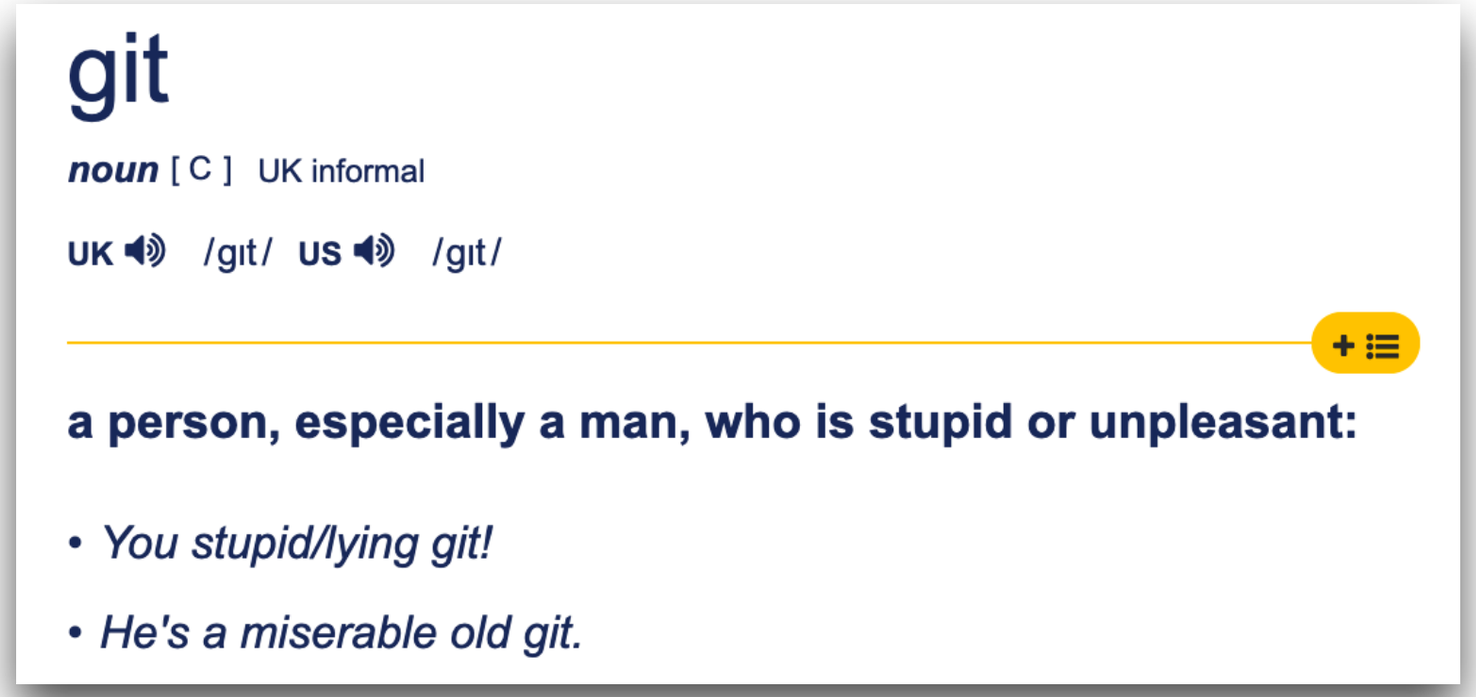
- *You stupid/lying git!*
- *He's a miserable old git.*

What you will learn:

- The **what, why, when, and how** of Git

- *<u>What</u> is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

- *<u>Why</u> should you use Git?*
  - **Save** code
  - **Share** code.
  - Modify ***other people's*** code.
  - **Revert** back to previous (working) code!
  - Public code is your **code "CV"**.
  - (You can use Git with private code)



- *<u>When</u> should you use Git?*
  - **Every time you code.**

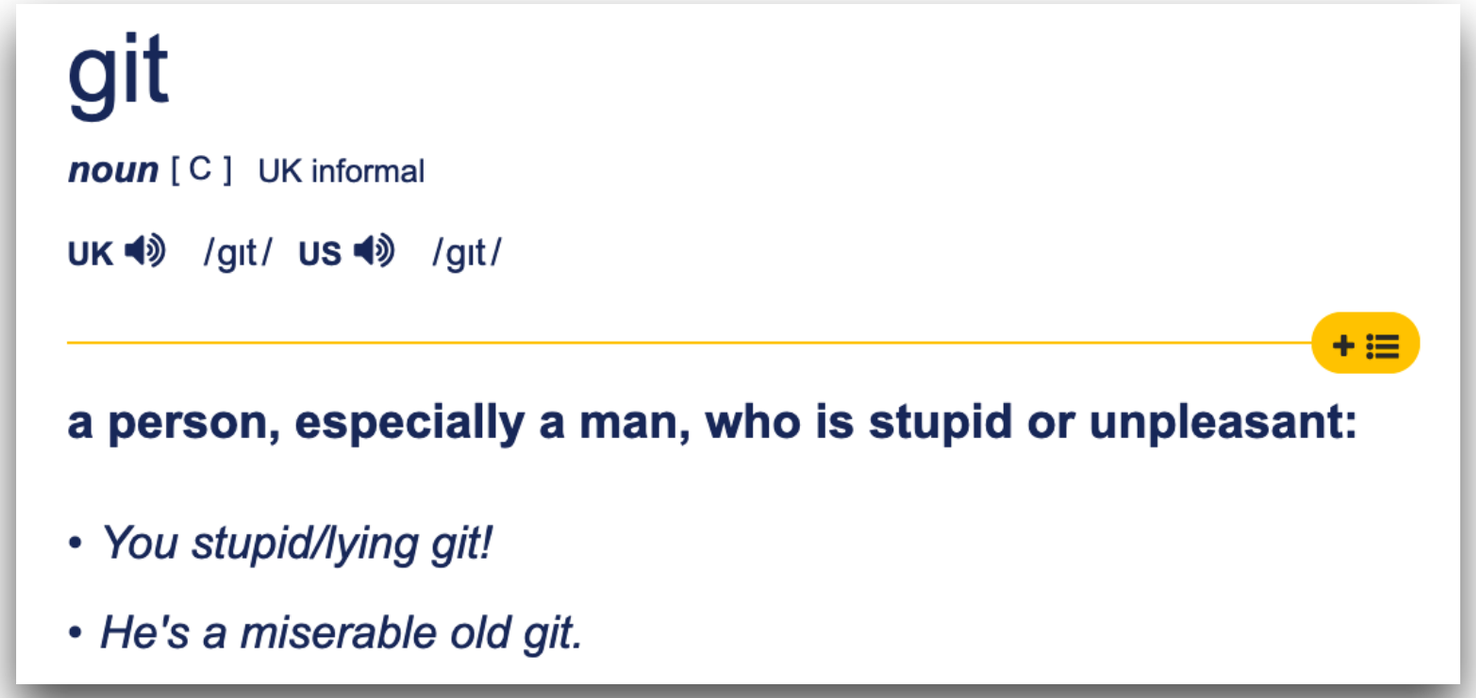
- *<u>How</u> do you use Git?*

What you will learn:

- The **what, why, when, and how** of Git

- *<u>What</u> is Git?*
  - According to Cambridge English Dictionary:
  - Version control software.

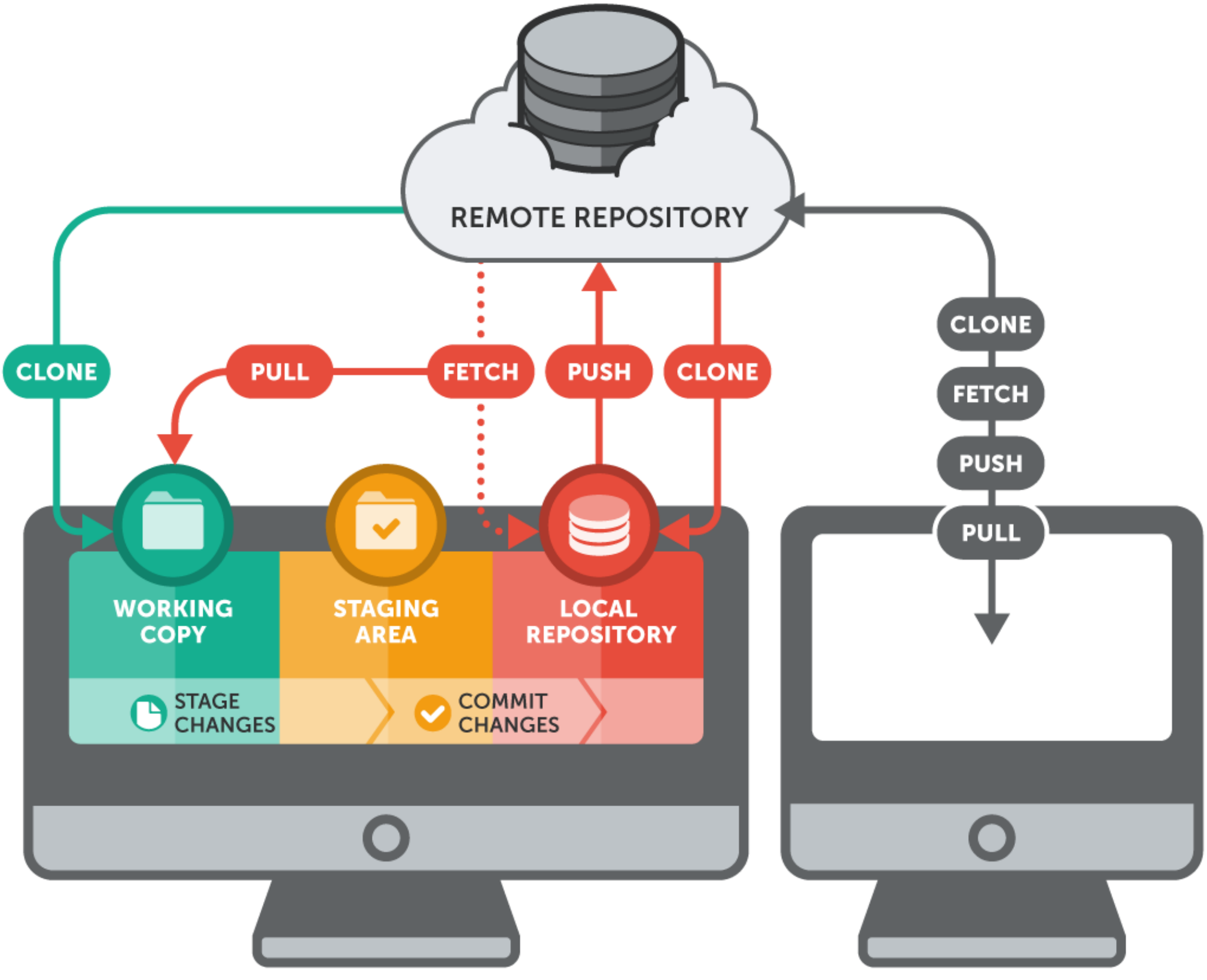
- *<u>Why</u> should you use Git?*
  - **Save** code
  - **Share** code.
  - Modify ***other people's*** code.
  - **Revert** back to previous (working) code!
  - Public code is your **code "CV"**.
  - (You can use Git with private code)



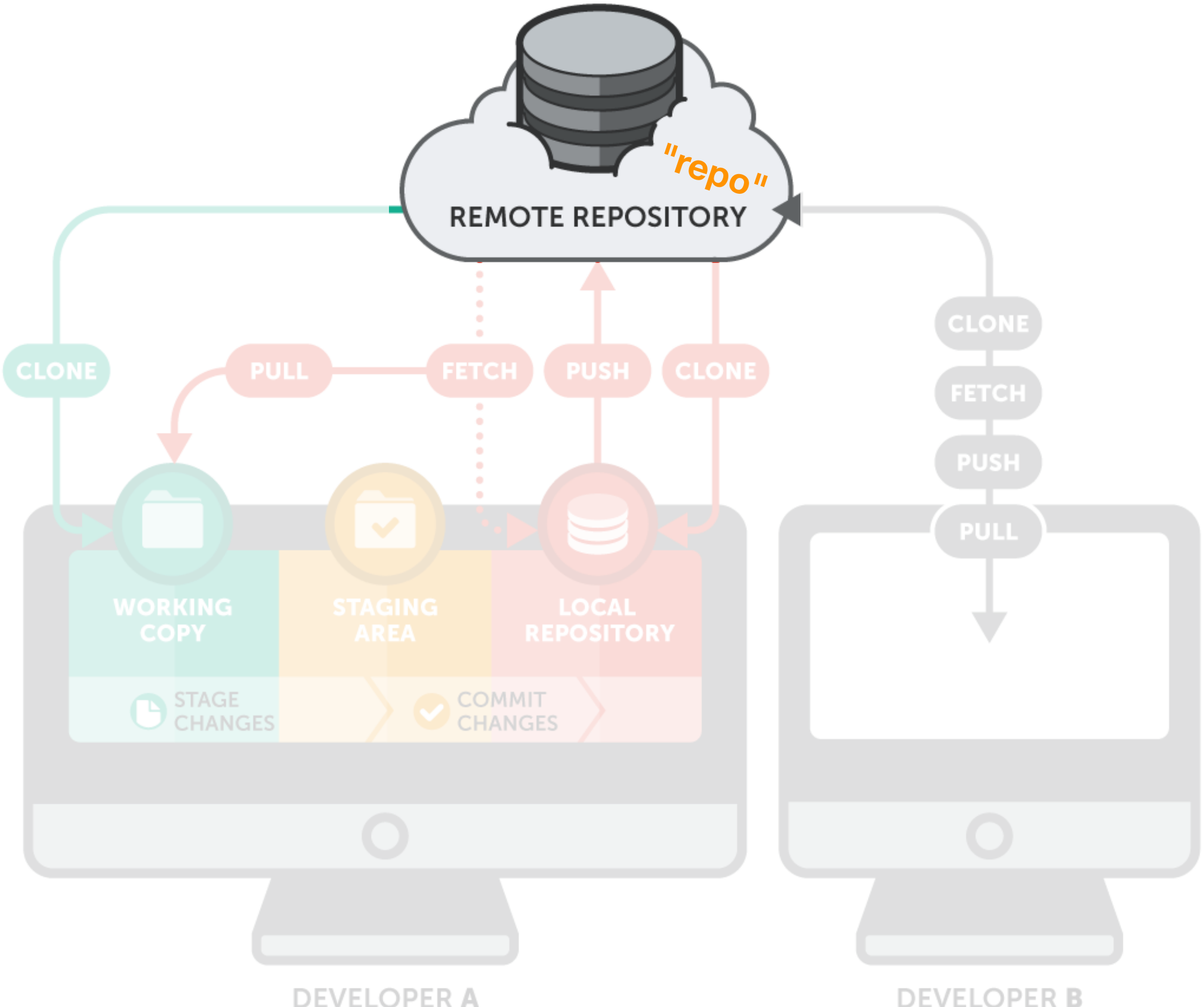
- *<u>When</u> should you use Git?*
  - **Every time you code.**

- *<u>How</u> do you use Git?*
  - *Step into my office...*

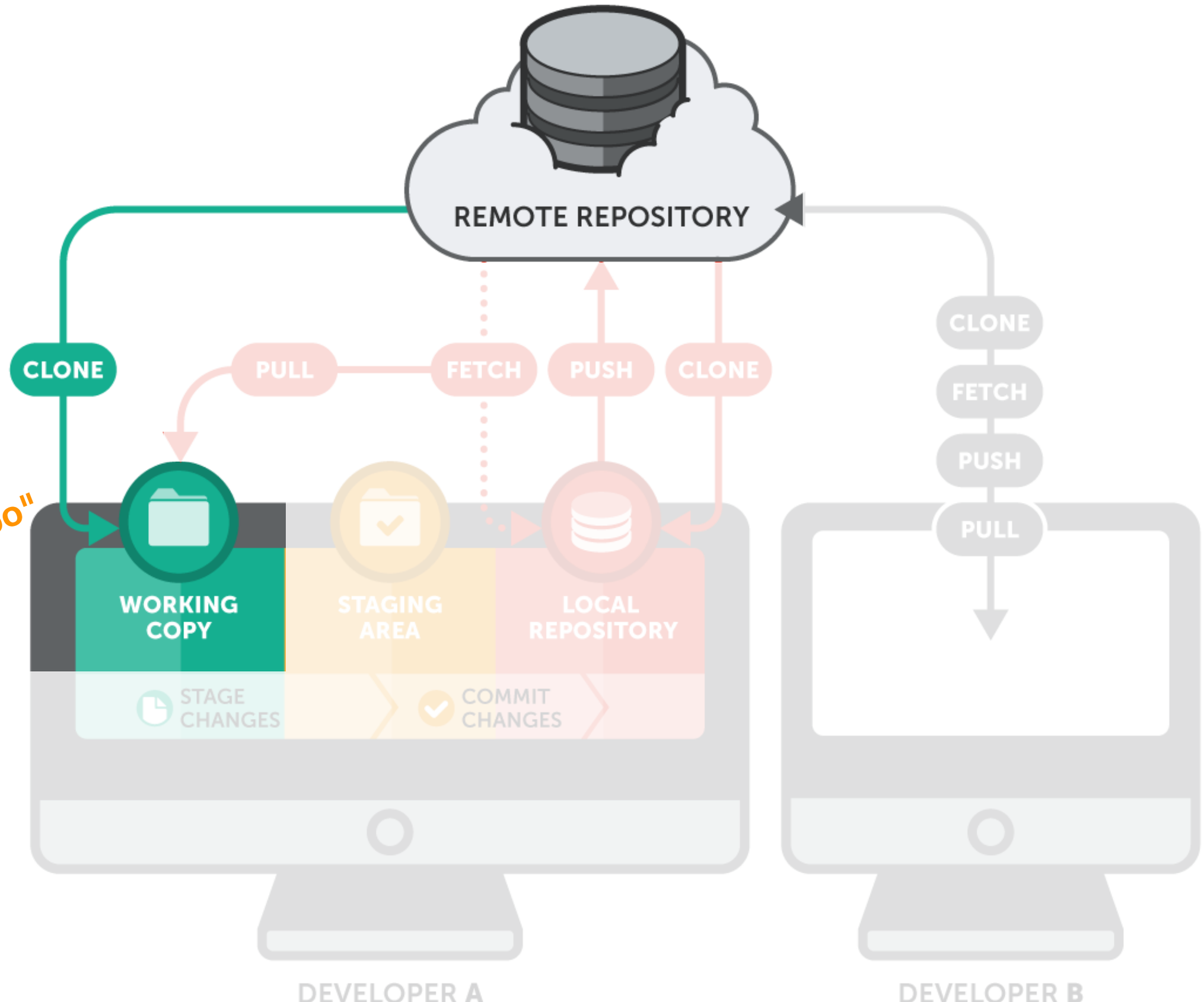
Git the Main Idea:



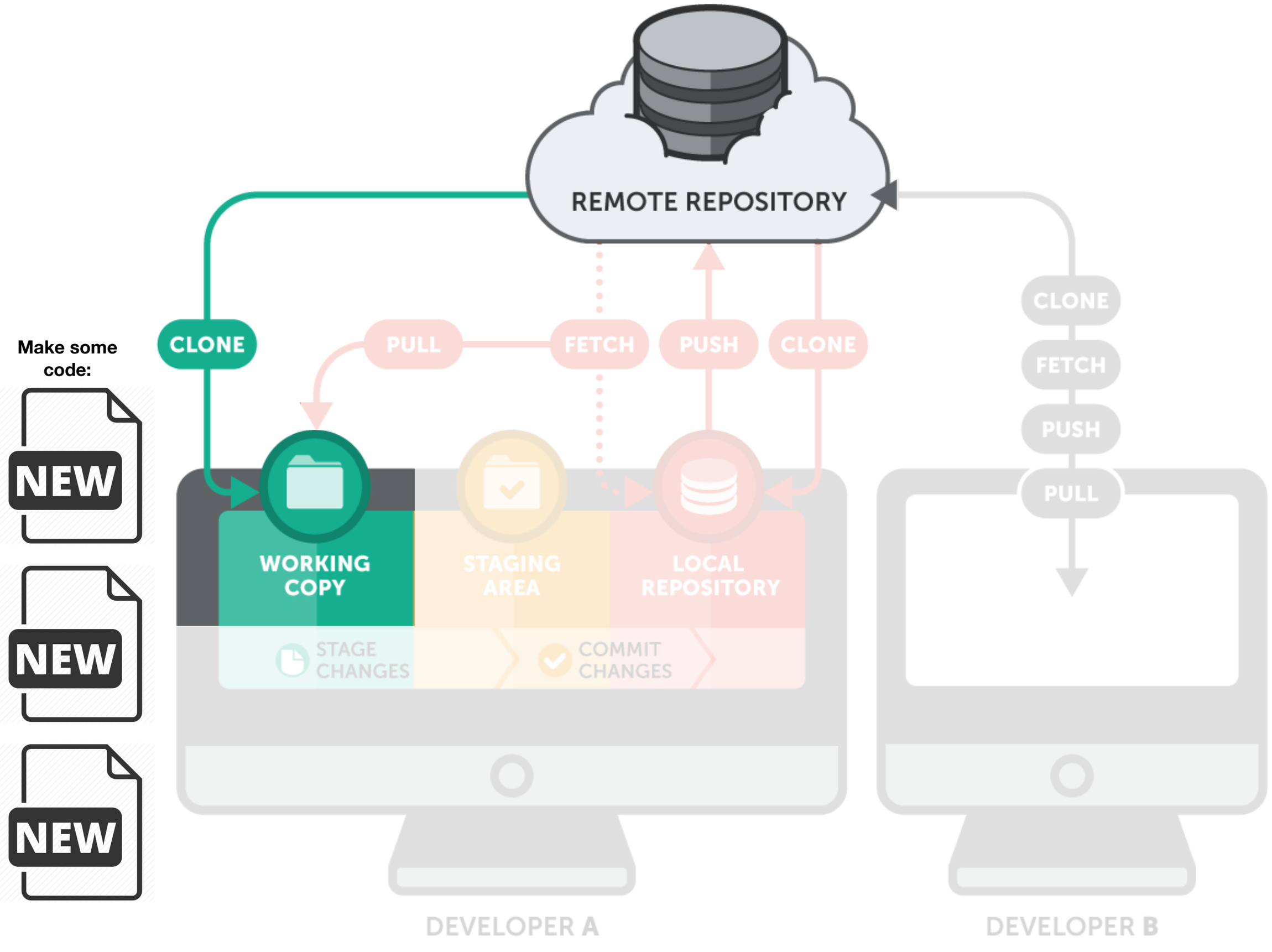
Git the Main Idea:



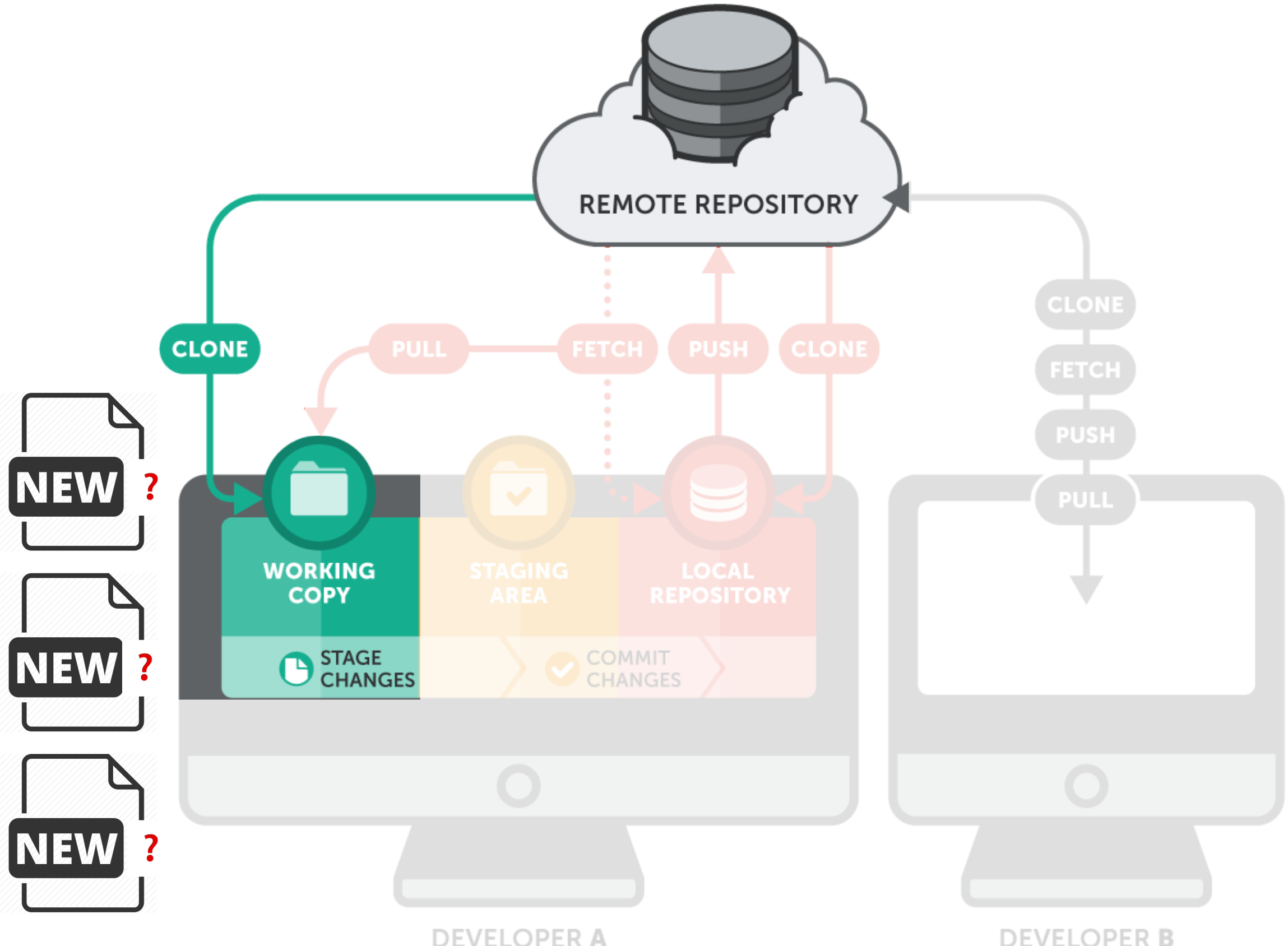
Git the Main Idea:



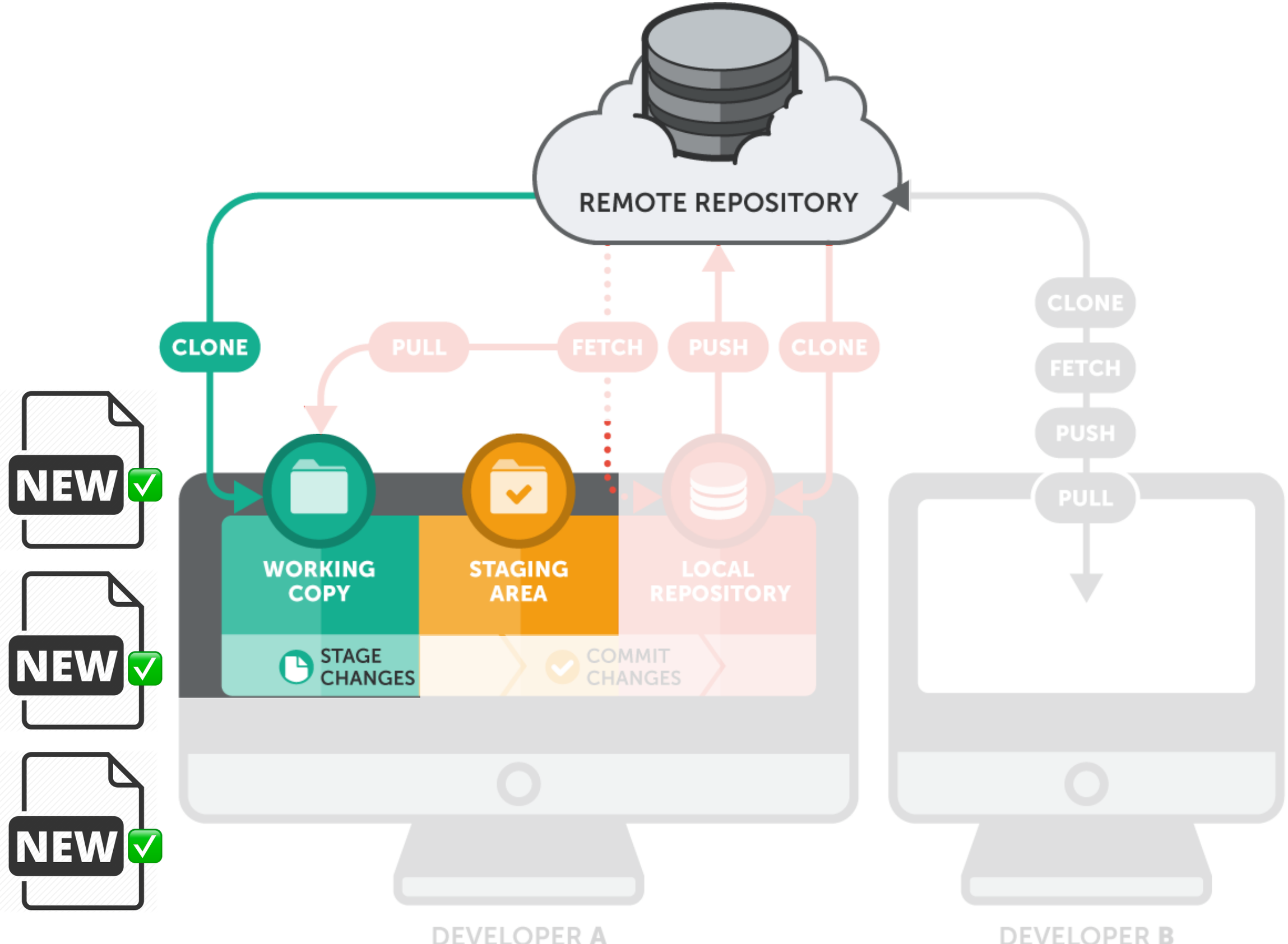
Git the Main Idea:



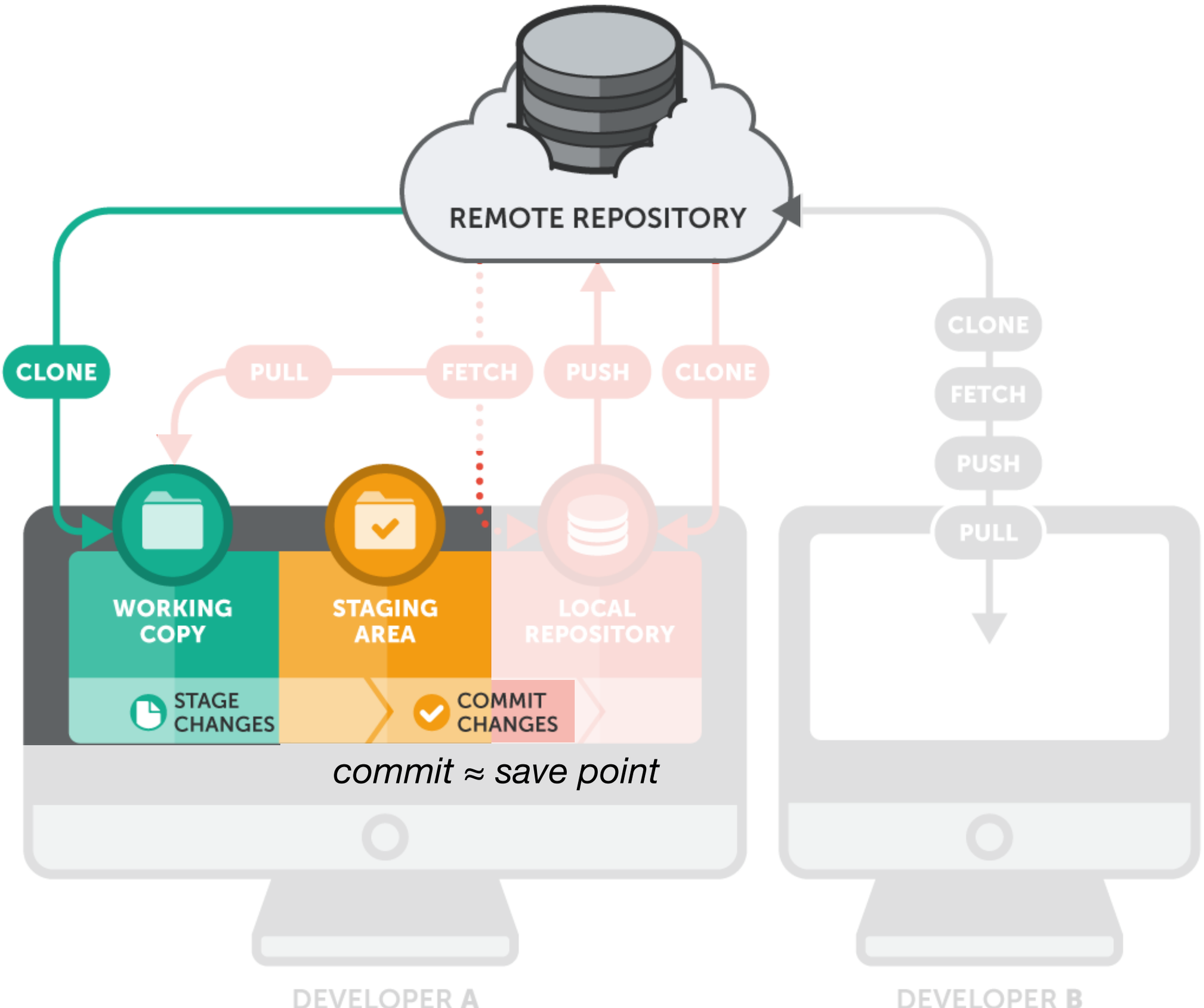
Git the Main Idea:



Git the Main Idea:

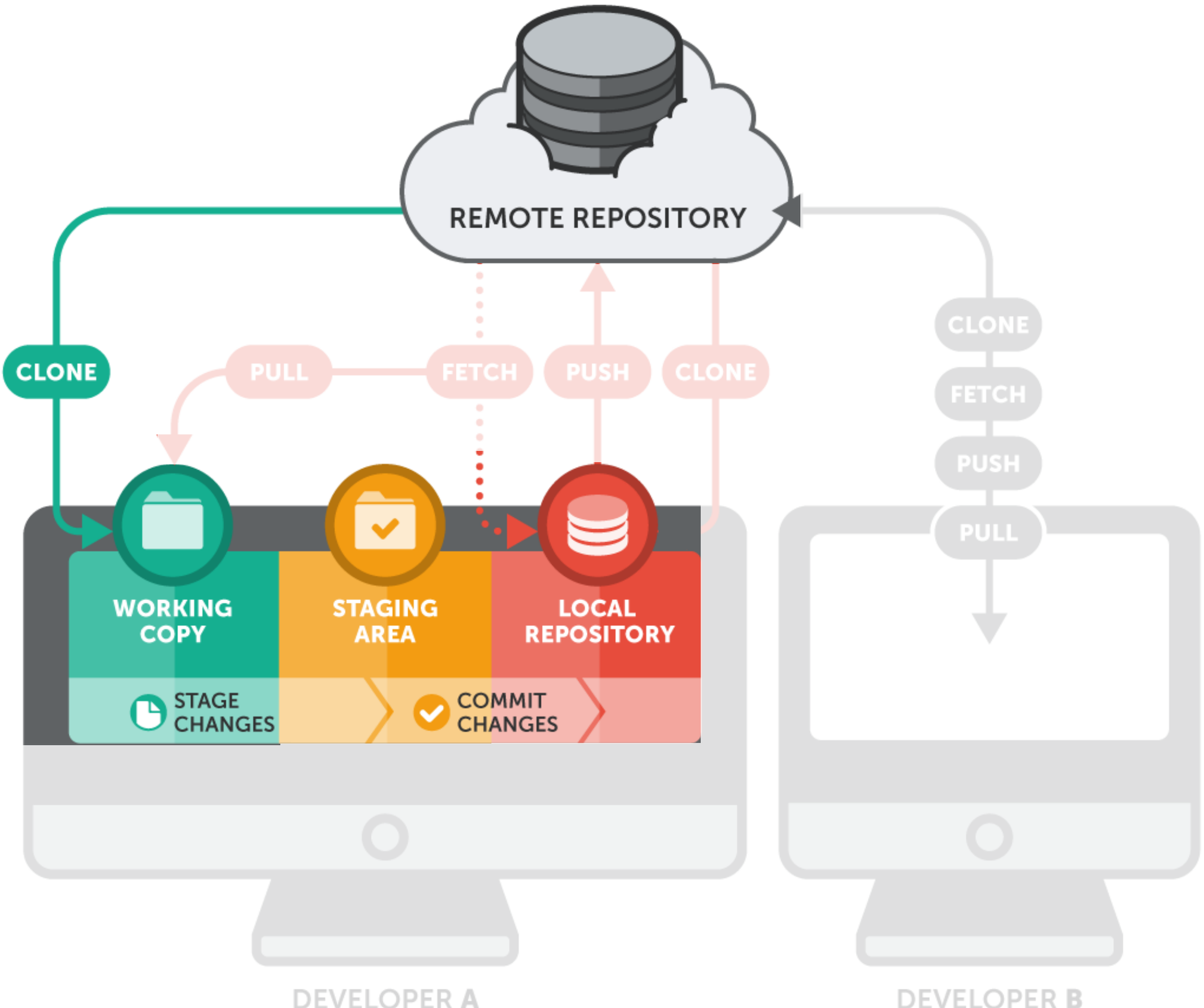


Git the Main Idea:

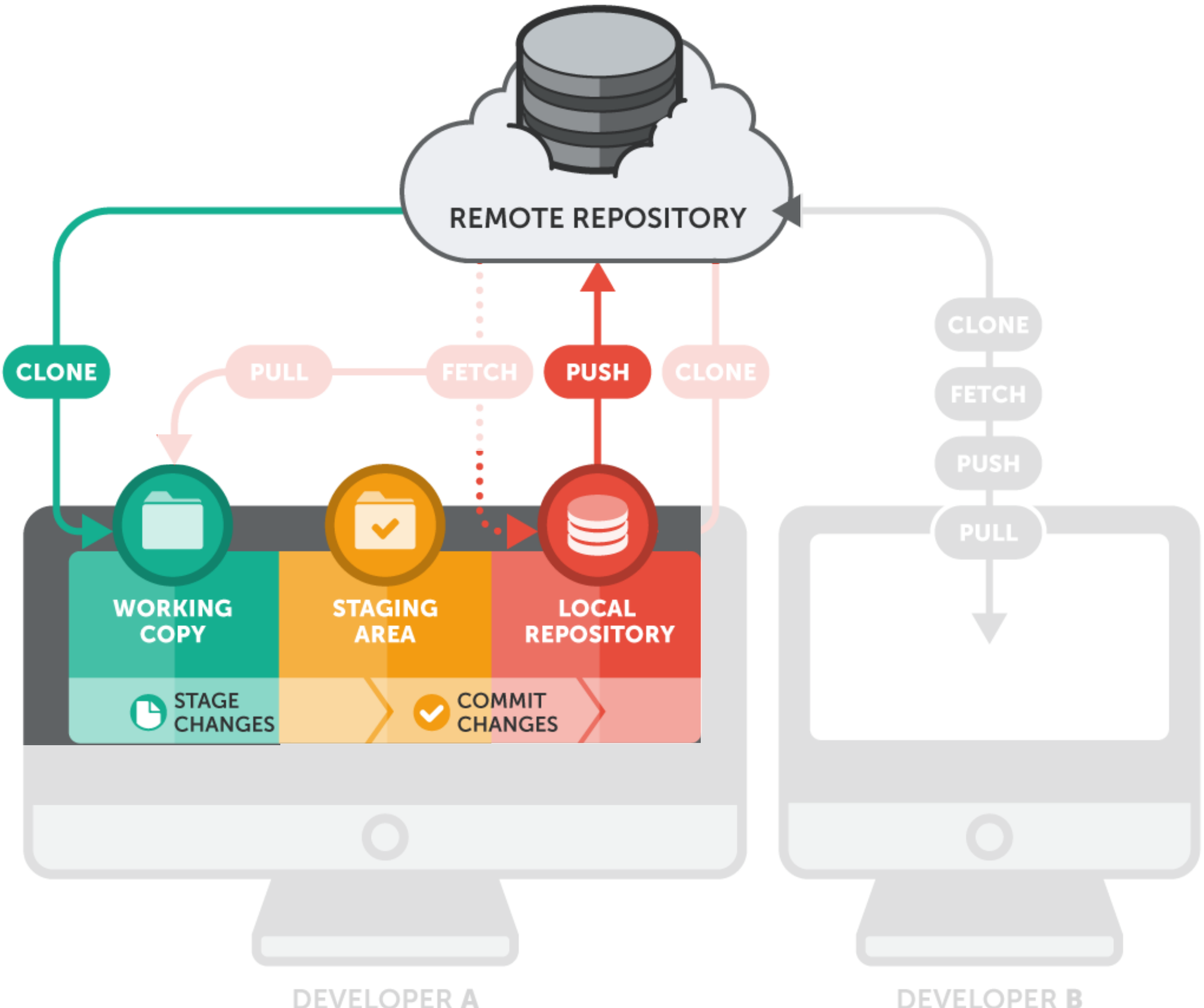


commit ≈ save point

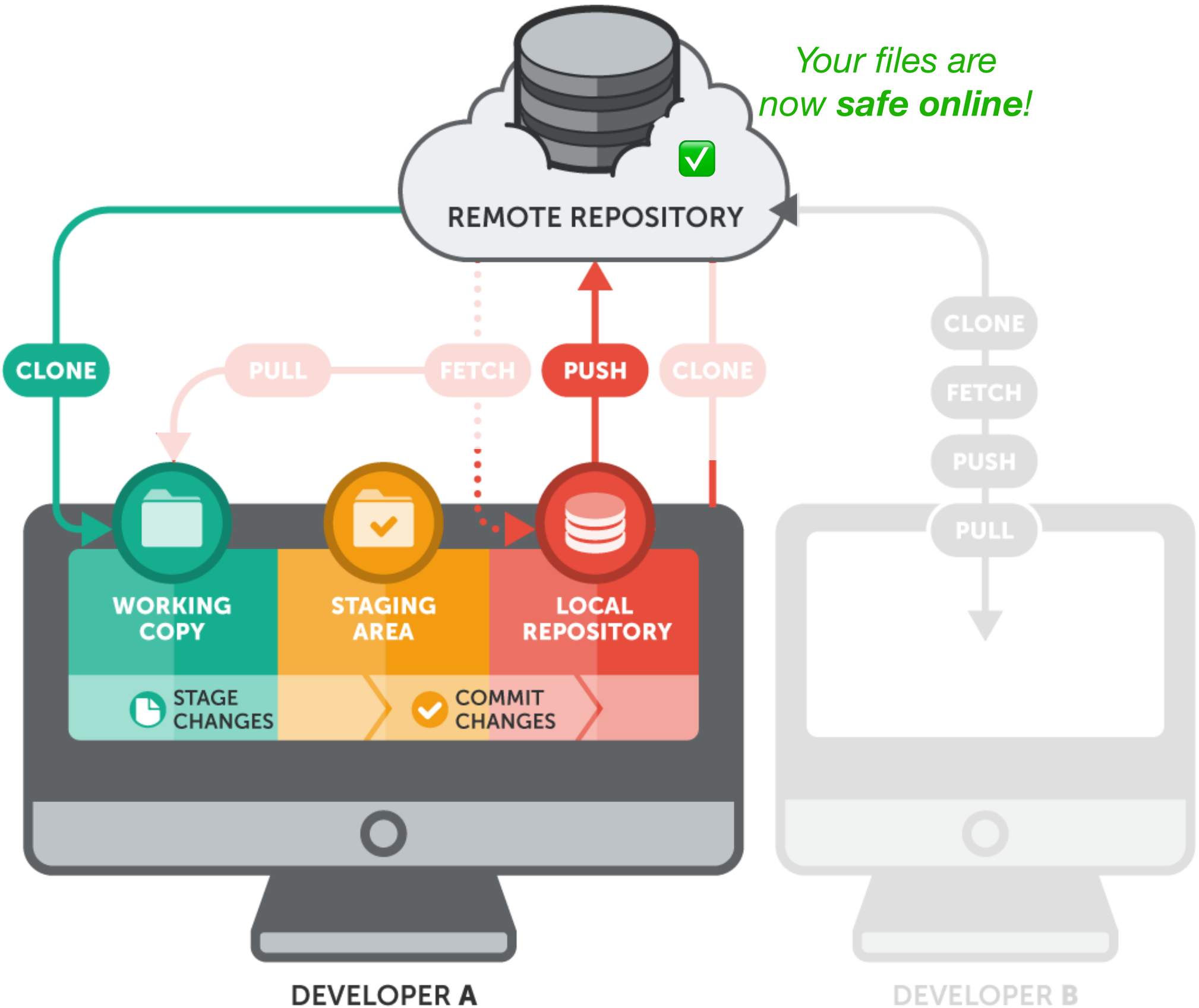
Git the Main Idea:



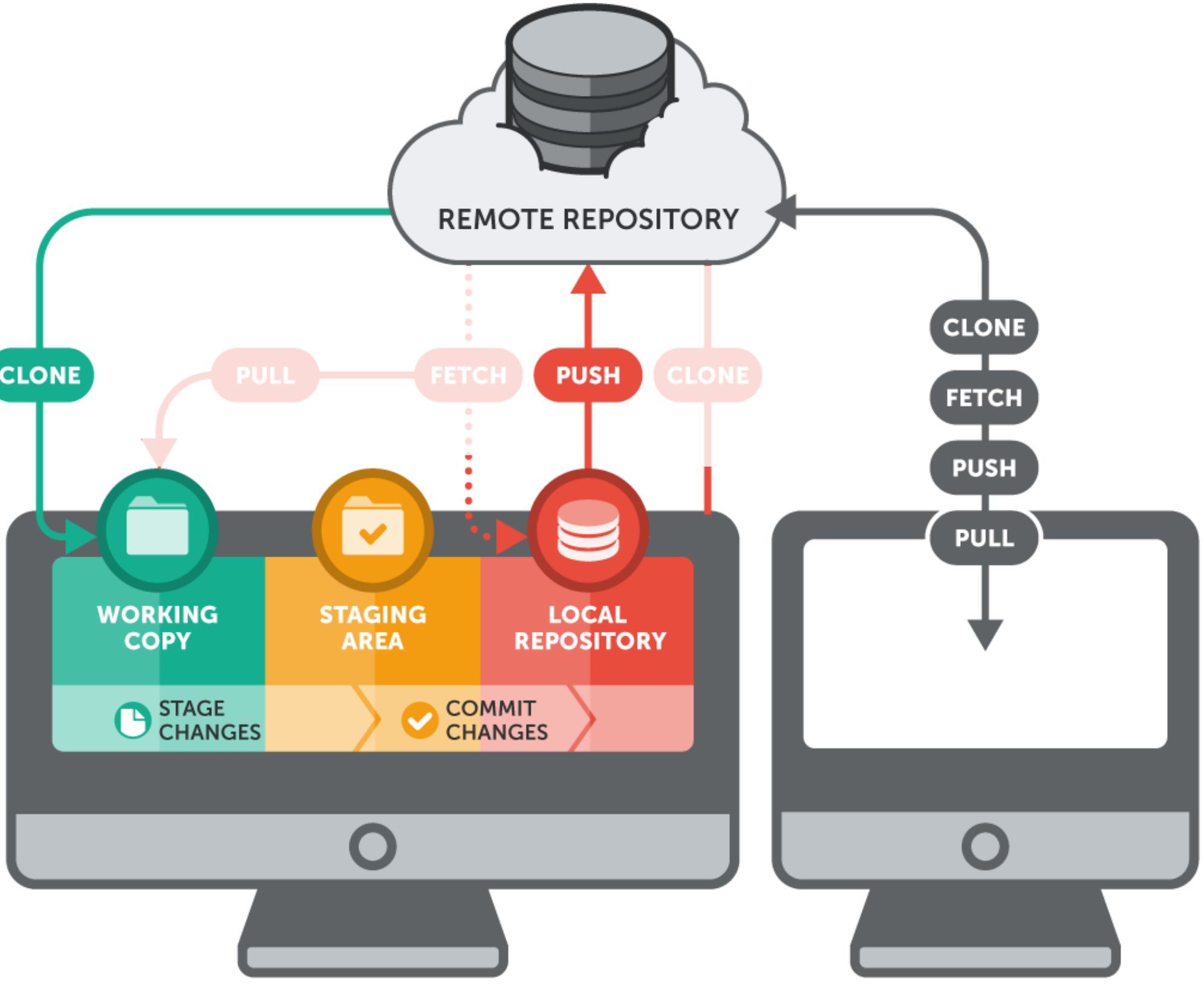
Git the Main Idea:



Git the Main Idea:

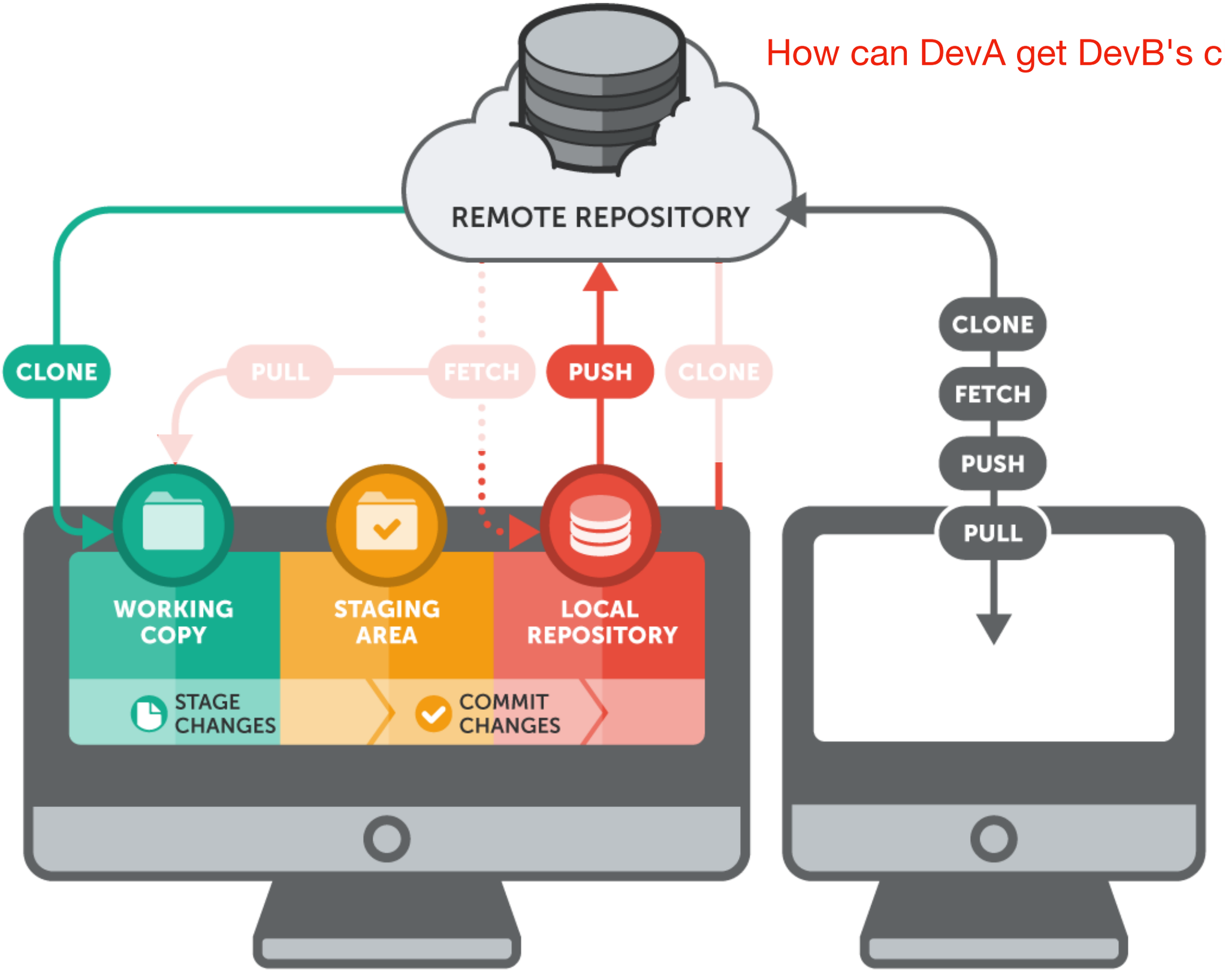


Git the Main Idea:



Git the Main Idea:

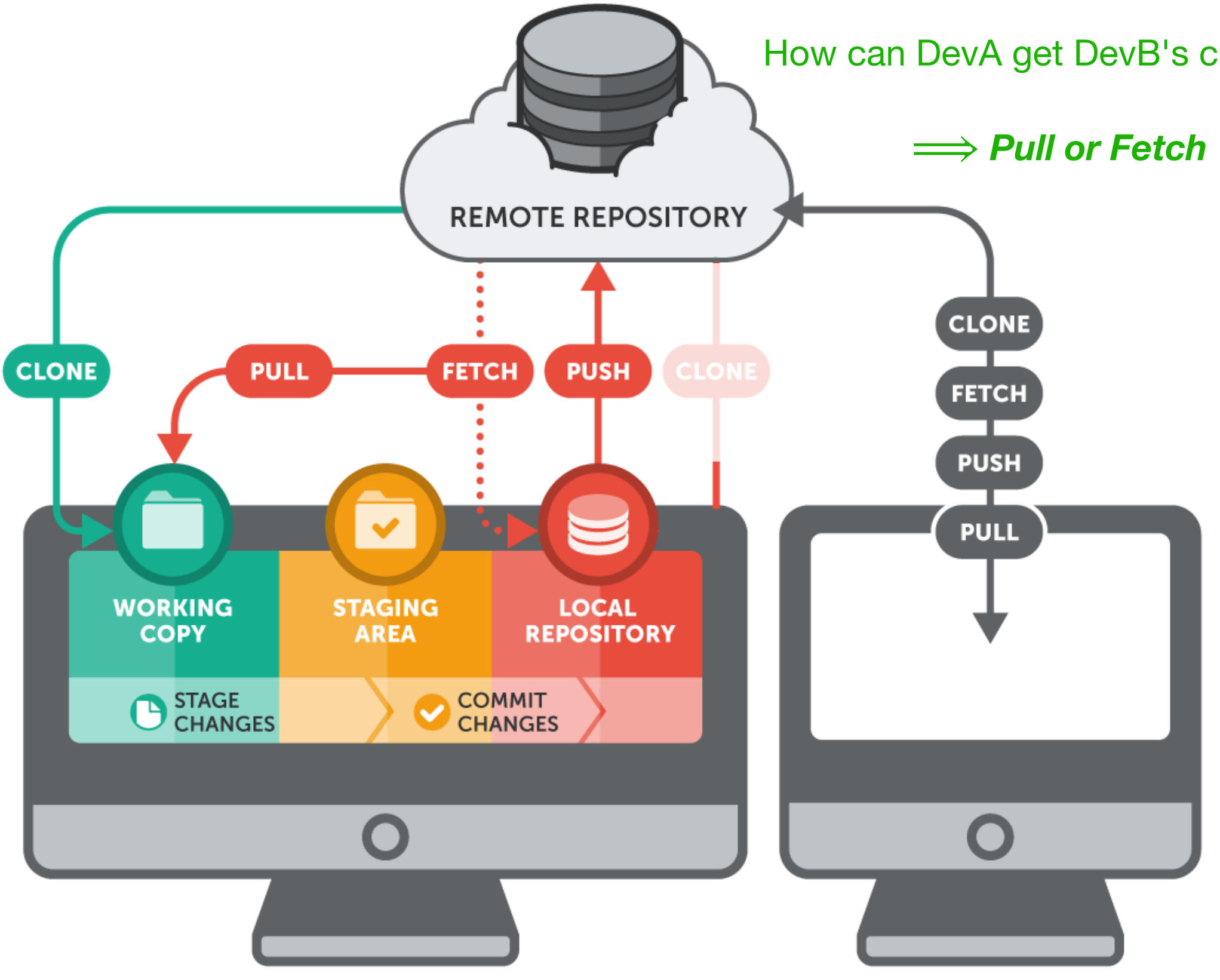
How can DevA get DevB's changes?



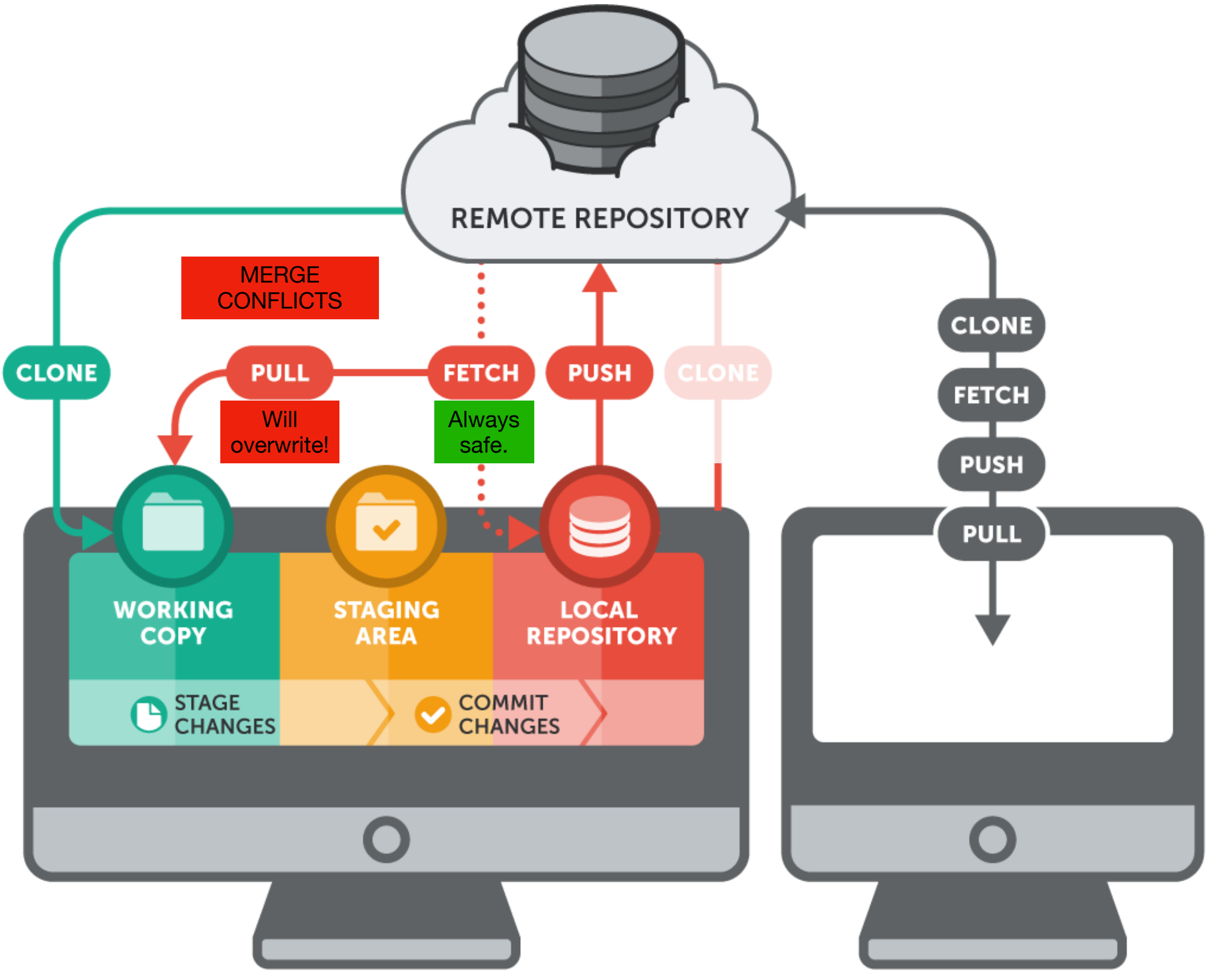
Git the Main Idea:

How can DevA get DevB's changes?

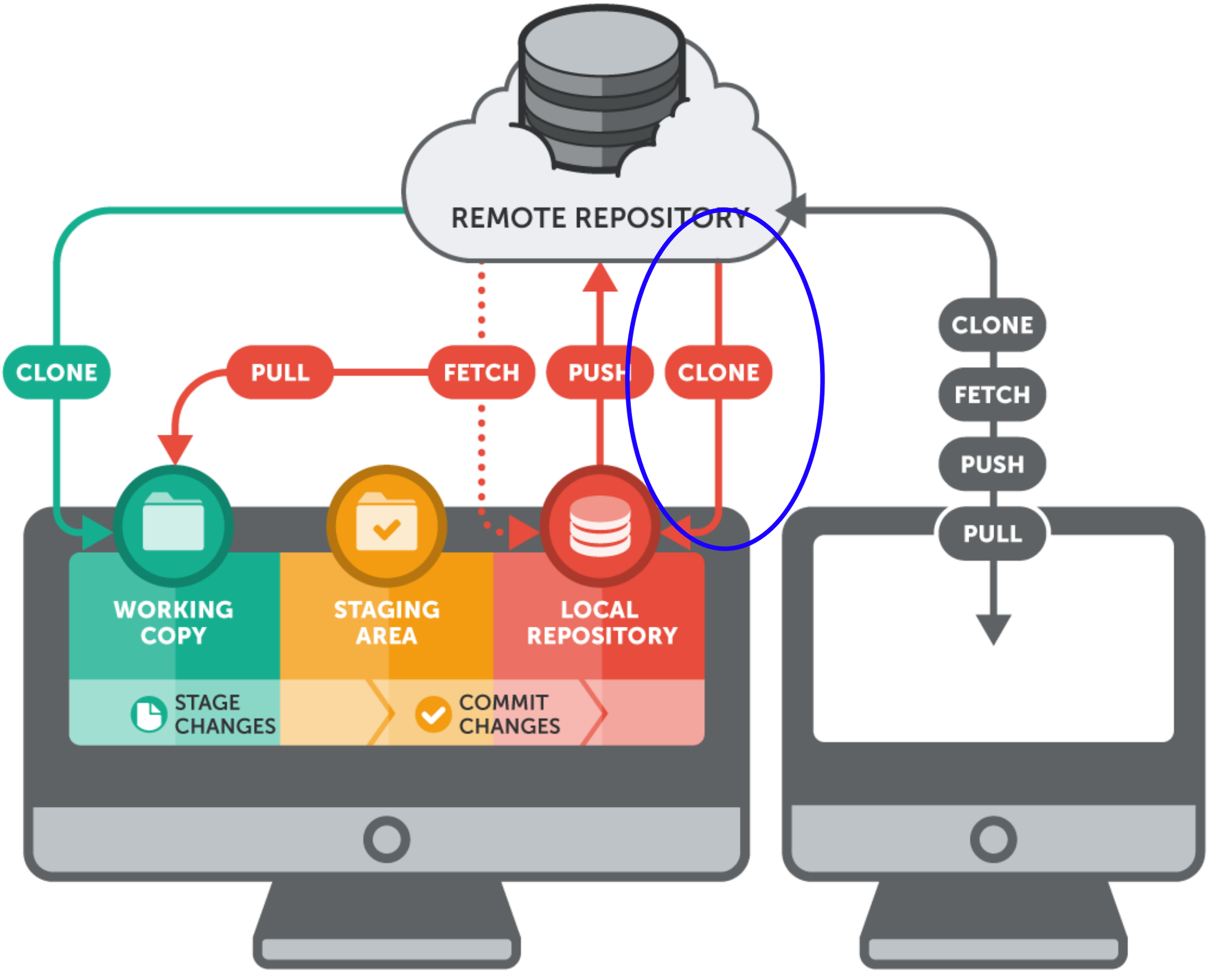
$\Longrightarrow$ ***Pull or Fetch***



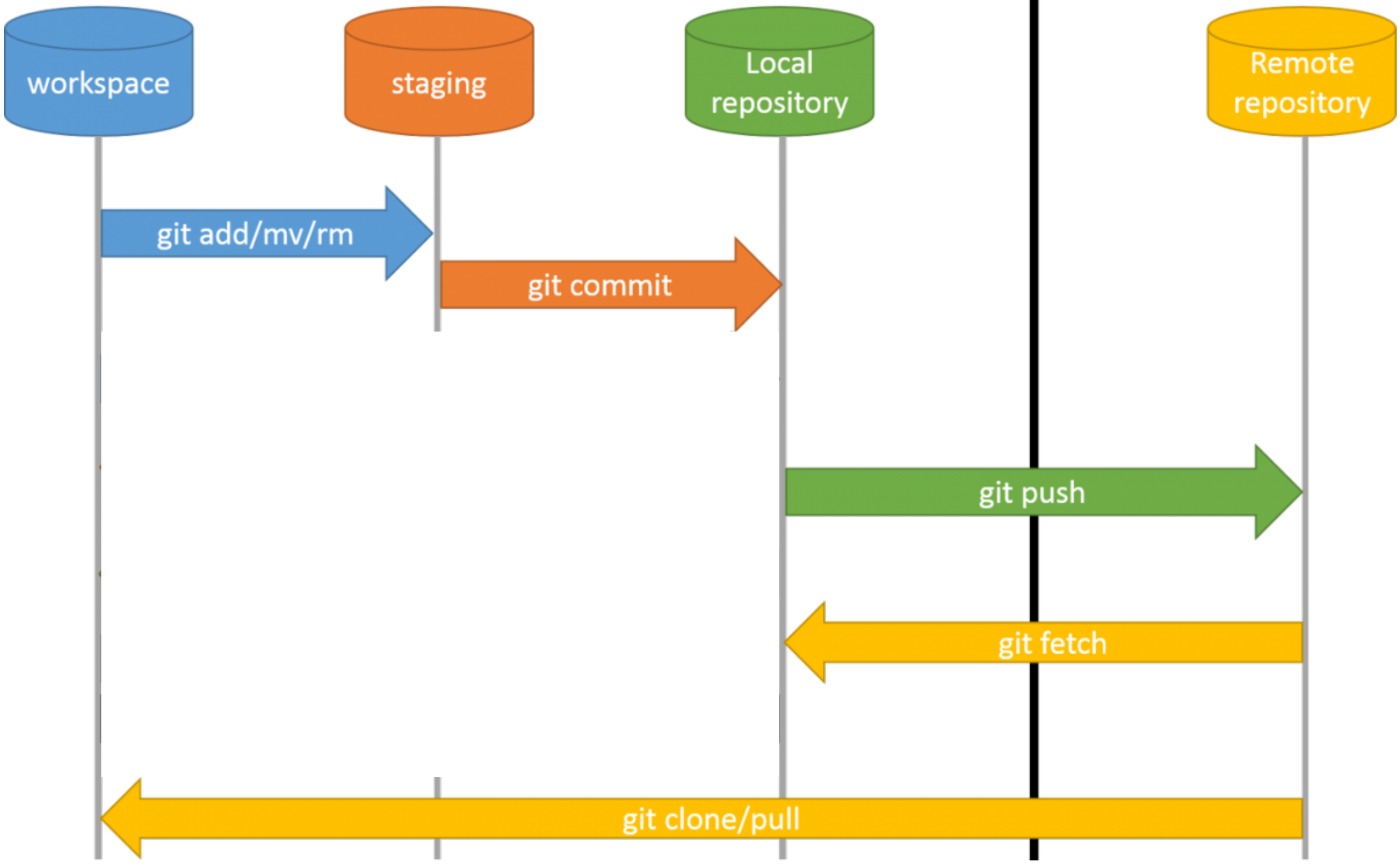
Git the Main Idea:



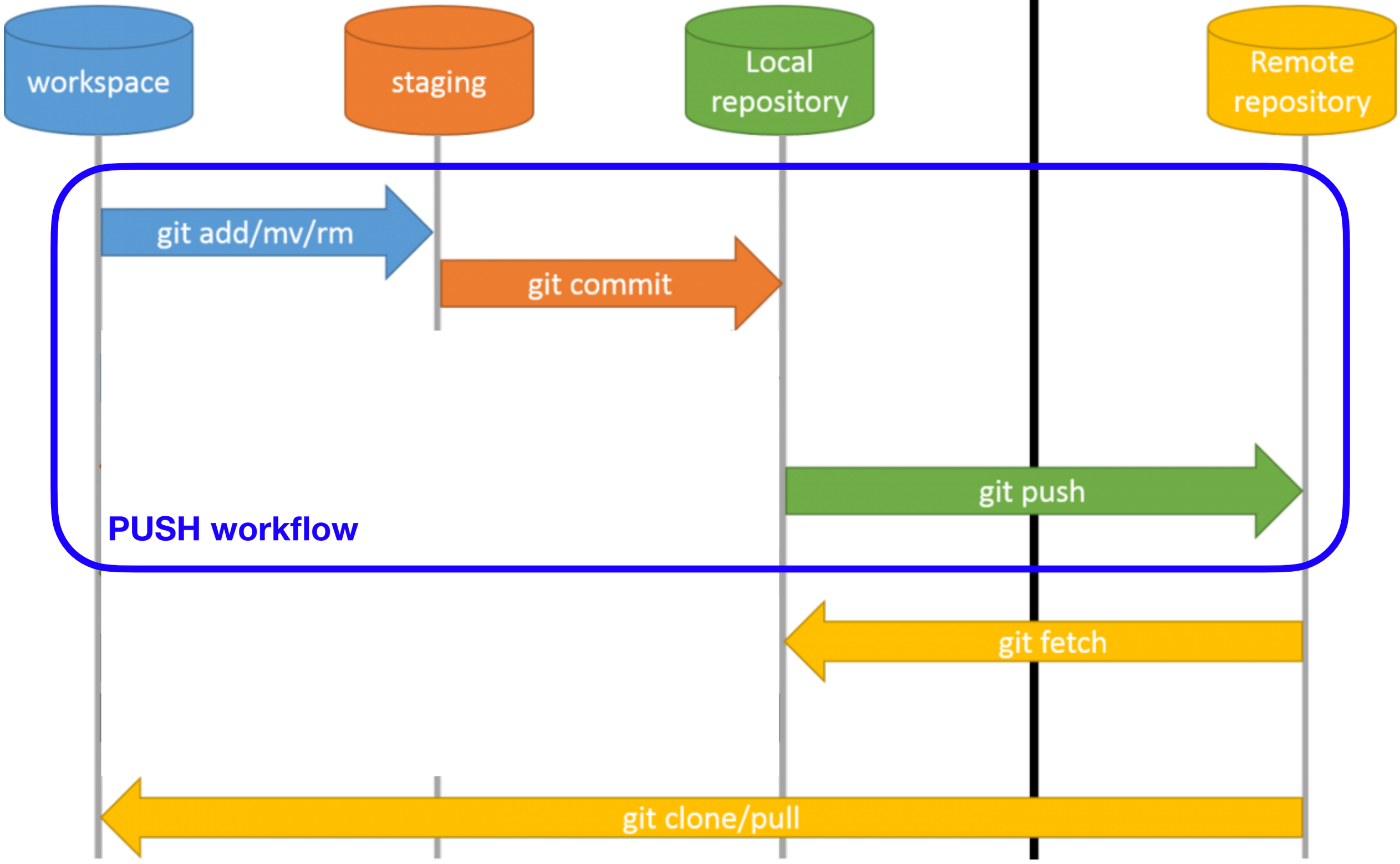
Git the Main Idea:



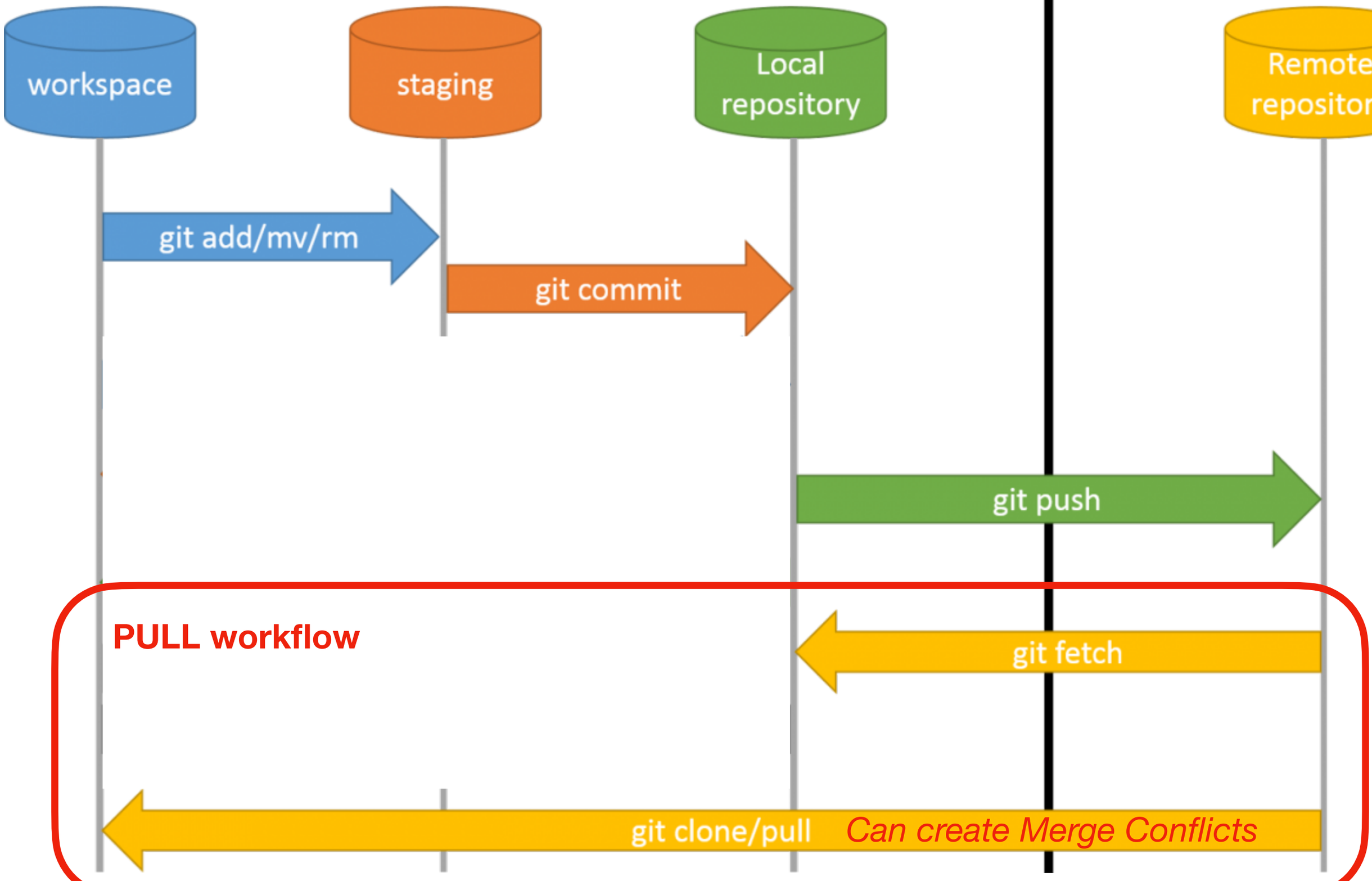
The Push-Pull Workflow



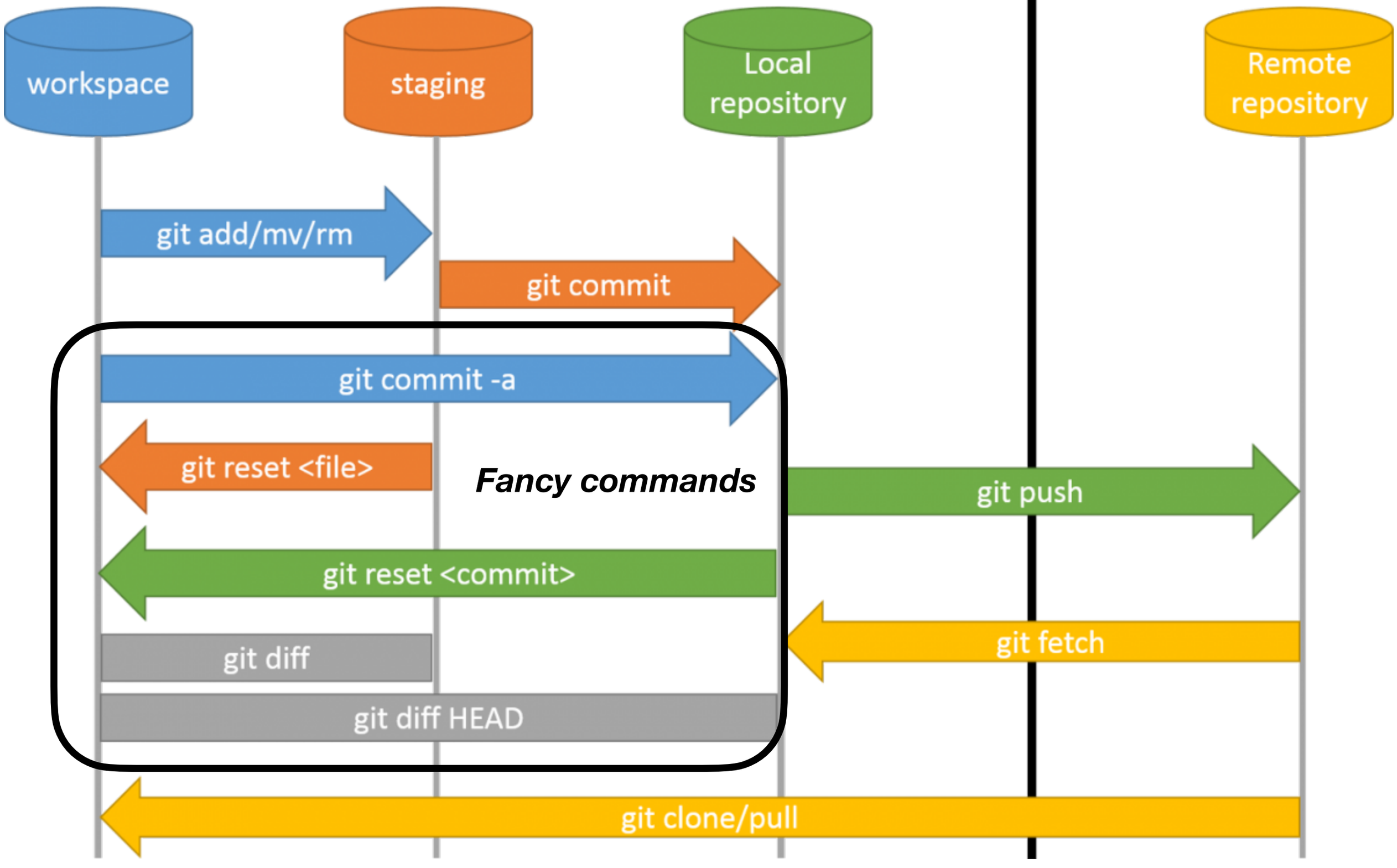
The Push-Pull Workflow



The Push-Pull Workflow

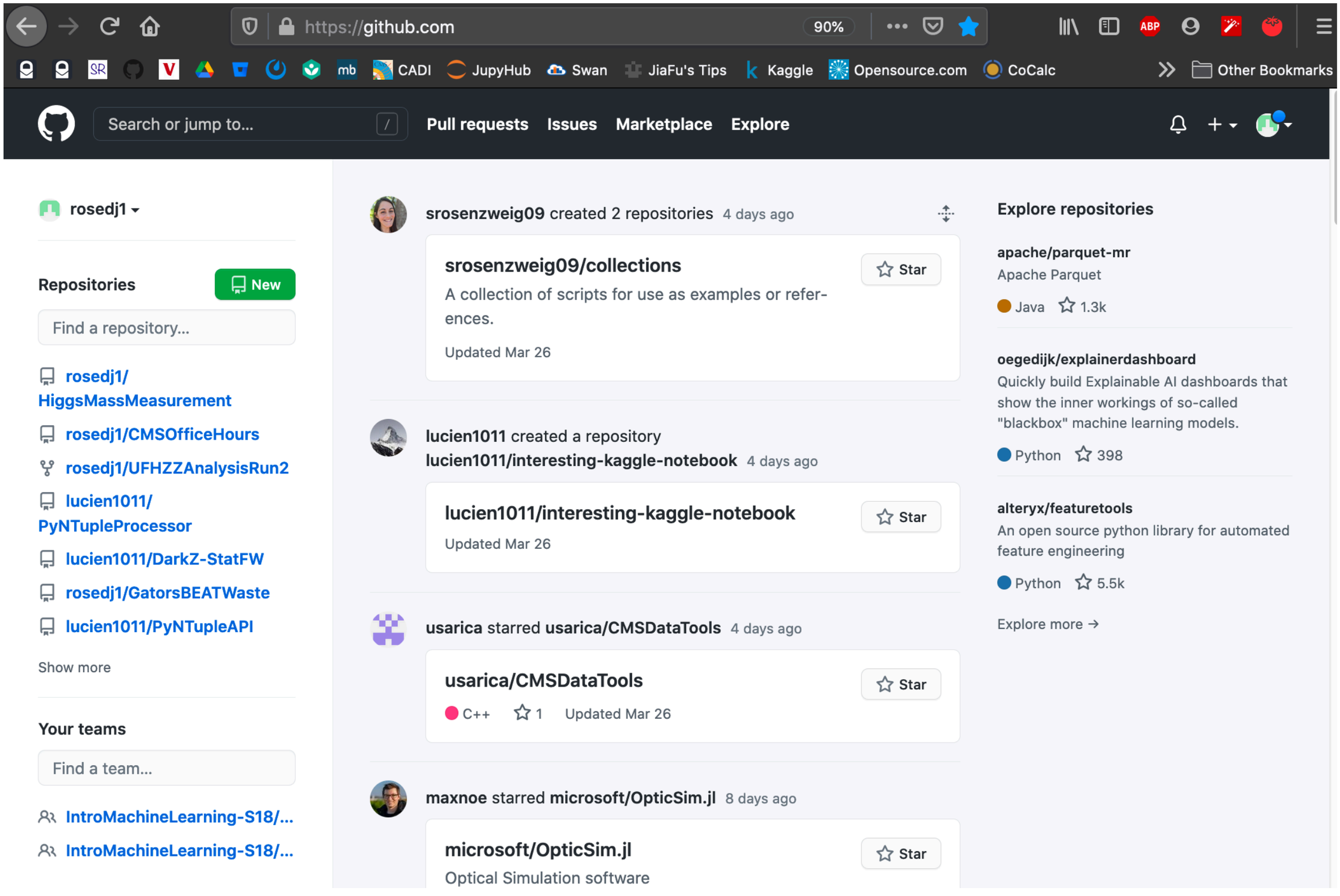


The Push-Pull Workflow



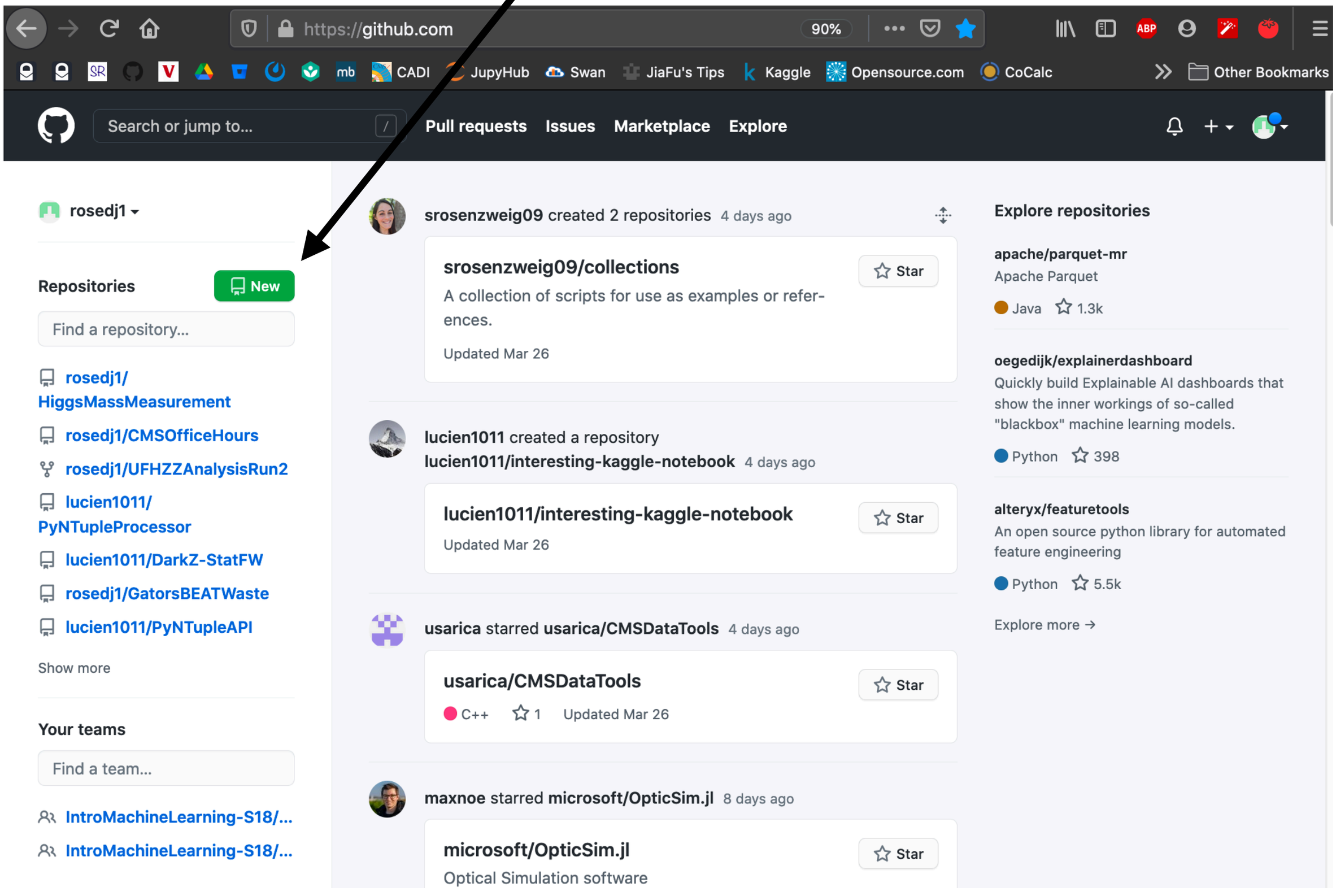
Let's practice! Make your own repo:

- Go to: https://github.com/



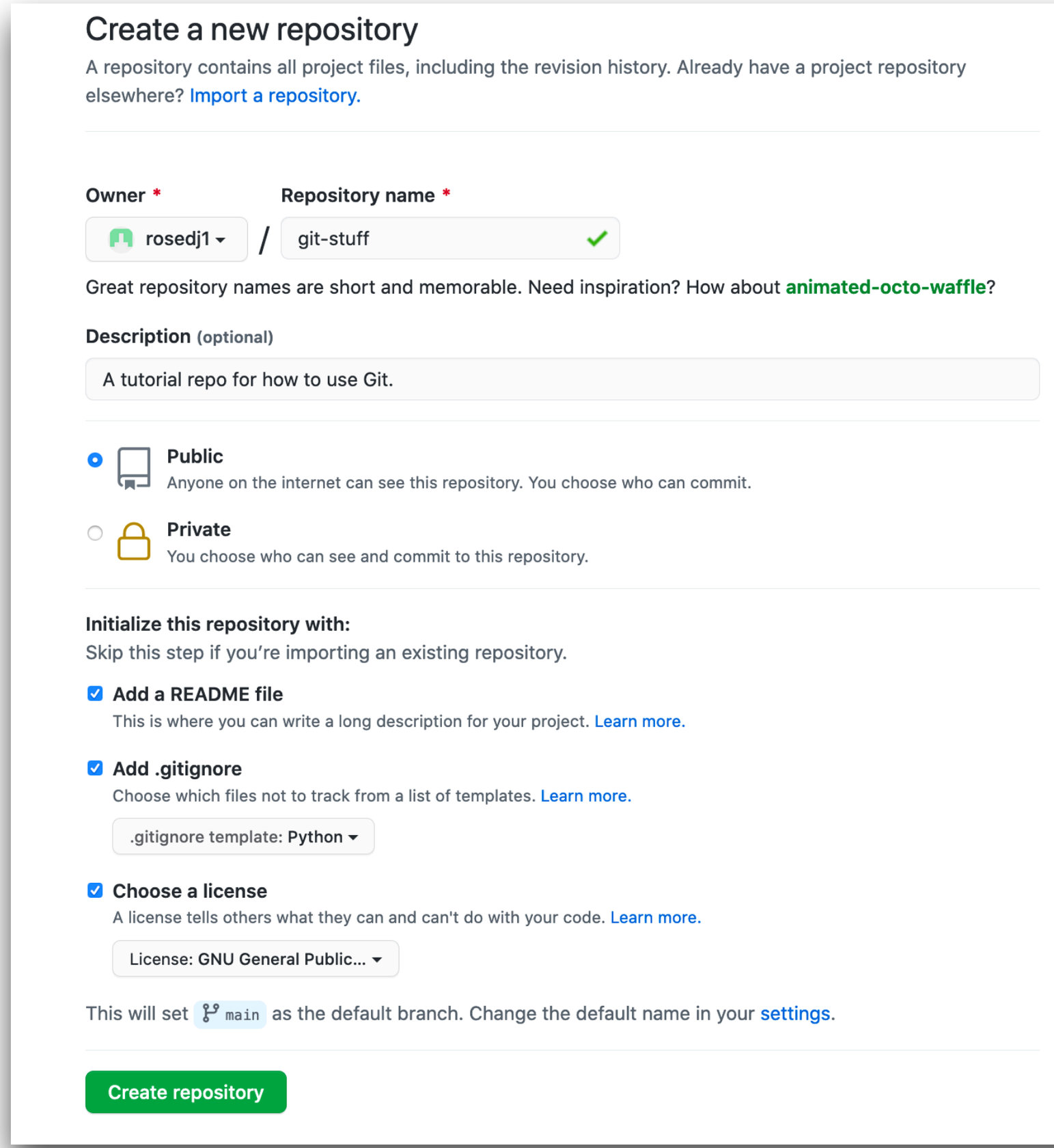
Let's practice! Make your own repo:

- Go to: https://github.com/ Click the **green 'New' Button**



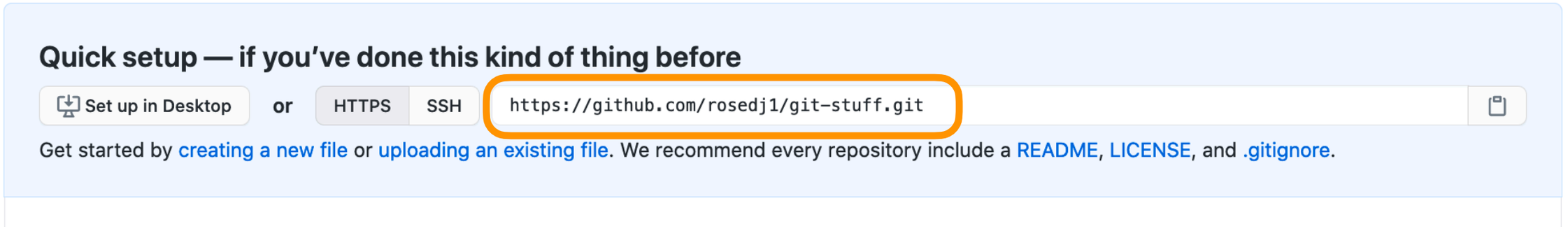
Let's practice! Make your own repo:

- Ideas for repos:
  - Store your programming code
  - Personal projects
  - Company's work (Google, FB)
  - Class/lecture notes, presentations, HW
  - *Anything you want to save*
    - Nothing too large (<=50 MB/file)
    - No pictures, huge PDFs



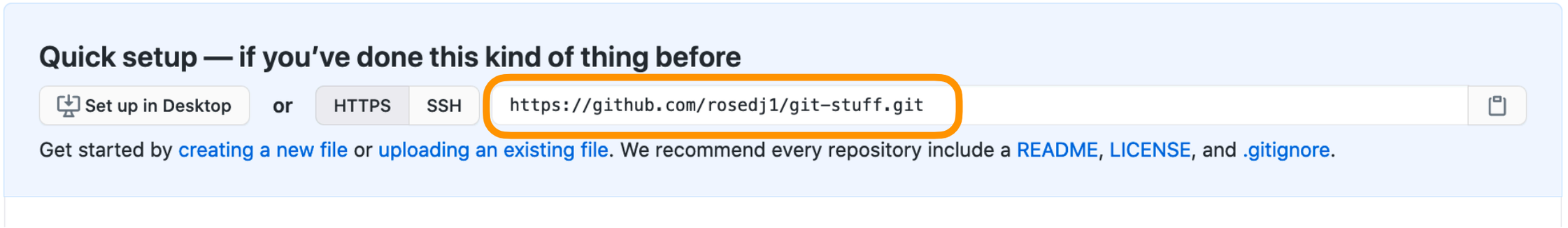
Let's practice! Make your own repo:

- Copy the URL:



Let's practice! Make your own repo:

- Copy the URL:



- Clone it to your local machine: `git clone https://...`

```
(base) Jakes-MacBook-Pro:$ mkdir ~/Practice
(base) Jakes-MacBook-Pro:$ cd !$
cd ~/Practice
(base) Jakes-MacBook-Pro:/Users/Jake/Practice/$ git clone https://github.com/rosedj1/git-stuff.git
Cloning into 'git-stuff'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
(base) Jakes-MacBook-Pro:/Users/Jake/Practice/$ ls
total 0
drwxr-xr-x  6 Jake  staff   192B Mar 30 15:14 git-stuff
(base) Jakes-MacBook-Pro:/Users/Jake/Practice/$ cd git-stuff/
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ pwd
/Users/Jake/Practice/git-stuff
```

Your local version of the remote repo

- Anything added inside your local repo can be **tracked by git.**

Let's practice! Add some code:

- Make some code in your local repo:

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "import numpy" > newcode.py
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ cat newcode.py
import numpy
```

Let's practice! Add some code:

- Make some code in your local repo:

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "import numpy" > newcode.py
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ cat newcode.py
import numpy
```

- What does Git currently know about?

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        newcode.py

nothing added to commit but untracked files present (use "git add" to track)
```

Let's practice! Add some code:

- Make some code in your local repo:

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "import numpy" > newcode.py
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ cat newcode.py
import numpy
```

- What does Git currently know about?

which branch you're on →

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        newcode.py

nothing added to commit but untracked files present (use "git add" to track)
```

The nickname of your remote repo

Git sees this new code but won't track it yet →

- Use git status often! *Informative.*

Let's practice! Add some code:

- Make some code in your local repo:

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "import numpy" > newcode.py
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ cat newcode.py
import numpy
```

- What does Git currently know about?

which branch you're on →

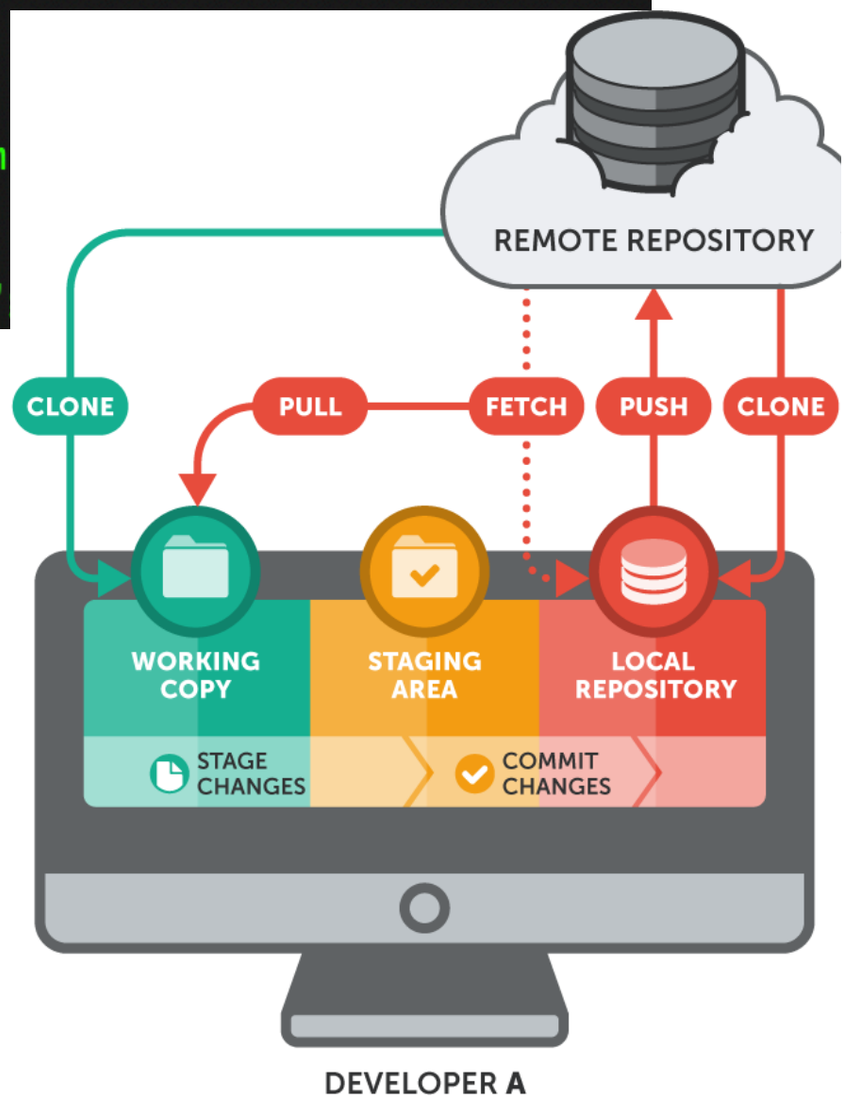
```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be comm
        newcode.py

nothing added to commit but untracked files present (use "
```

The nickname of your remote repo

Git sees this new code but won't track it yet →



- Use git status often! *Informative.*

- Q: In which part of the workflow do **untracked files** live?

Let's practice! Add some code:

- Make some code in your local repo:

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "import numpy" > newcode.py
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ cat newcode.py
import numpy
```

- What does Git currently know about?

which branch you're on →

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be comm
        newcode.py

nothing added to commit but untracked files present (use "
```

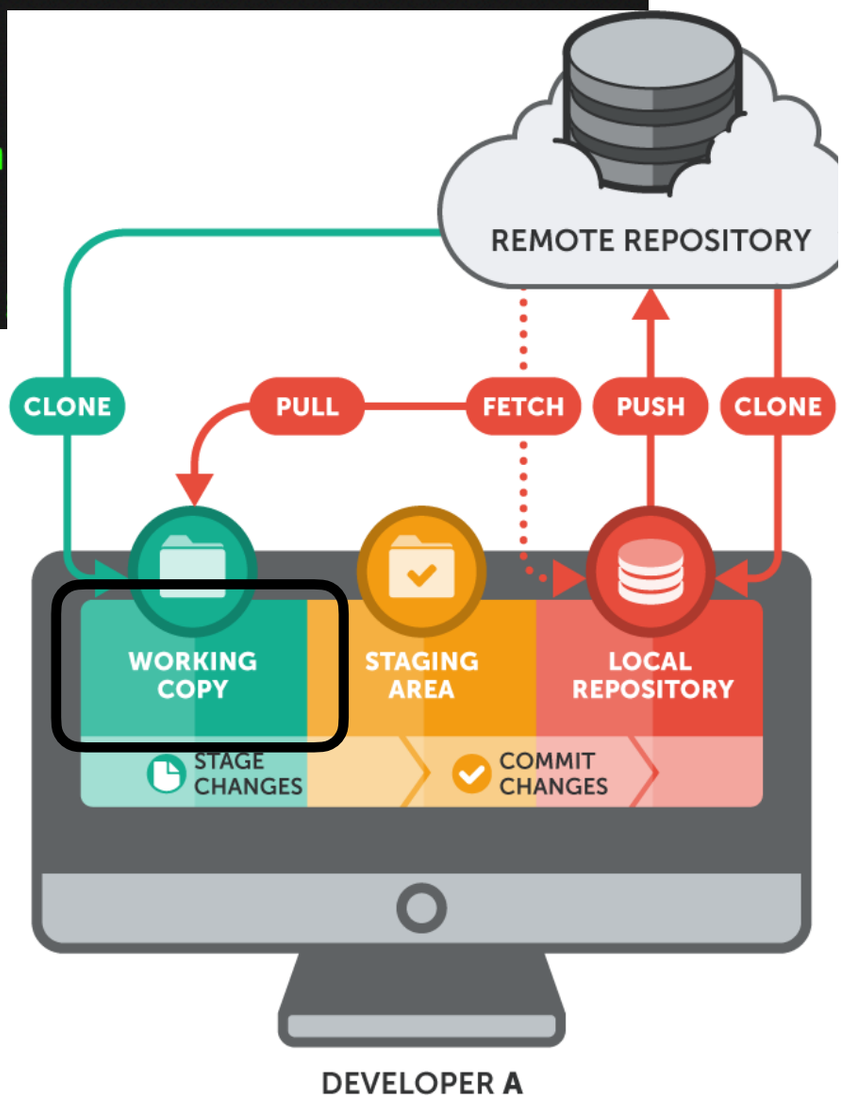
The nickname of your remote repo

Git sees this new code but won't track it yet →

- Use git status often! *Informative.*

- Q: In which part of the workflow do **untracked files** live?

Untracked files live in your **Working Area**



Let's practice! Add some code:

- Make some code in your local repo:

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "import numpy" > newcode.py
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ cat newcode.py
import numpy
```

- What does Git currently know about?

which branch you're on →

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be comm
        newcode.py

nothing added to commit but untracked files present (use "
```

The nickname of your remote repo

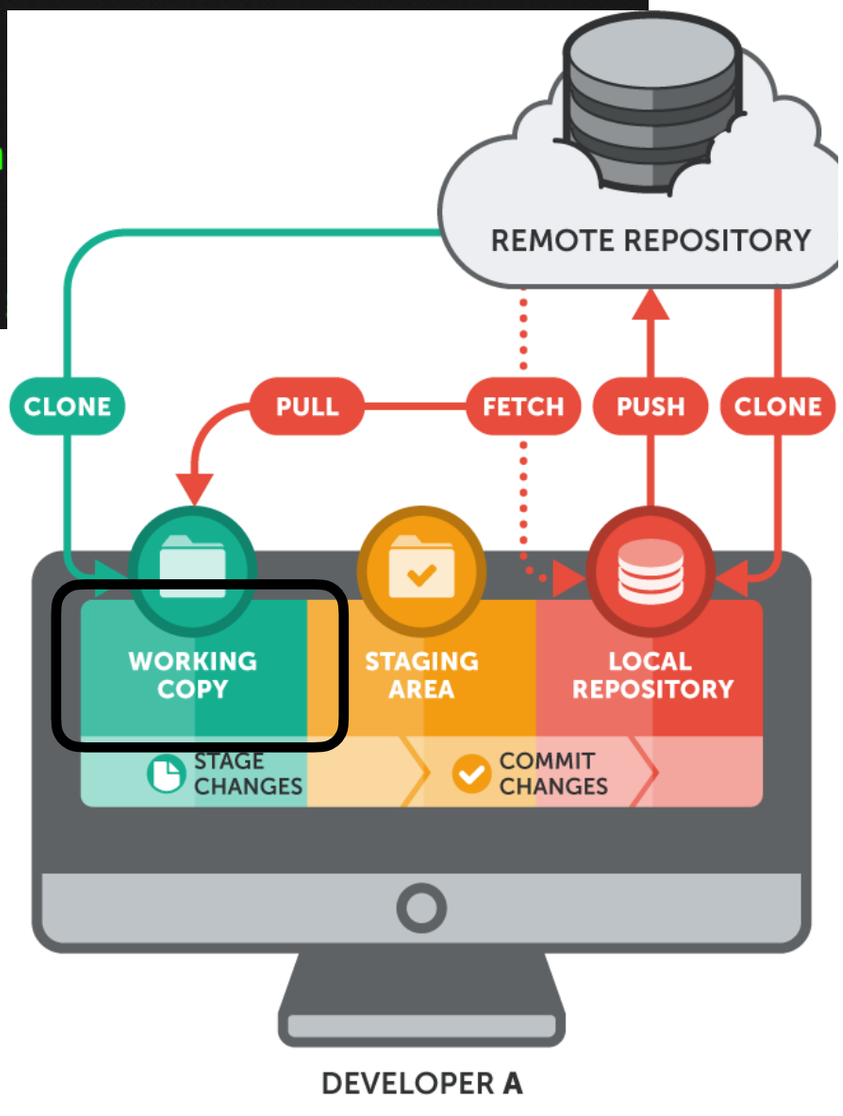
Git sees this new code but won't track it yet →

- Use git status often! *Informative.*

- Q: In which part of the workflow do **untracked files** live?

- Q: How do we add untracked files to the **Staging Area**?

Untracked files live in your **Working Area**



Let's practice! Add MORE code:

- Q: How do we add untracked files to the **Staging Area**?
  - git add <file>

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   newcode.py
```

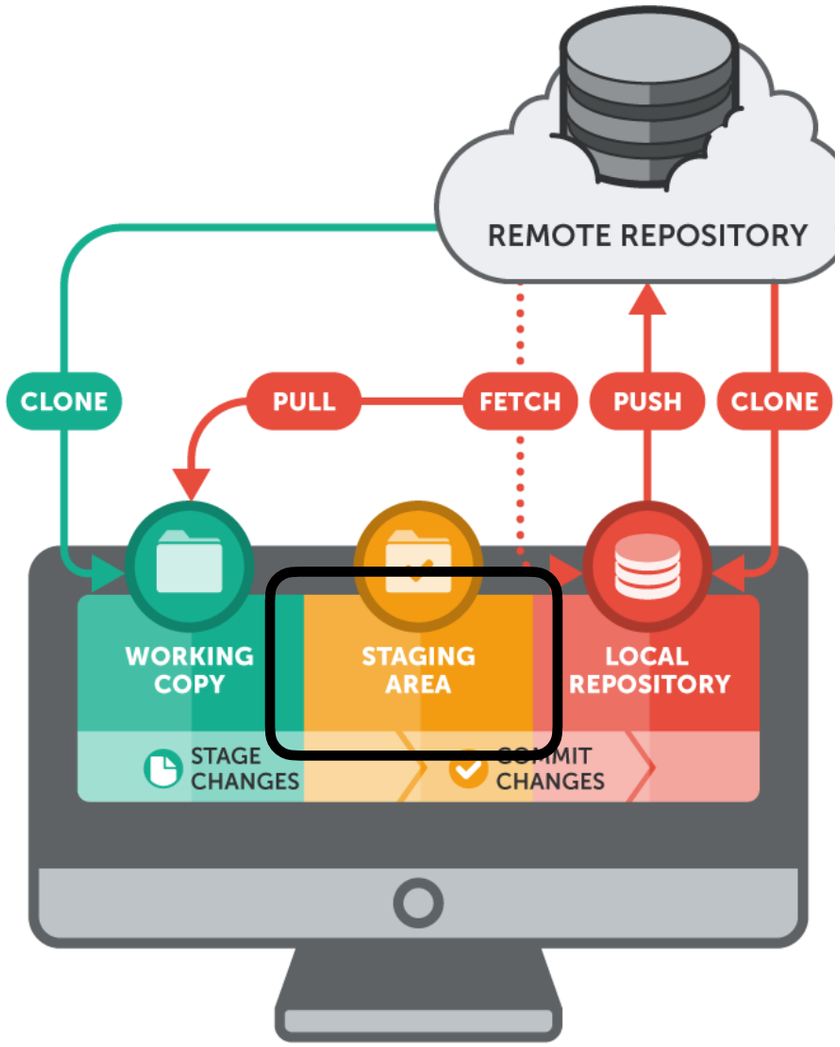
- Now this file is **staged**.
  - Git is ready to commit. *Are you?*

- *Don't commit yet!*
  Create another file called "second.txt".
  Then `git status`.

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ echo "some words" > second.txt
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   newcode.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        second.txt
```



Added files live in your **Staging Area**

Q: *What is this Git message telling us?*

newcode.py has **already been staged**.
It is ready to be **committed** (saved).

But second.txt is **untracked** (new).
This file is still in the Working Area.

Let's practice! Commit your code:

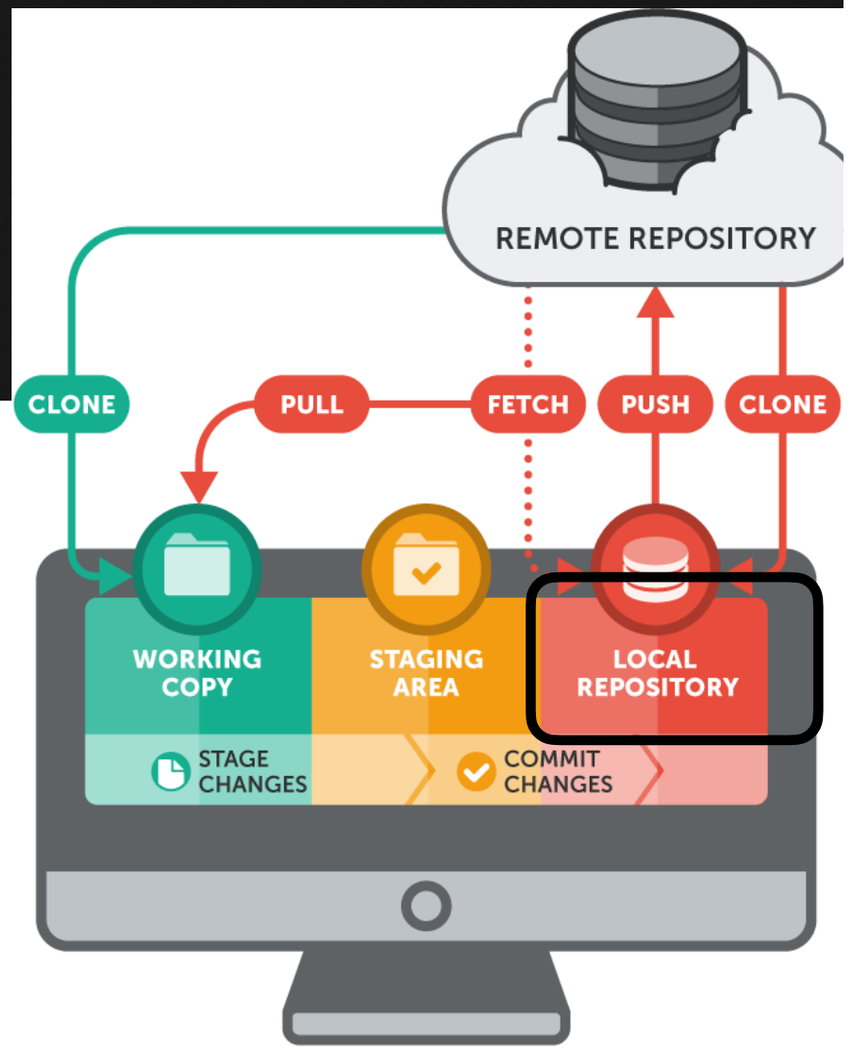
```
[(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git commit -m "Added new code for testing." -m "It simply imports numpy."
[main 2638c0b] Added new code for testing.
 1 file changed, 1 insertion(+)
 create mode 100644 newcode.py
[(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        second.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Primary message

Secondary message



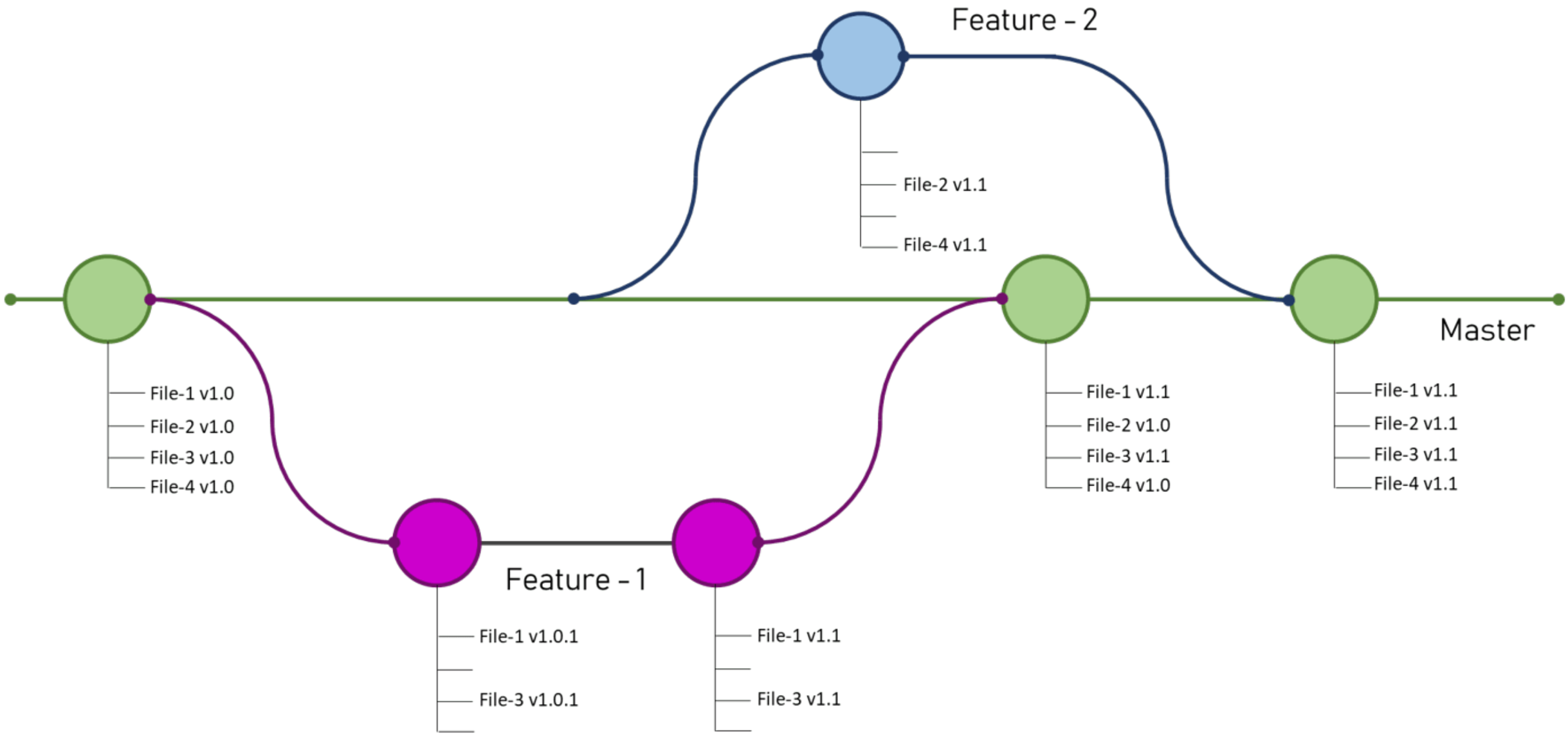
- Now your code is saved in your **Local Repo**!

- Q: How do make this local code talk to your remote repo?
  - Push it up!

```
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git branch
* main
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git remote
origin
(base) Jakes-MacBook-Pro:/Jake/Practice/git-stuff/$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rosedj1/git-stuff.git
   414df0b..2638c0b  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Branches

- So far you've been working on a branch called **"main"** (master)

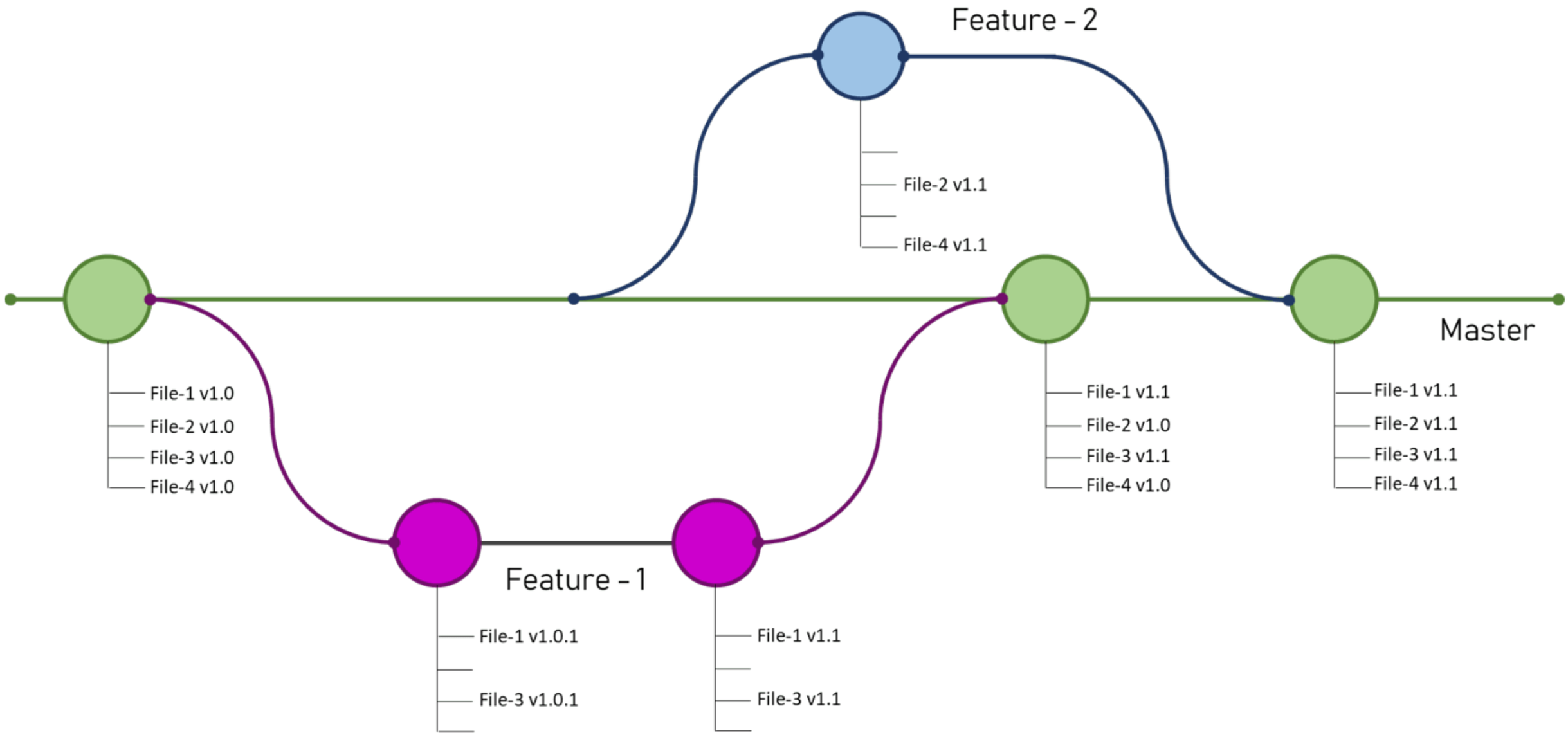


Branches

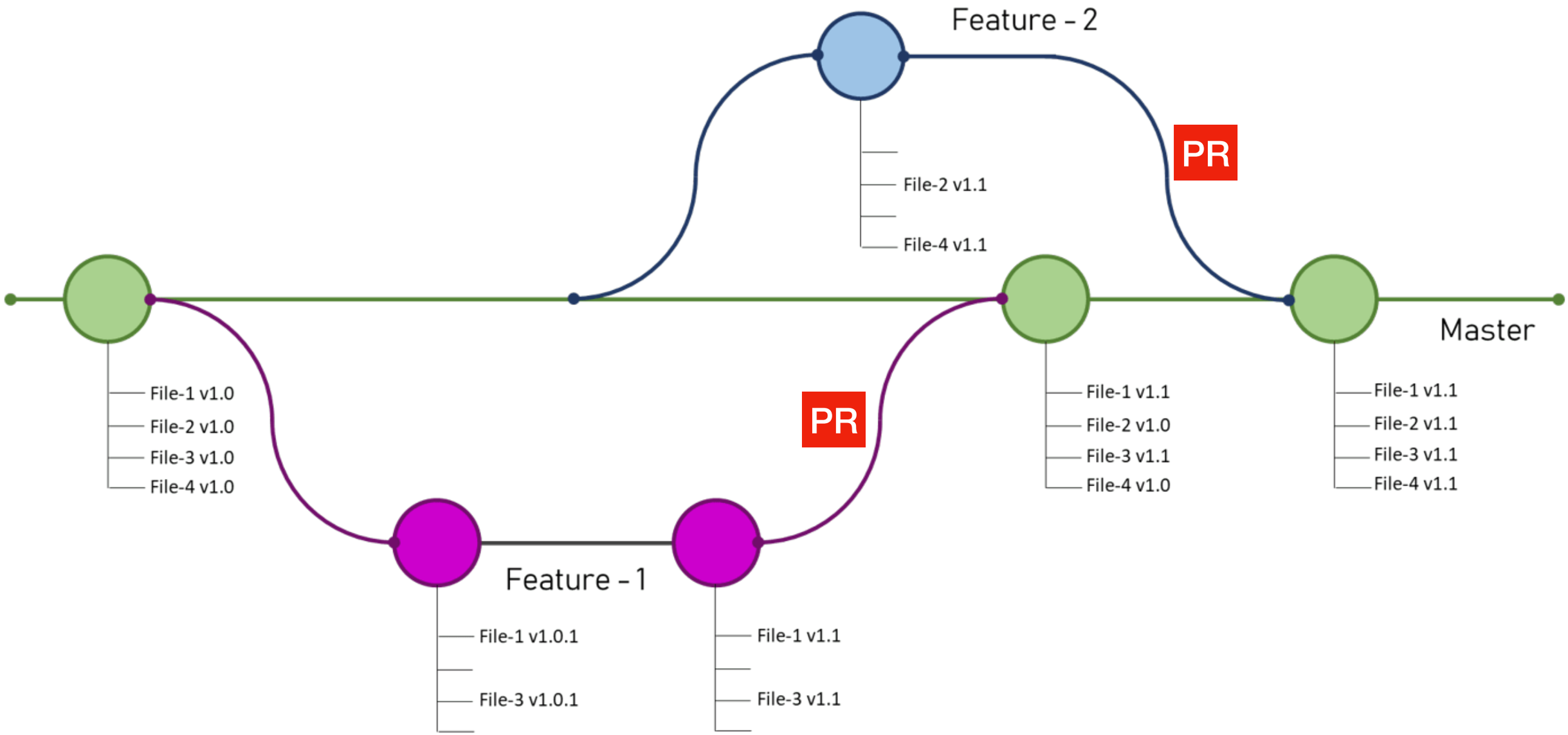
- So far you've been working on a branch called **"main"** (master)

- It's a better idea to work on your own branch: **"Feature-1"**

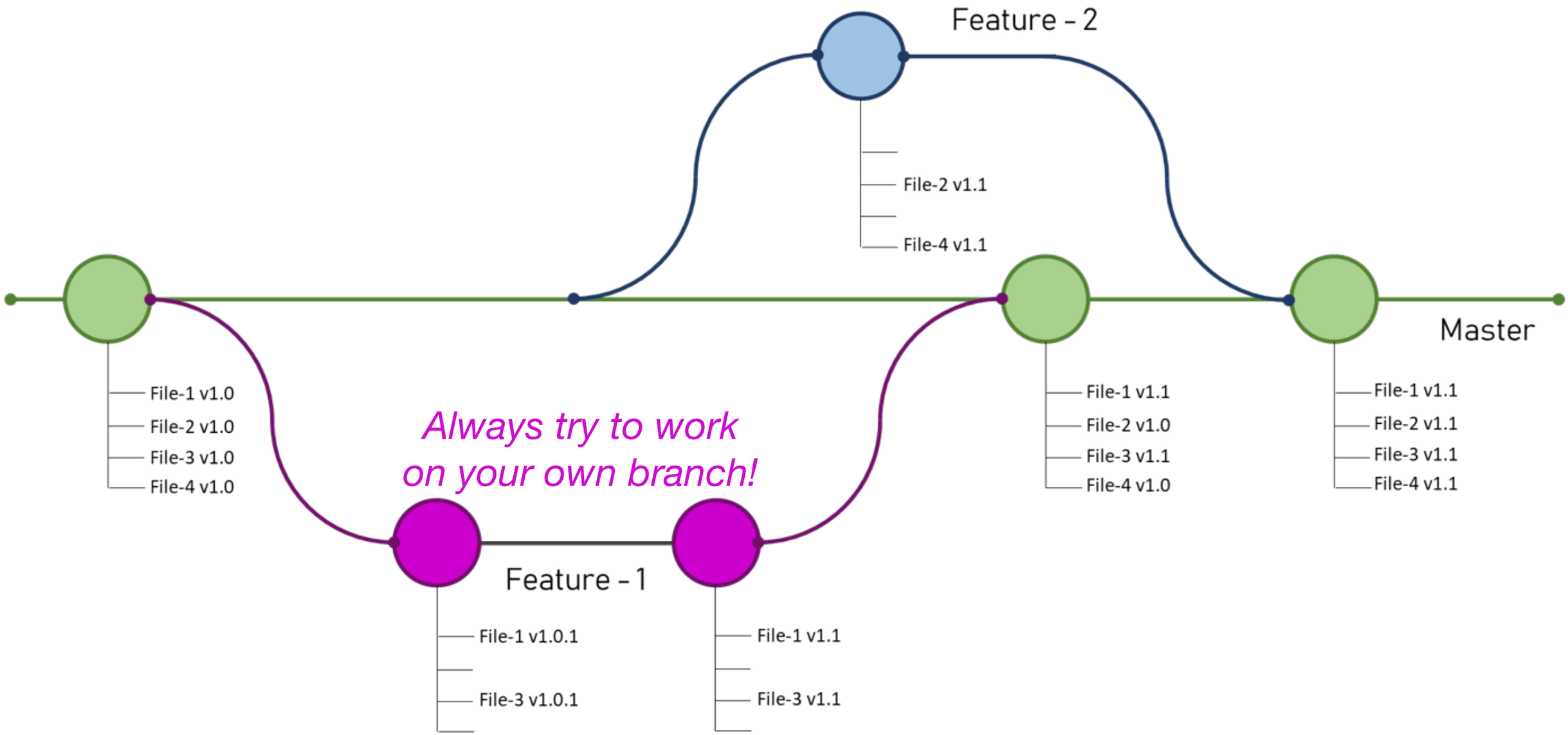
```
git checkout -b Feature-1
```



Branches: Pull Requests merge branches



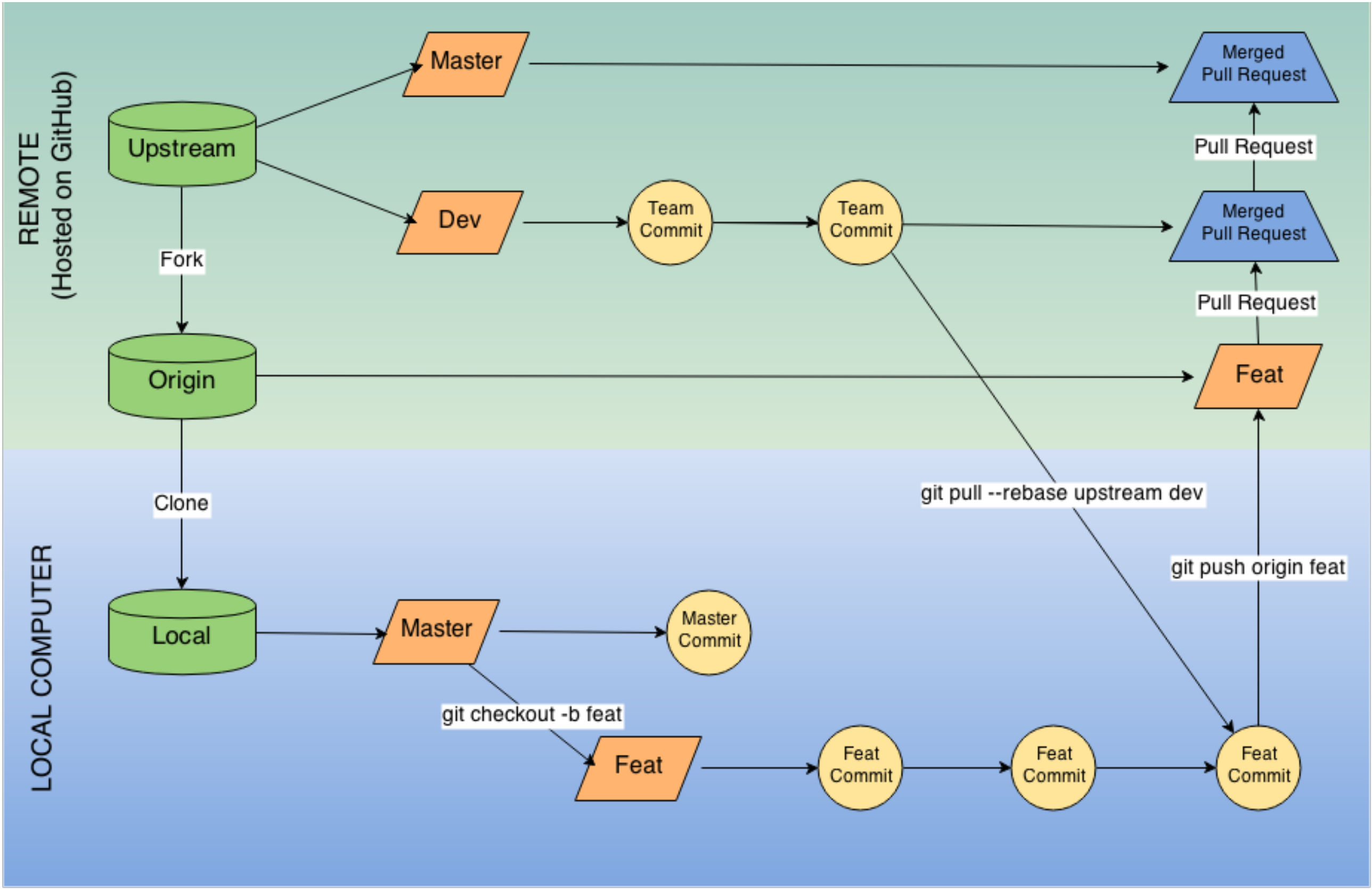
Branches: Use your own branch



Redo Task 1, but do it the safe way

- Make your own branch
- Make new code
- Then Add, Commit, and Push to remote repo:
  - Add: git add <newestfile.txt>
  - Commit: git commit -m "Some descriptive message."
  - Push: git push origin feature-1
- Make a PR.
- Merge PR.

Git Workflow



Other commands

- **git reset --hard** //Used to reset your repo and delete all commits tracked by GitHub so if you pull it will completely overwrite everything and will not try to merge.
- **git stash** //git stash will store your local changes without pushing them to the repo
- **git stash pop** //apply the changes you stashed again
- **git log --graph --oneline**
- **git diff file1..file2**
- **git diff file1...file2**

Now you know these commands

Git Tips:

- Push often!
  - Like every day
- Don't work on the main branch
  - Use your own branch
  - Push to remote
  - Make PR to merge your branch into main
- Use detailed messages
  - Not only in commit messages, but in PRs, etc.

Let's practice! Task 2: Fork this repo

- In terminal, go to your home area:
  - `cd ~`
- Point to CMS OH repo:
  - Go to: https://github.com/rosedj1/
  - Click **CMS_Office_Hours**
  - Click green button
  - Copy HTTPS link
  - Go to: https://github.com/rosedj1/
    - Click CMS_Office_Hours
    - Click green button
    - Copy HTTPS link ow your turn to fork the repo
    - https://github.com/rosedj1/CMS_Office_Hours
- Clone remote repo to your **local area**
  - This is YOUR local repo!
  - Will not affect the remote repo (unless you tell it to)
  - `git clone https://...`
- Make (checkout) a new branch
  - Do not work on the master branch!
  - `git checkout -b <mynewbranch>`
- Begin creating new code!
  - Begin creating new code!
- Now Sean's changes are on HIS remote repo on his own branch (not master)
- Do PR into origin/master
- Everyone else can pull down Sean's changes.

Backup Slides

Commits

- A commit is a **snapshot of your files**

git diff

- Just show browser diffs

Useful branch commands

- git branch -a
  - Shows all branches
- git branch -r
  - Show all remote branches
- git branch -v
  - verbose
- git branch -h
  - See help options.
- git checkout -b **mynewbranch** origin/**onlinebranch**
  - Checkout onlinebranch that exists on the remote repo and call it mynewbranch

Terminology

- remote: online
- repository
- checkout
- fork
- branch
- origin
- HEAD
- push
- pull
- Pull Request (PR)
- stash

Let's get started

- Git is available on lxplus, HPG, etc.
- In your terminal, go to some work area (~/public/ or .../CMSSW_X)
  - /afs.cern.ch/work/d/drosenzw/HiggsMassMeasurement/

Show how to fork

- Now your turn to fork the repo
  - https://github.com/rosedj1/CMS_Office_Hours

```
git clone https://...
```

- Then clone that remote repo to your **local area**
  - This is your local repo!

```
git checkout -b mynewbranch
```

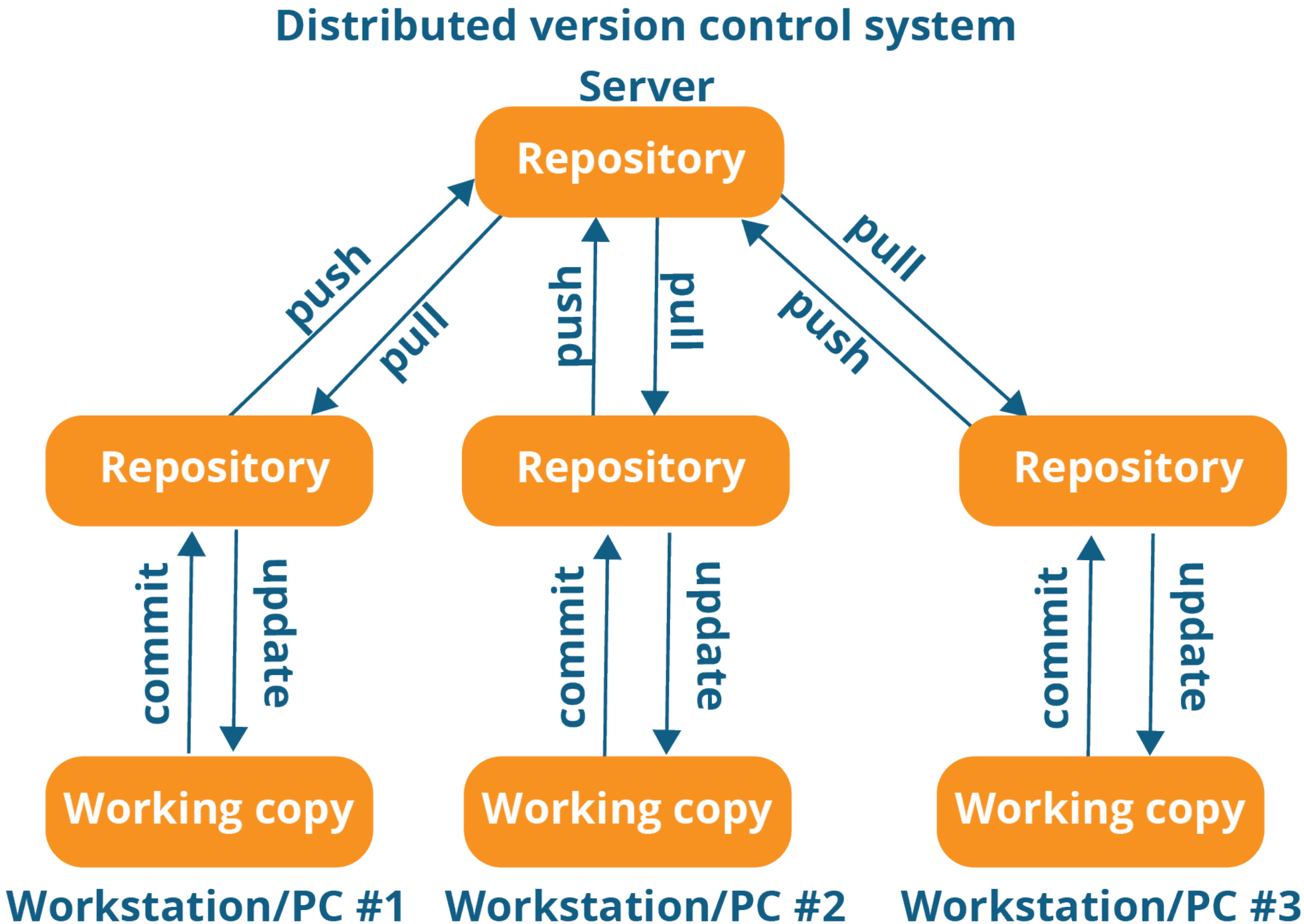
- Make (checkout) a new branch
  - Don't work the master branch!
- Begin creating new code!

Questions to answer:

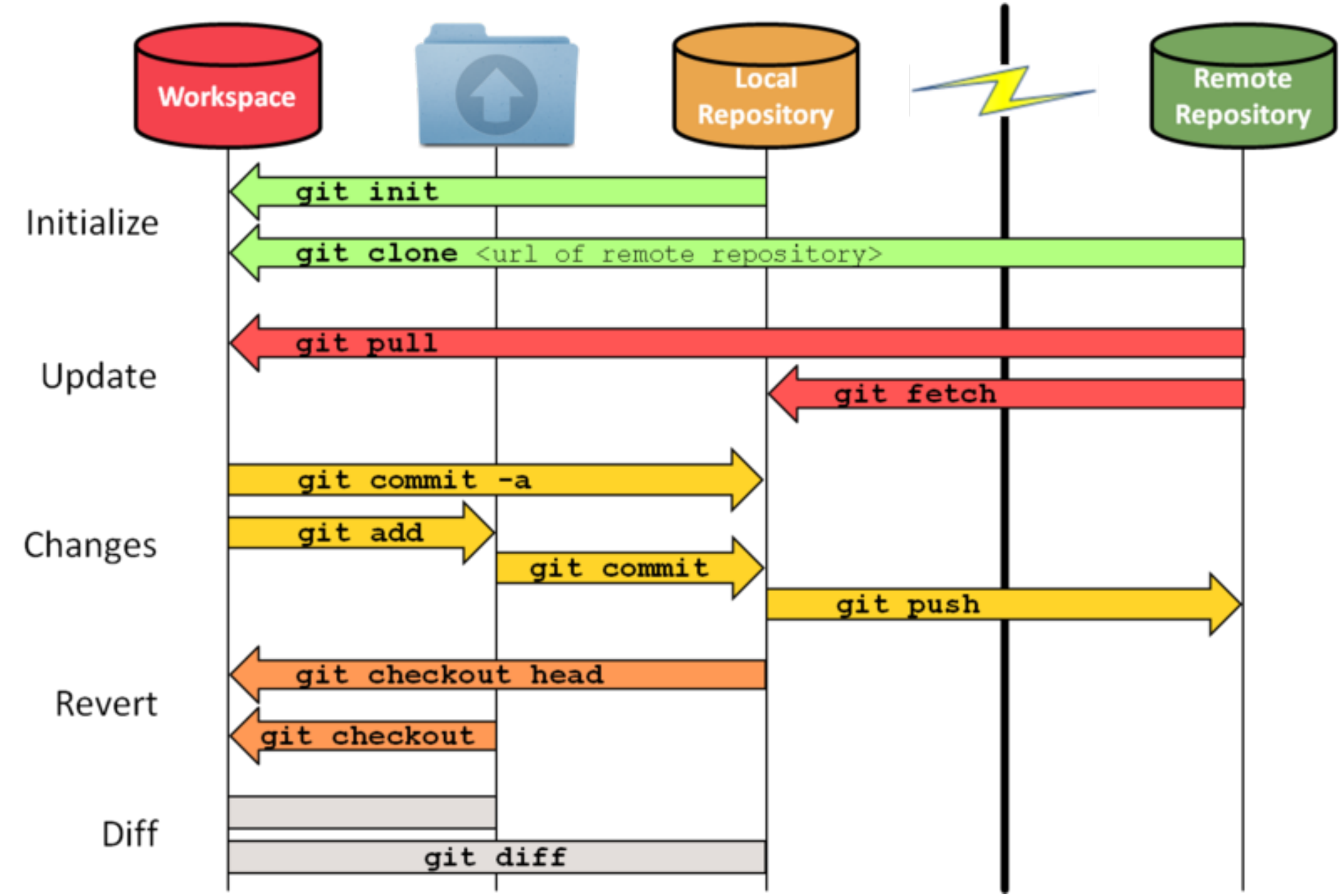
- Does fetching create a new commit?

- Why clone into commit local repo?

Main Idea


Distributed version control system
Server
Repository
push
pull
push
pull
pull
push
Repository
Repository
commit
update
commit
update
commit
update
Working copy
Working copy
Working copy
Workstation/PC #1
Workstation/PC #2
Workstation/PC #3

Git Workflow: Common Commands



What is Git?

- Git is a software that **tracks your files over time**
  - Changes
  - History

- This is *version control!*

- There are many hosts:
  - GitHub
  - BitBucket
  - GitLab
  - gogs
  - gitea

- Git is available on lxplus, HPG, etc.

What is Git?

-

Why use Git?

- Save your code

Where can I find Git?

- All

When can we learn Git?

Right now!