Git: Version Control System

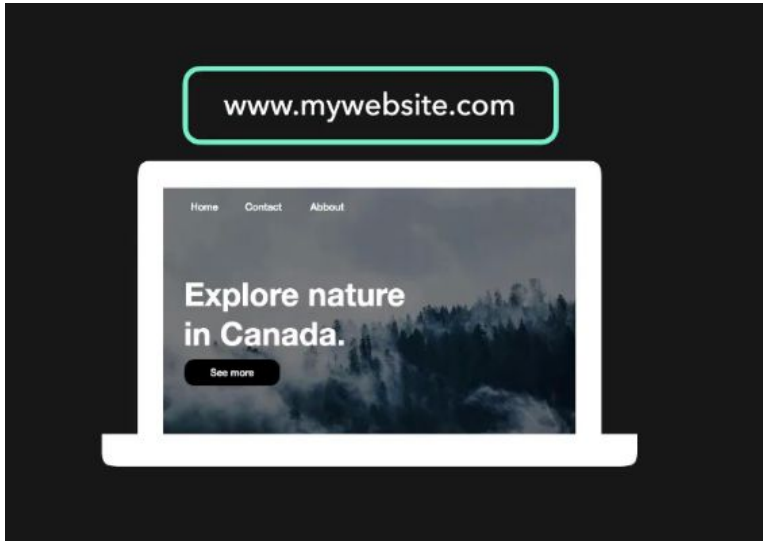
Waheed Iqbal

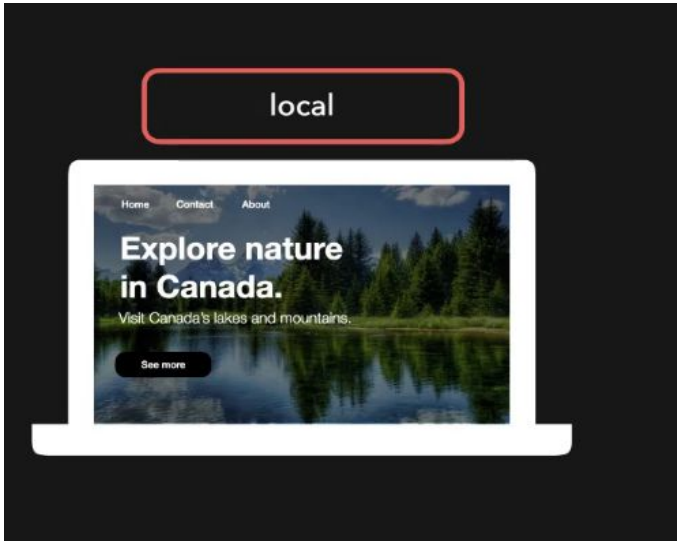
DevOps (Fall 2023)
Department of Data Science, FCIT, University of the Punjab
Lahore, Pakistan

Git: Version Control System

- Git is a distributed version control system that is widely used for tracking changes in source code during software development.
- It was created by Linus Torvalds in 2005 and has since become the de facto standard for version control in the software development industry.
- Torvalds is best known for developing the Linux kernel, the core component of the Linux operating system.

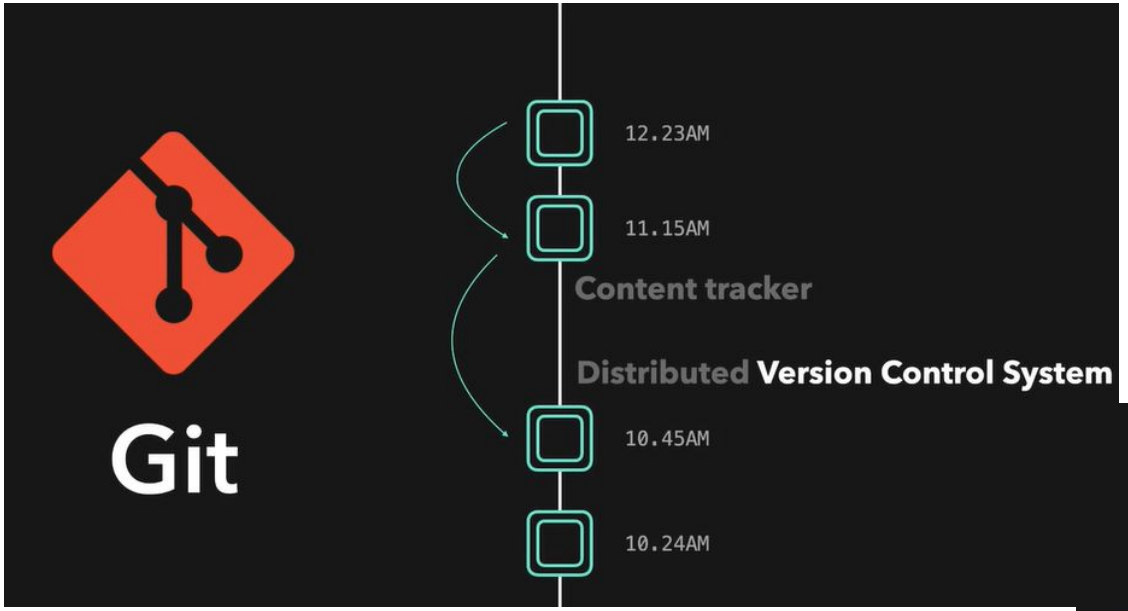
Git Introduction

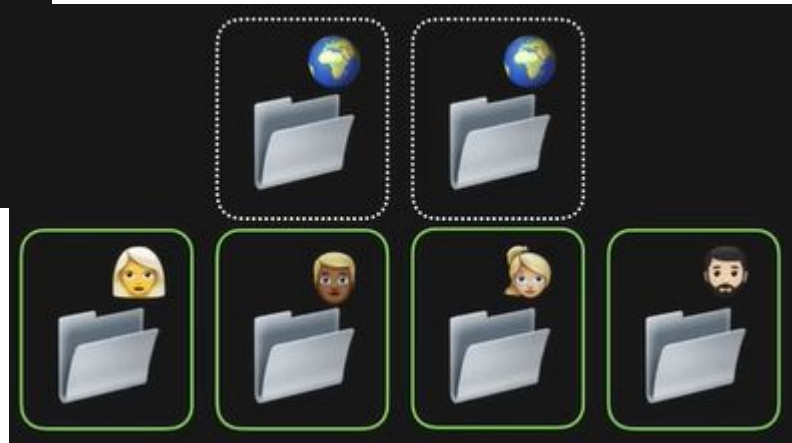




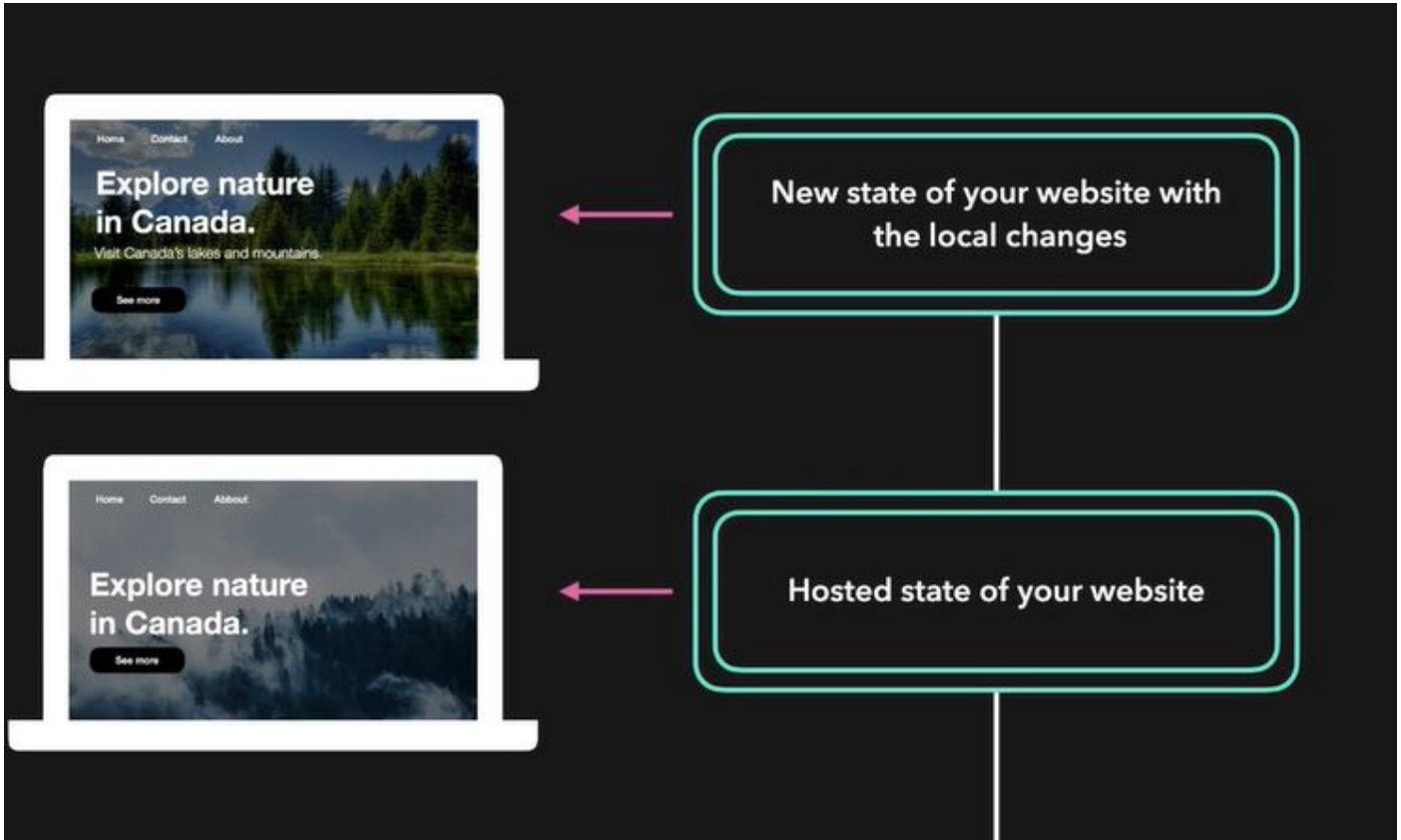
If you wanted to change your live while you have already updated local version, how would you do?

Git Introduction (Cont.)

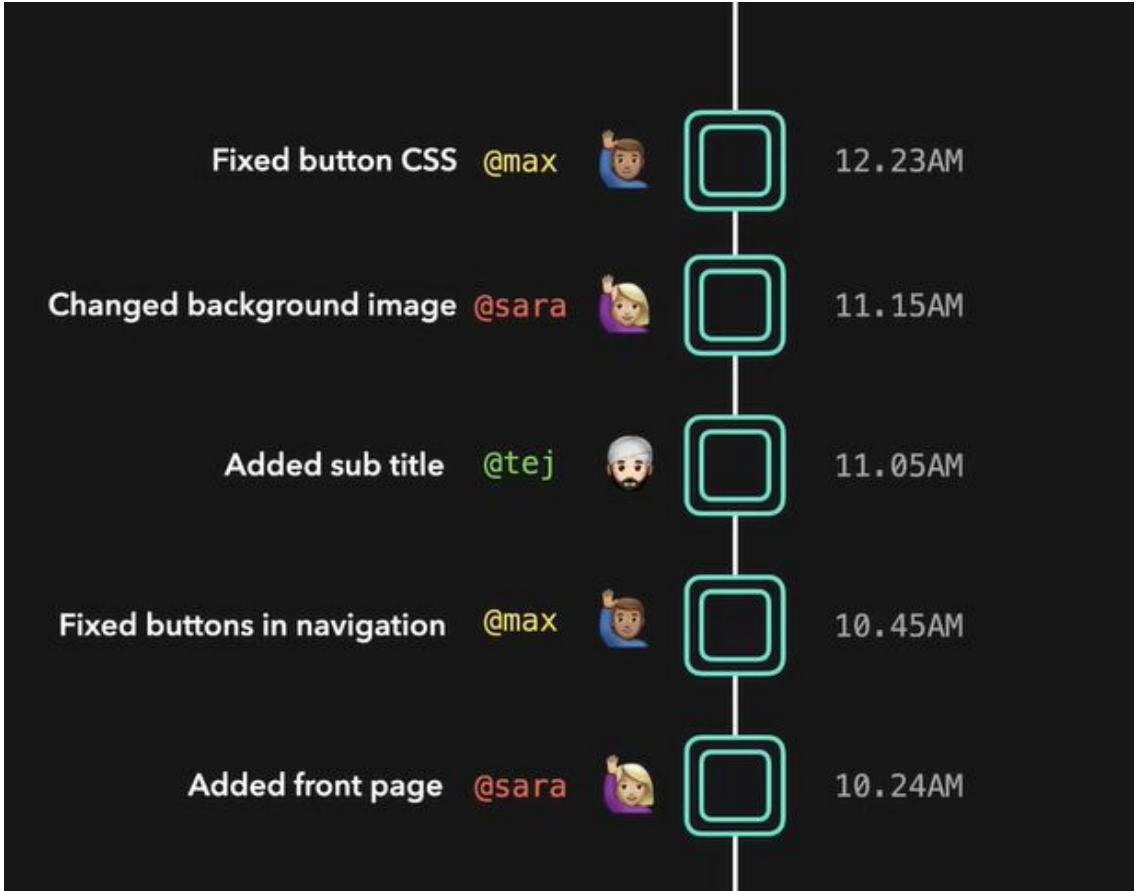




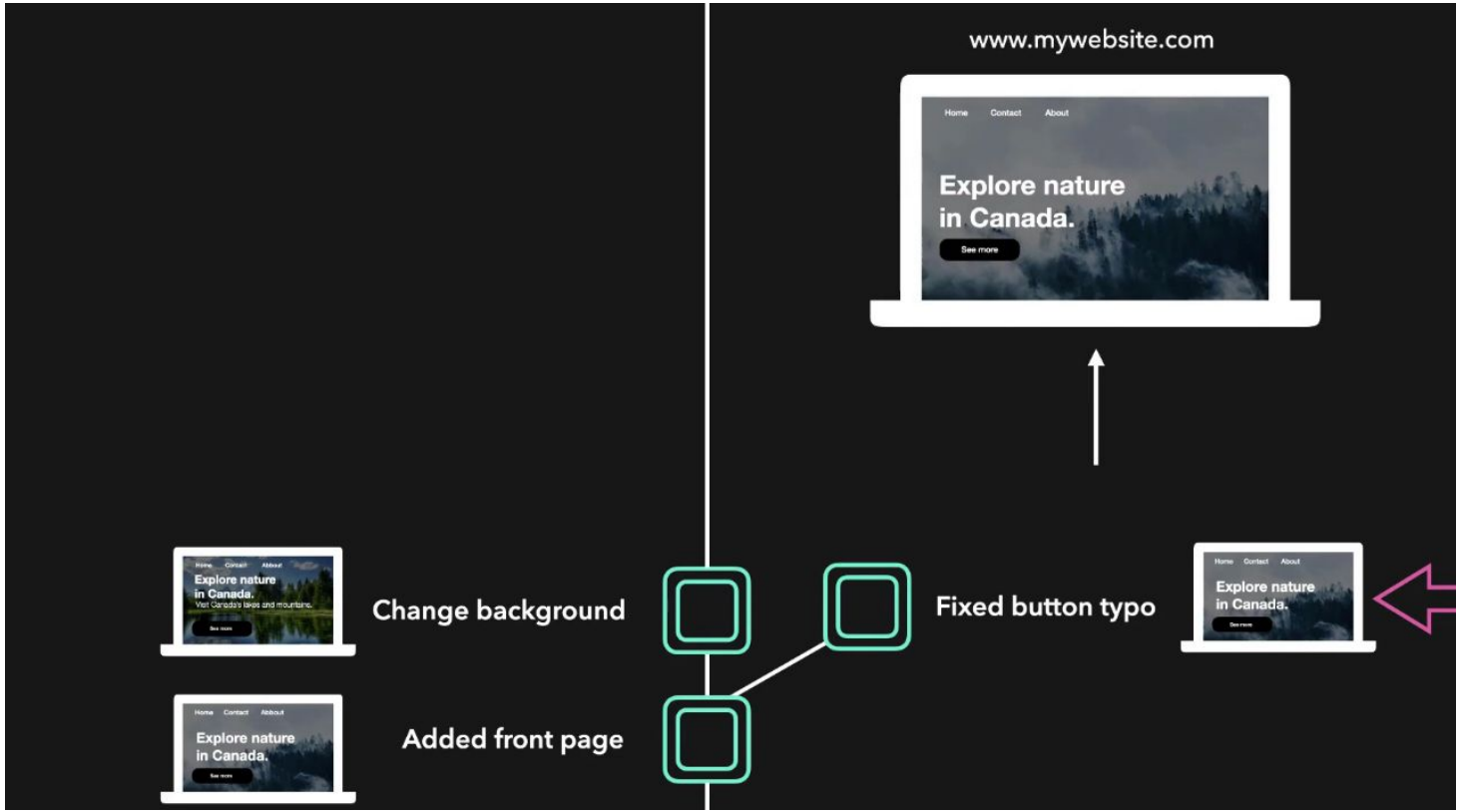
Git Introduction (Cont.)



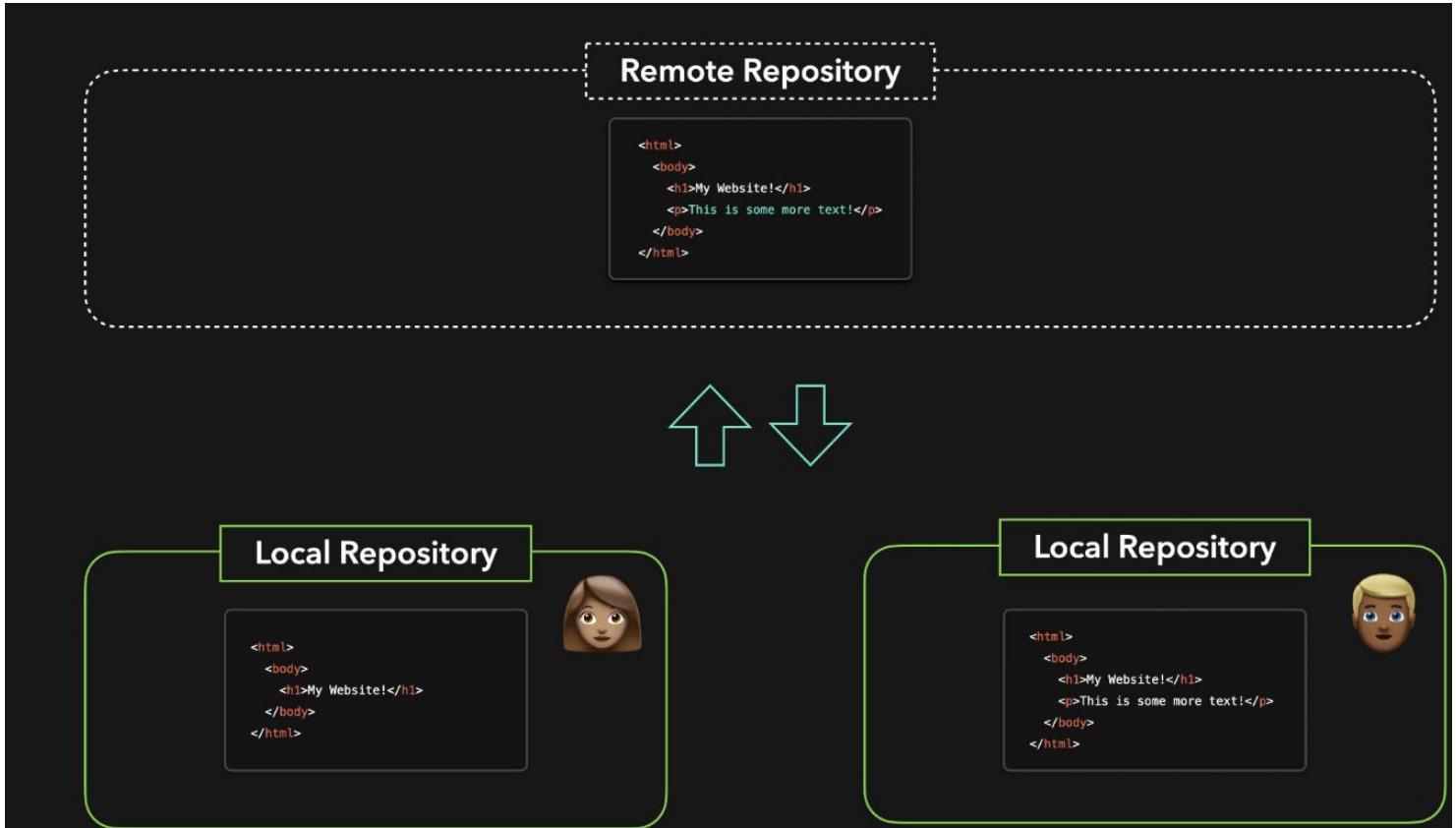
Git Introduction (Cont.)



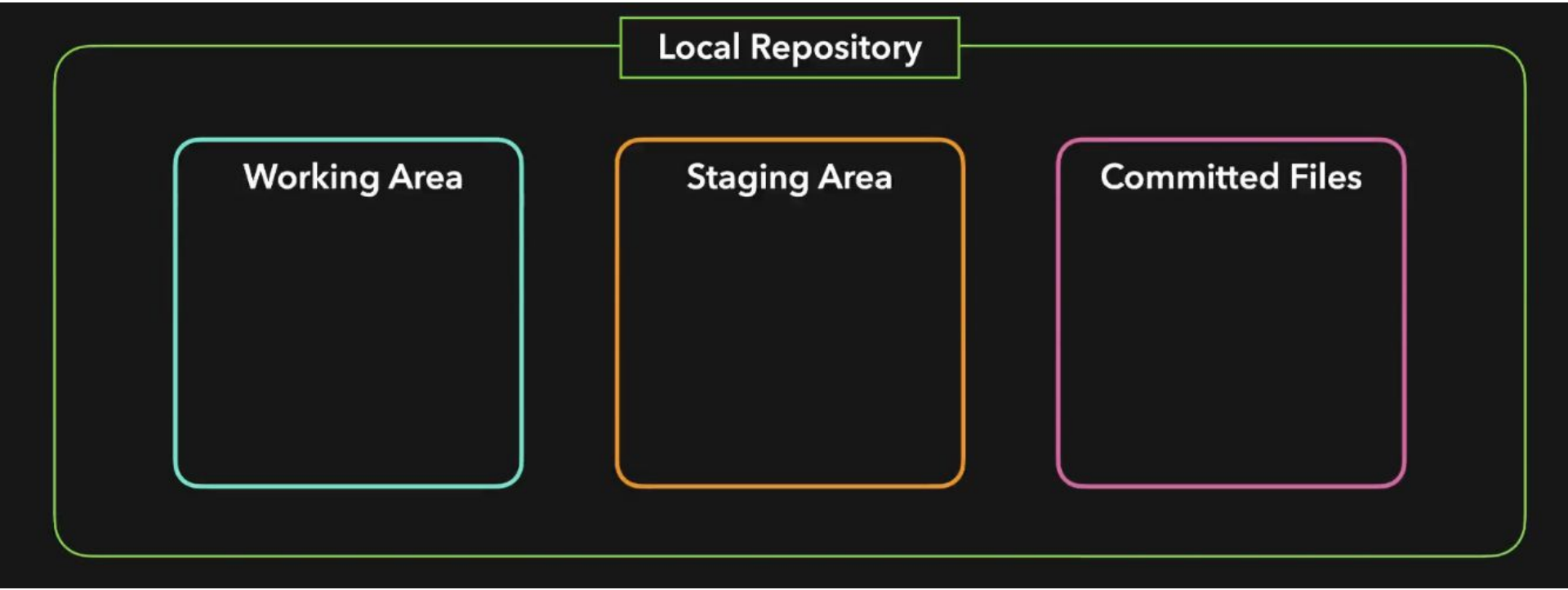
Git Introduction (Cont.)



Git Introduction (Cont.)



Git Introduction (Cont.)



Install git

1. Git packages are available using apt.
2. It's a good idea to make sure you're running the latest version. To do so, Navigate to your command prompt shell and run the following command to make sure everything is up-to-date:

   sudo apt-get update
3. To install Git, run the following command:

   sudo apt-get install git-all
4. Once the command output has completed, you can verify the installation by typing:

   git version

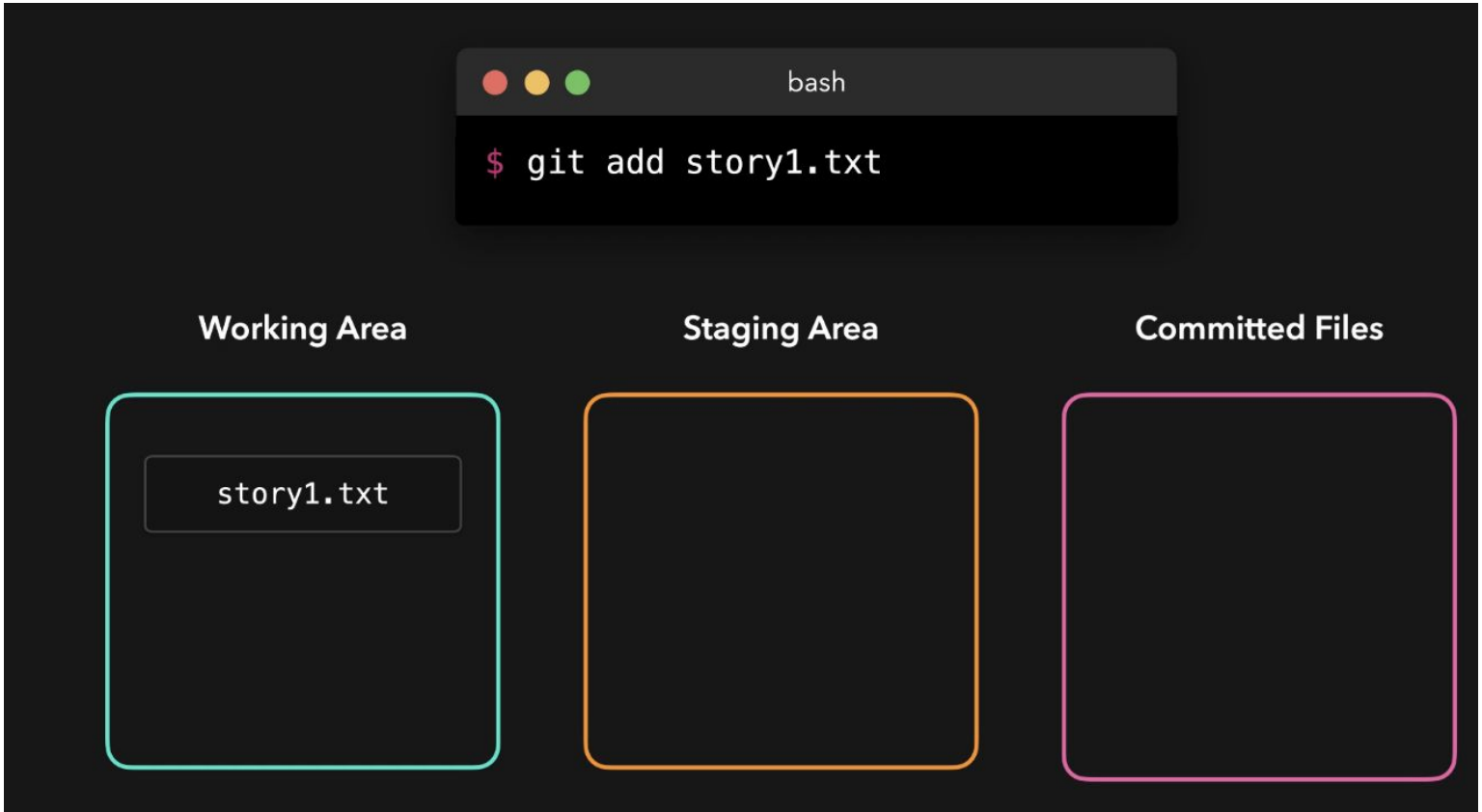
git init

You can do initialize the git in any directory

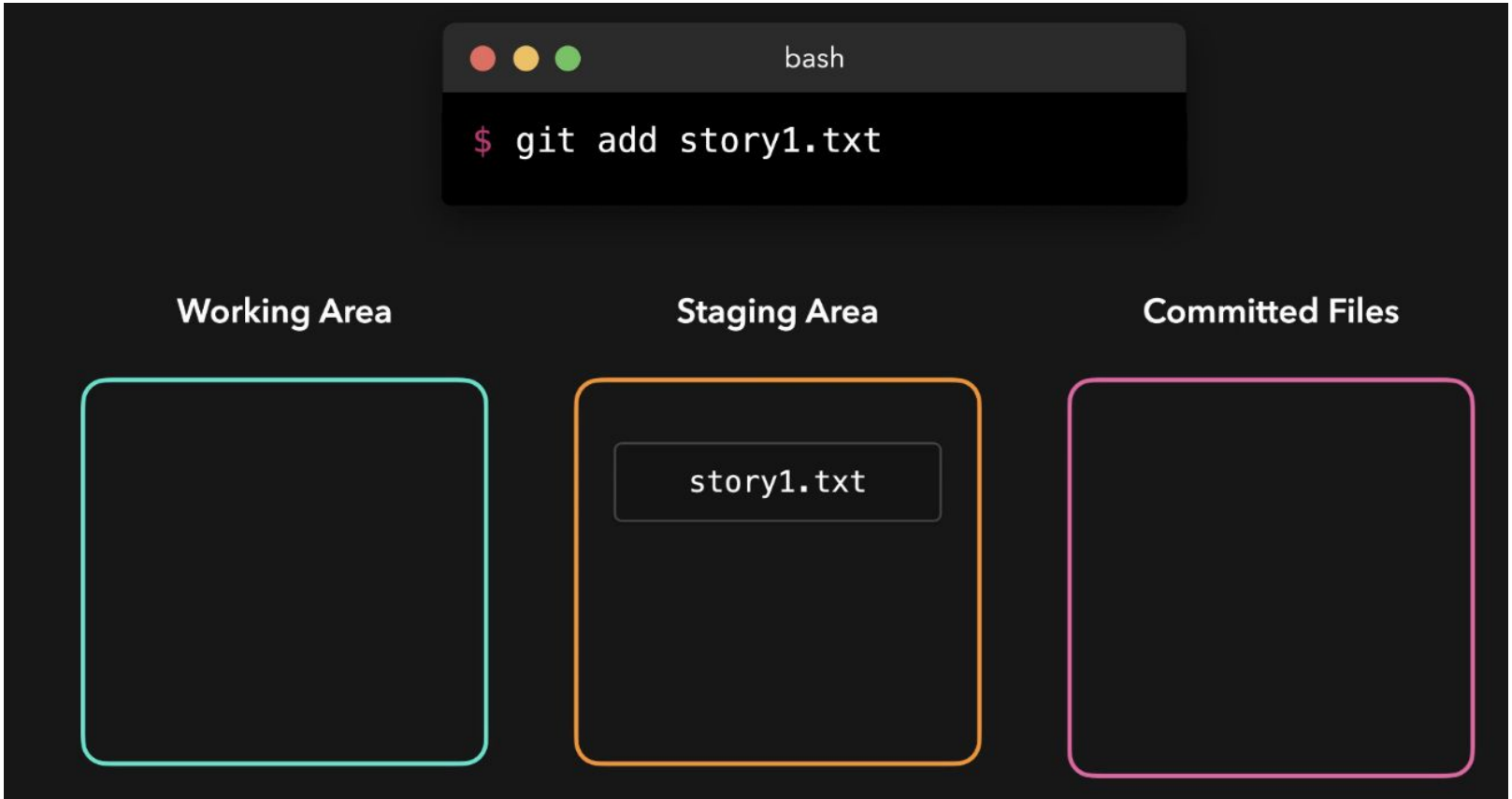
```
bash
$ git init
Initialized empty Git repository in
/Users/lydiahallie/Desktop/myproject/.git/
```

```
bash
$ ls -a
.   ..   .git
```

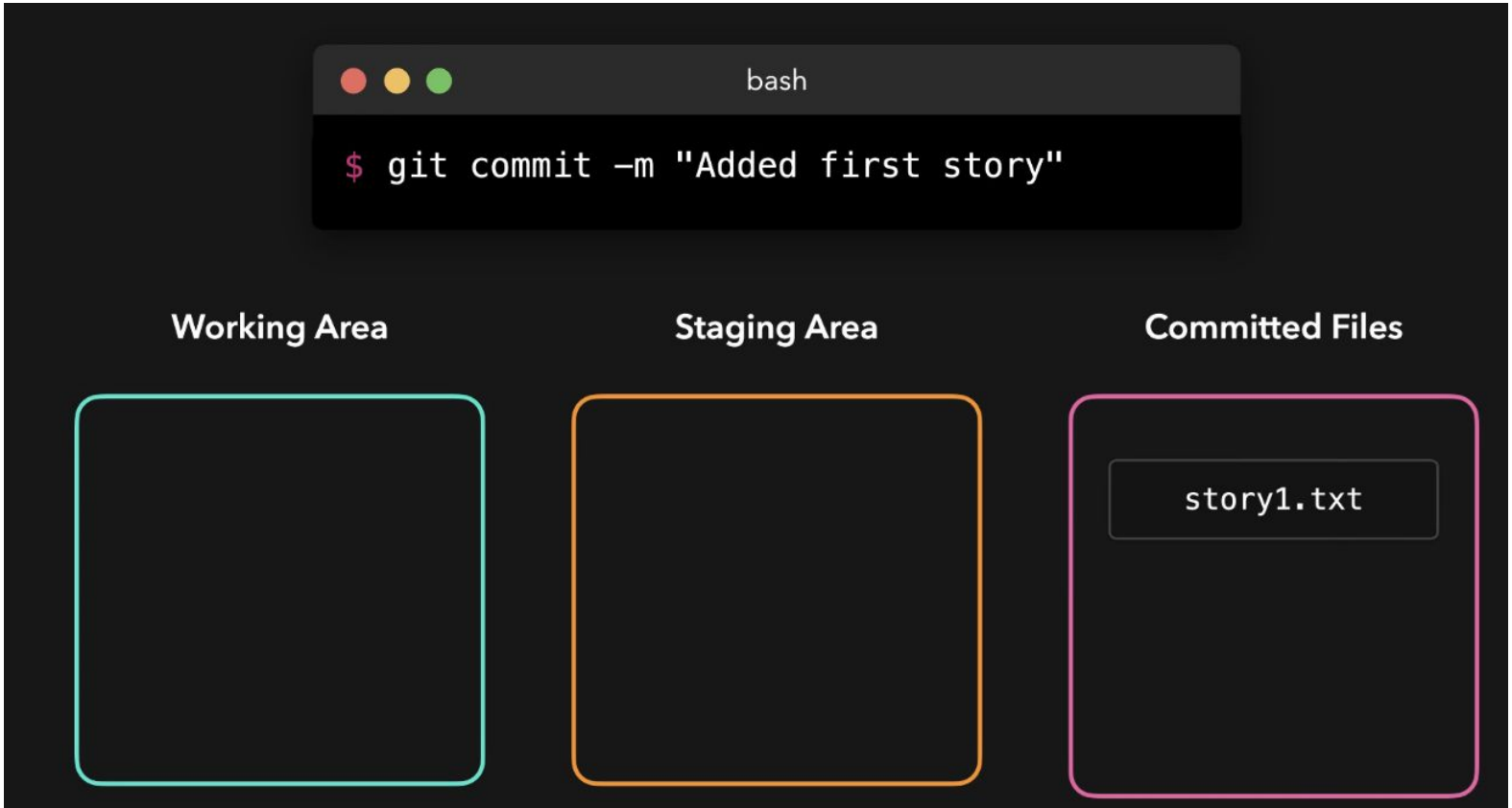
git add


bash
$ git add story1.txt
Working Area
Staging Area
Committed Files
story1.txt

git add

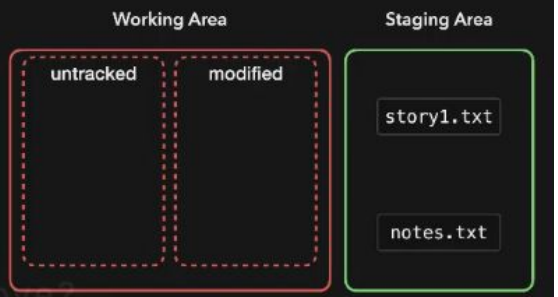

bash
$ git add story1.txt
Working Area
Staging Area
Committed Files
story1.txt

git commit


bash
$ git commit -m "Added first story"
Working Area
Staging Area
Committed Files
story1.txt

git rm

```
sarah (master)$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   notes.txt
        modified:   story1.txt

sarah (master)$
sarah (master)$ git rm
fatal: No pathspec was given. Which files should I remove?
sarah (master)$
sarah (master)$
sarah (master)$ git rm notes.txt
error: the following file has changes staged in the index:
    notes.txt
(use --cached to keep the file, or -f to force removal)
sarah (master)$
```



Git log

```
bash

$ git log
commit 67c833e3…ecb7df62f (HEAD -> master)
Author: John Doe <john@doe>
Date:   Sun Jun 14 14:45:07 2020 -0700

    Added first story
```

Git log (Cont.)

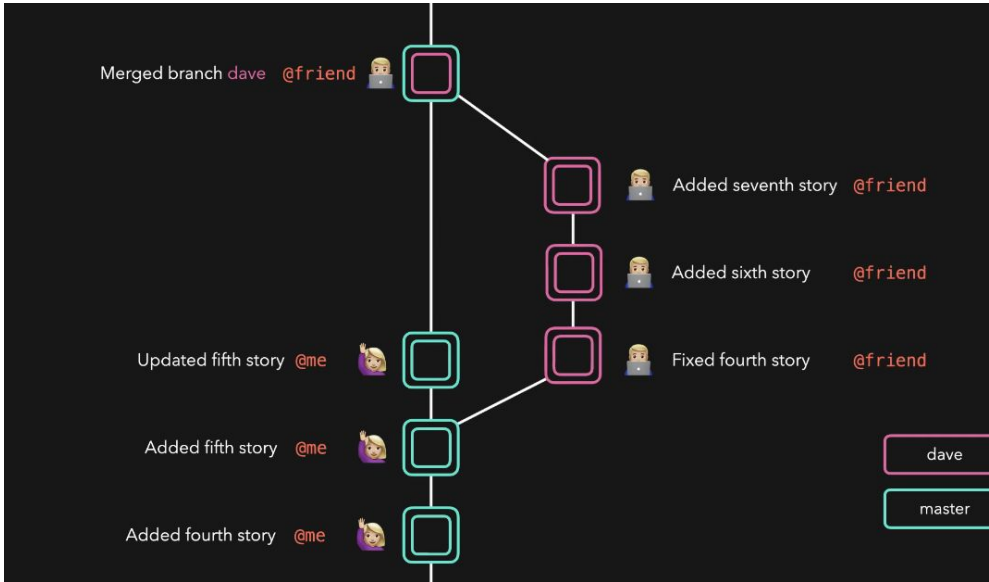
Try:

- git log --name-only
- git log -n 3
- git log --oneline
- git log –graph –decorate

```
* commit 7f7c3368a77b0a907afc02ee82a182cdb474807c (HEAD -> waheed)
  Author: Waheed Iqbal <waheed751@gmail.com>
  Date:   Wed Oct 4 11:43:28 2023 +0500

      wi-story added

* commit 7f68cc04360ba6bd7b8d6e15bc7c7ec3fcd970e1 (master)
  Author: Waheed Iqbal <waheed751@gmail.com>
  Date:   Wed Oct 4 11:41:22 2023 +0500

      stories added

* commit 348772ccf985cb4717b367bf9f23748b1ded5764
  Author: Waheed Iqbal <waheed751@gmail.com>
  Date:   Tue Oct 3 09:19:07 2023 +0500

      updted again

* commit c540012163818c7bc3f377c40f0ca4b427448899
  Author: Waheed Iqbal <waheed751@gmail.com>
  Date:   Tue Oct 3 09:17:45 2023 +0500

      added a new story file
```

Git Branches

- Branches are a fundamental concept that allows you to work on different lines of development within the same repository.
- You can create, switch between, merge, and delete branches to manage and organize your project's development.
- Branches are pointers to specific commit.



Git Branches (Cont.)

```
# Create a new branch
$ git branch sarah
# Switch to an existing branch
$ git checkout sarah

# Create a new branch and Switch to it
$ git checkout -b max

# Delete a branch
$ git branch -d max

# List all branches
$ git branch
```

Fast-Forward Merge

- A fast-forward merge occurs when there is a linear history between the source branch and the target branch.
- It means that there are no new commits on the target branch since the source branch diverged from it.
- When you perform a fast-forward merge, Git simply moves the branch pointer of the target branch to the latest commit of the source branch. This effectively incorporates the changes from the source branch into the target branch.
- Fast-forward merges are typically clean and result in a linear commit history.

```
A - B (master)
     \
      C - D (feature)
```

```
A - B - C - D (master, feature)
```

No Fast-Forward Merge

- A non-fast-forward merge occurs when there have been new commits on the target branch since the source branch diverged.
- This type of merge creates a merge commit that combines the changes from both branches.
- Non-fast-forward merges are useful when you want to preserve the history of both branches and indicate that a merge has occurred.

```
A - B - E (master)
     \
      C - D (feature)
```

```
A - B - E - M (master)
     \   /
      C - D (feature)
```

Git Merging Branches

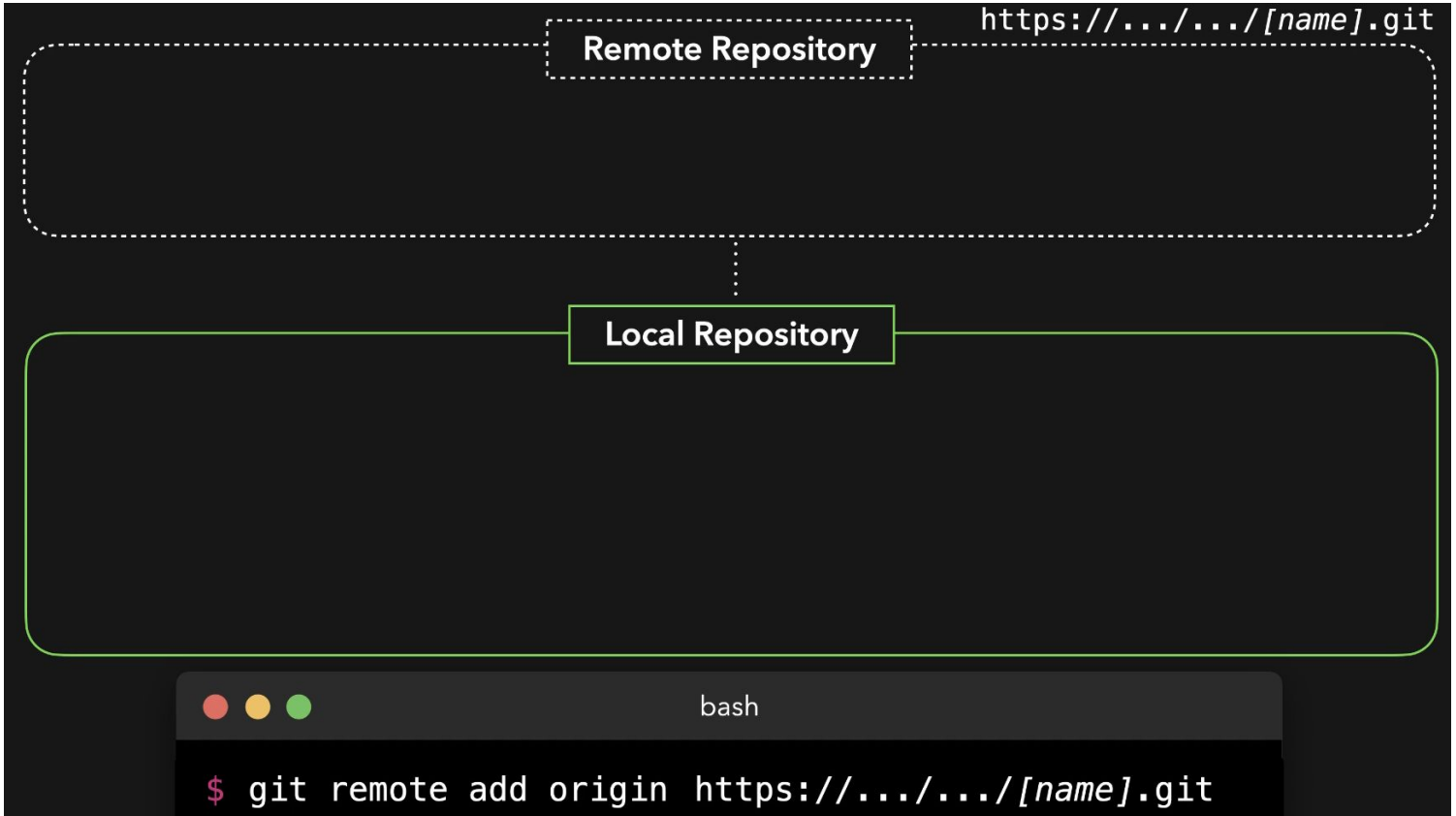
1. Create a new local repository and add a file into it. This will happen in master branch
2. Create a new branch (test) and add add another file in this branch.
3. Check what is different in master branch vs newly created branch.
4. Merge the test branch changes to the master branch:
    a. Checkout master branch
    b. Issue command: git merge test

Remote Repositories

There are many remote git repositories. Some of famous are:

- GitHub
- GitLab
- BitBucket

Remote Repositories (Cont.)

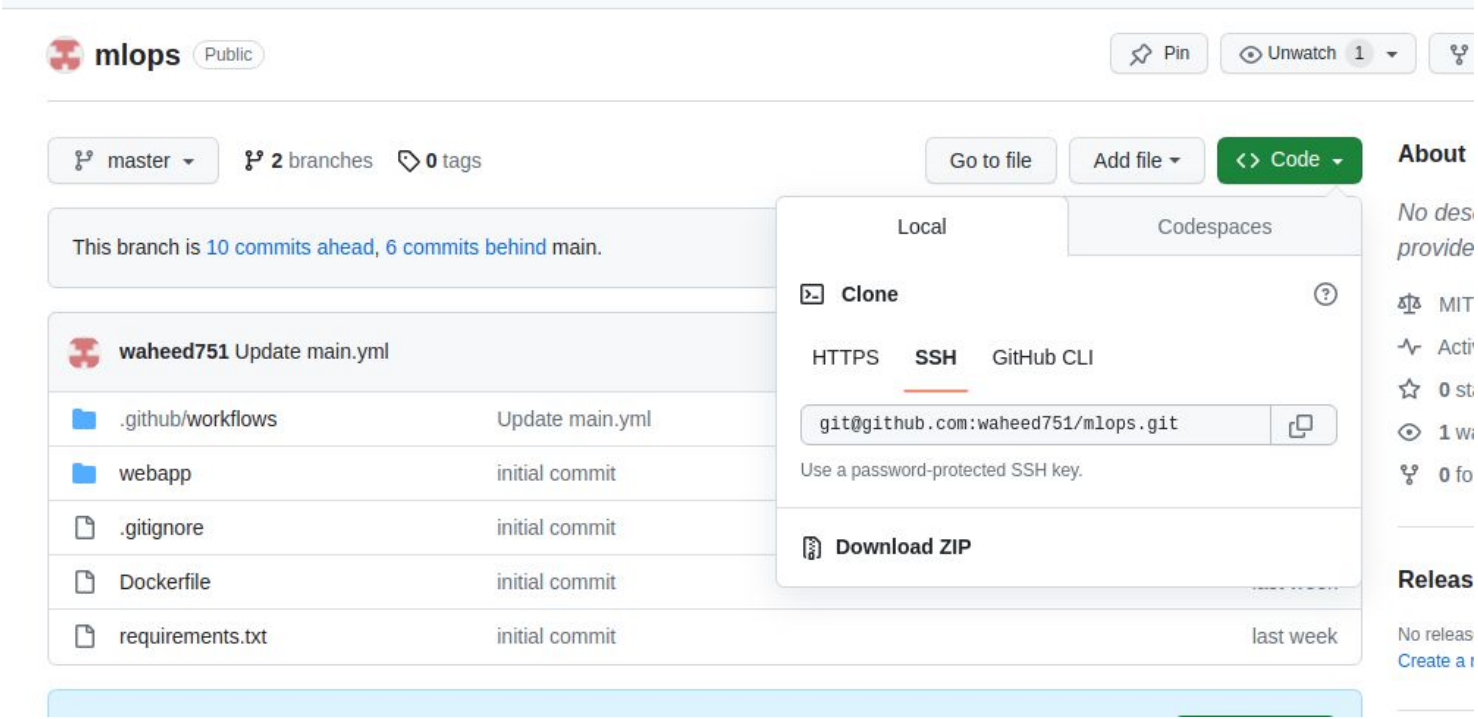


Remote Repositories (Cont.)

```
git remote add origin [REMOTE REPO LINK]

git remote -v

git remote show origin
```

Git clone

git clone [link]



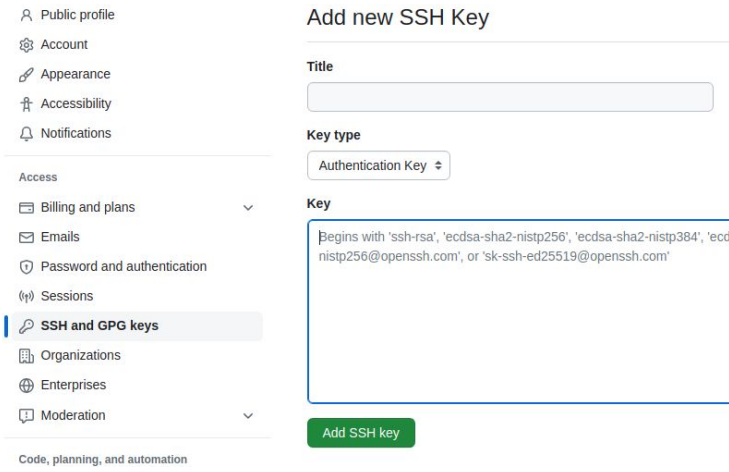
Git clone (Cont.)

bash

```
$ git clone git@github.com:account/remote-repo.git

Cloning into 'remote-repo'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 2948 (delta 28), reused 18 (delta 6)
Receiving objects: 100% (2948/2948), 1.93 MiB | 2.53 MiB/s, done.
Resolving deltas: 100% (1526/1526), done.
```

Create GitHub Account and Add SSH Key

- You can sign up for your github account
- Add ssh key to ensure you can push the commits to the remote repository
- In linux machine, you can create a local key pair using:
  - ssh-keygen -t rsa -f /home/wi/.ssh/gh-wi [replace the path and name as you like]
  - Once you have key pair, copy the content of your public key (which will have .pub extension) and add it to the github as authentication key
  - Add the private key to your terminal session by using: ssh-add [path of the private key file]
  - Now you can use your github repositories locally



Pull Requests

- In Git, a pull request (often abbreviated as "PR") is a feature which propose and manage changes to a repository.
- Pull requests are commonly associated with platforms that offer Git repository hosting services, such as GitHub, GitLab, and Bitbucket.
- Here's an overview of how pull requests work:

  - Creating a Pull Request:
  - A pull request is initiated by a contributor who has made changes in their fork or branch of a repository.
  - The contributor creates a pull request to propose that these changes be merged into the main or target branch of the original repository.

  - Review and Discussion:
  - Once a pull request is created, it can be reviewed by other contributors or team members.

  - Merging the Pull Request:
  - After reviewing and addressing feedback, if the changes are approved and meet the project's quality standards, a project maintainer or collaborator can merge the pull request.
  - Merging combines the changes from the contributor's branch into the target branch (often the main branch).

  - Closing the Pull Request:
  - Once the pull request is merged, it is usually closed, indicating that the proposed changes have been incorporated into the main branch.
  - The closed pull request remains accessible for reference and historical purposes.

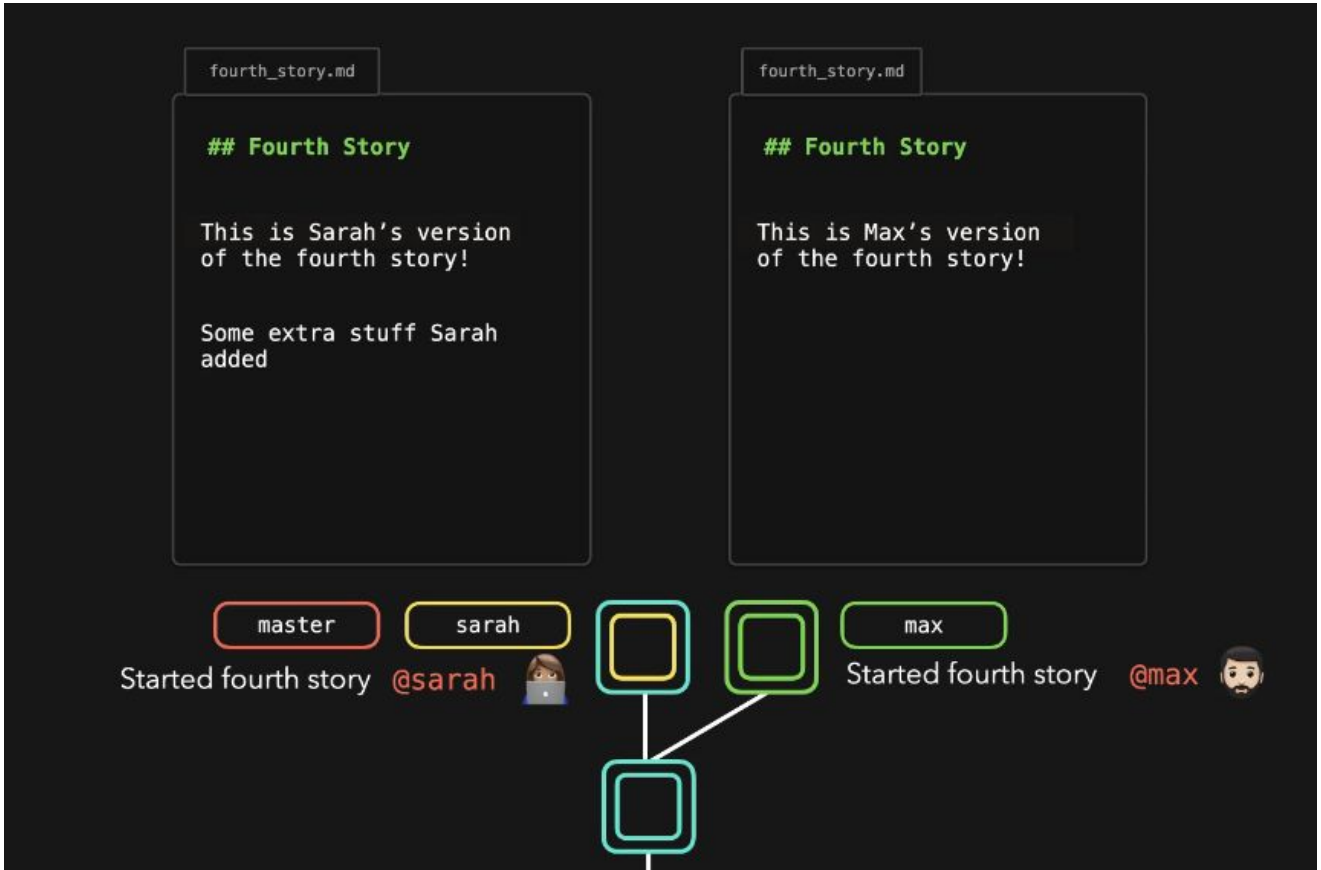
Git fetch and pull

- git fetch fetches changes from a remote without merging them, while git pull fetches and merges the changes into your current branch.
- git pull is often used to update your working branch with the latest changes from the remote repository.
- To perform a pull from the default remote repository (usually "origin") and the currently checked-out branch, use simple git pull.
- To pull changes from a specific remote and branch, you can specify both the remote and branch names:

```
git pull <remote_name> <branch_name>
```

- When you run git pull, Git automatically fetches changes from the specified remote and branch and merges them into your current branch.
- If there are conflicts between your local changes and the remote changes, Git will prompt you to resolve them.

Git Merge Conflicts



Git Merge Conflicts (Cont.)

```
max (master)$ git commit -m "Add index of stories"
[master 06a64b7] Add index of stories
 1 file changed, 4 insertions(+)
 create mode 100644 story-index.txt
max (master)$ git push origin master
To http://git.example.com/sarah/story-blog.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'http://git.example.com/sarah/story-blog.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Git Merge Conflicts (Cont.)

```
max (master)$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
From http://git.example.com/sarah/story-blog
   efd6700..725e2e3  master     -> origin/master
hint: Pulling without specifying how to reconcile divergent branches is
hint: discouraged. You can squelch this message by running one of the following
hint: commands sometime before your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
CONFLICT (add/add): Merge conflict in story-index.txt
Auto-merging story-index.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
max (master)$ cat story-index.txt
<<<<<<< HEAD
1. The Lion and the Mooose
2. The Frogs and the Ox
3. The Fox and the Grapes
4. The Donkey and the Dog
=======
1. The Lion and the Mouse
2. The Frogs and the Ox
3. The Fox and the Grapes
>>>>>>> 725e2e33f0b243af785f784fa5e
max (master)$
```