Table of contents

Installation of Git

Download the Git binaries for your specific machine

- **Windows**
  - Download Git on Windows https://git-scm.com/download/win
- **Linux**
  - Debian - Ubuntu
    - Execute the below command for Ubuntu Git Installation. `sudo apt install git-all`
- **Mac**
  - Installing git for Mac from : https://git-scm.com/download/mac

> Execute all below commands in `Git Bash` shell . Install git on respective OS from https://git-scm.com

- After installation, check the version of the git run the below command

```
git --version
# git version 2.31.1.windows.1
```

> Execute all below commands in `Git Bash` shell

Git Local Operations

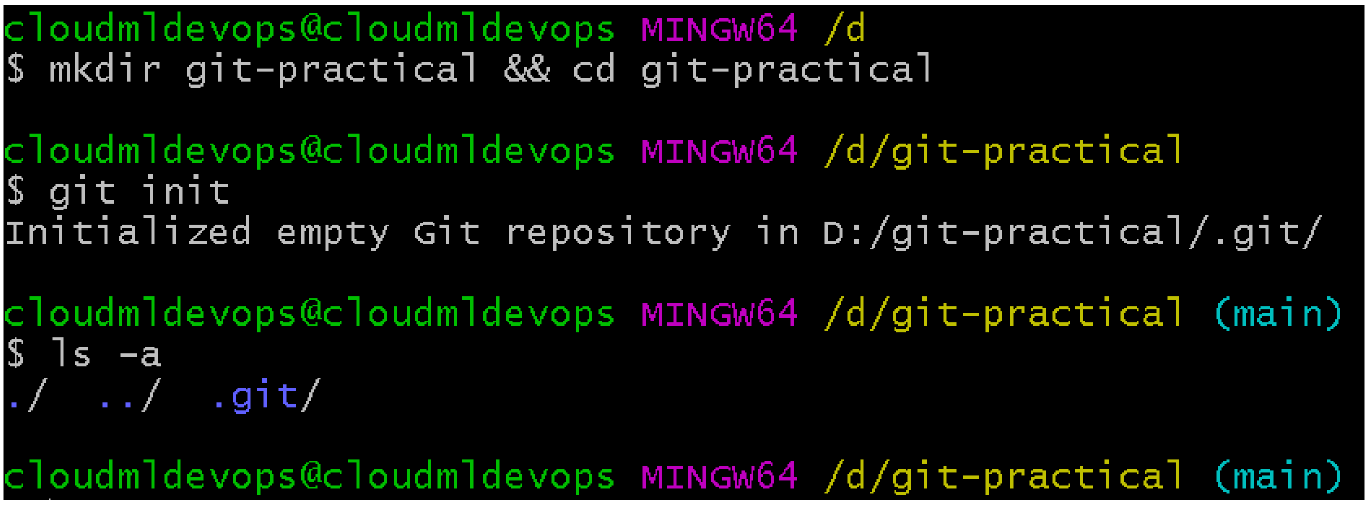
Create Local Repository, adding files and commit changes

Initialize a Git Repository

- Run `git --version` to check git version.
- Start a new repository. Check Your git Settings
- Create a new Project Folder and initialize git inside it.

```
git --version
mkdir git-practical && cd git-practical
git init
ls -al  # List all the files recursively
```

- `main/master` -> default branch name.



```
cloudmldevops@cloudmldevops MINGW64 /d
$ mkdir git-practical && cd git-practical

cloudmldevops@cloudmldevops MINGW64 /d/git-practical
$ git init
Initialized empty Git repository in D:/git-practical/.git/

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ ls -a
./  ../  .git/

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
```

- `.git` directory -> We will not modify anything inside this path.

- The `git init` command creates an empty Git repository - basically a `.git` directory with subdirectories for `objects, refs/heads, refs/tags, and template files`.

- Git configuration can be `global` and `local`.

- For local git repo config, use:

```
git config --local user.name "TestUser"
git config --local user.email "testuser@example.com"
git config --list
```

- This will result in [user] section added to .git/config file:

```
cat .git/config

[user]
name = Yourname
email = name@example.com
```

- You can use global config also if you only have one git server.

```
git config --global user.email "testuser@example.com"
git config --global user.name "TestUser"
```

- Then the [user] section will be present at `~/.gitconfig`, with the same content as in the `.git/config` file.
- To get the specific config value that is currently set

```
git config user.name
git config user.email
```

Working with Git Local Repo

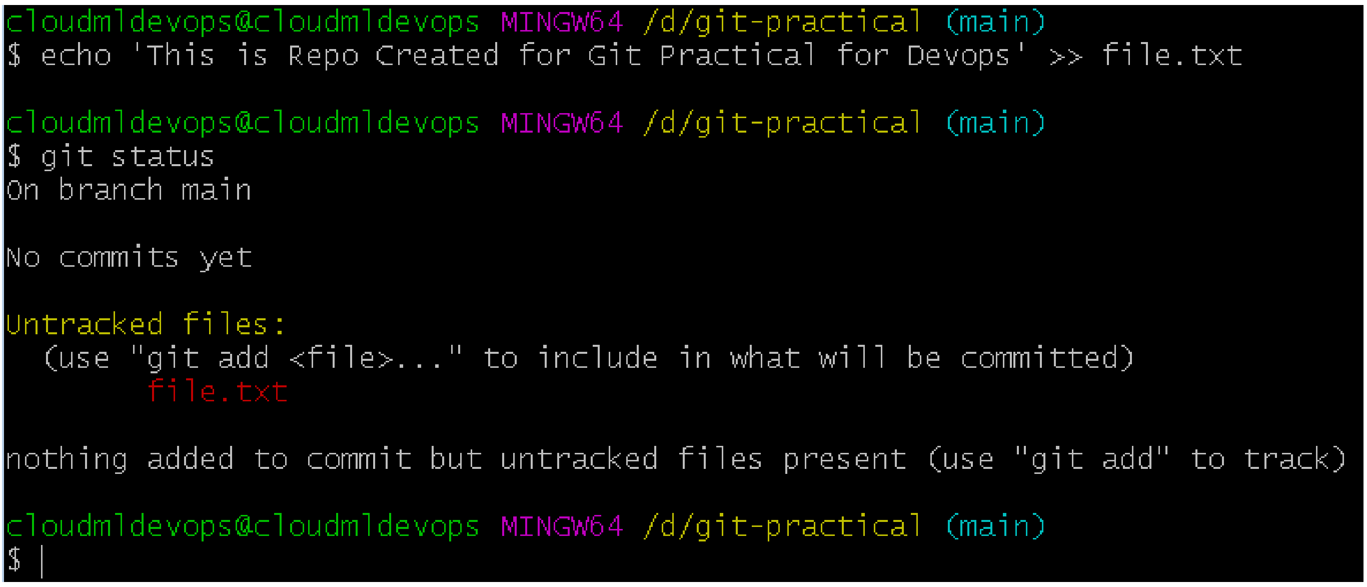
- Lets add or update files i.e code in this repository directory

```
ls -al
echo 'This is Repo Created for Git Practical for Devops' >> file.txt
```

- To see the status of git

```
git status
```

- Default branch will be `main` branch.
- `Untracked files` are files that are not added to the git.



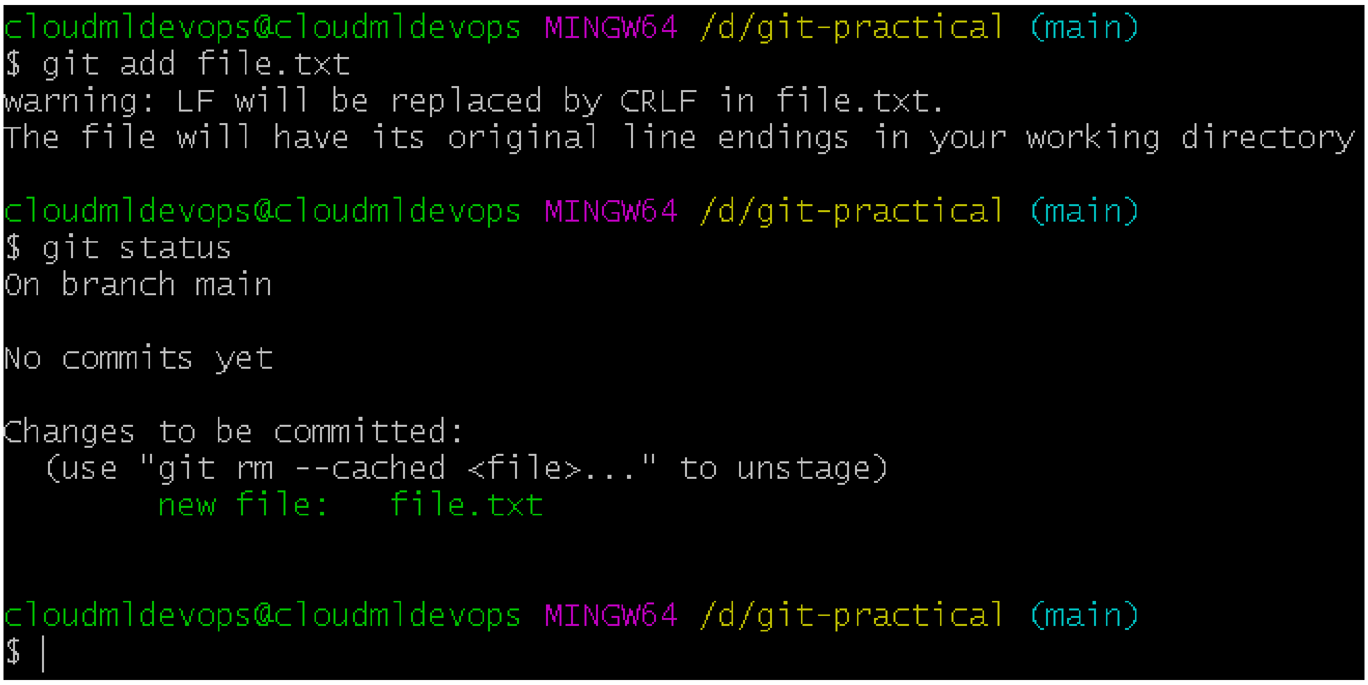
```
cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ echo 'This is Repo Created for Git Practical for Devops' >> file.txt

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git status
On branch main

No commits yet

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

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

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ |
```

git add

- The `git add` command adds new or changed files in your working directory to the Git `staging area`.

- As you're working, you change and save a file, or multiple files. Then, before you commit, you must `git add`.

```
git add file.txt
git status
```

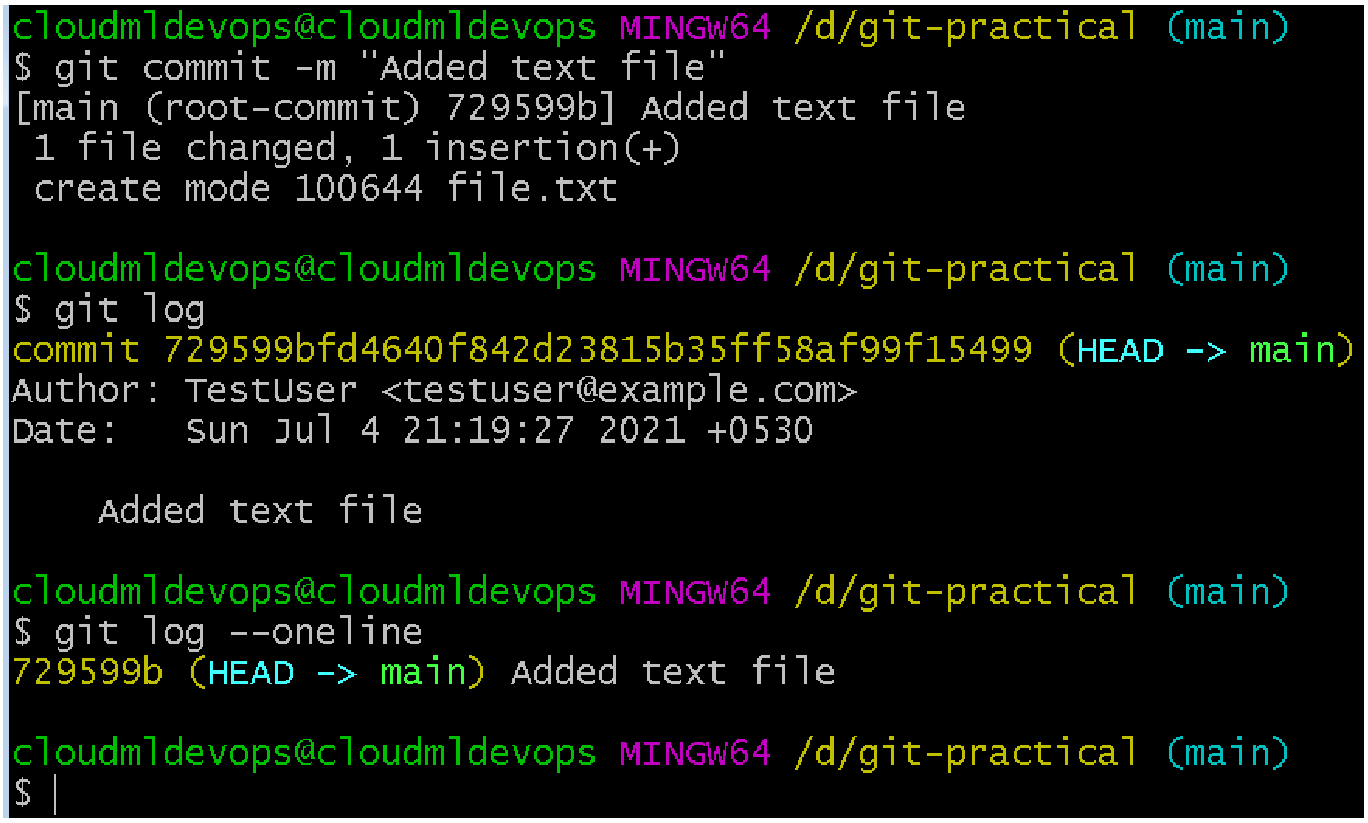


```
cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git add file.txt
warning: LF will be replaced by CRLF in file.txt.
The file will have its original line endings in your working directory

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file.txt


cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ |
```

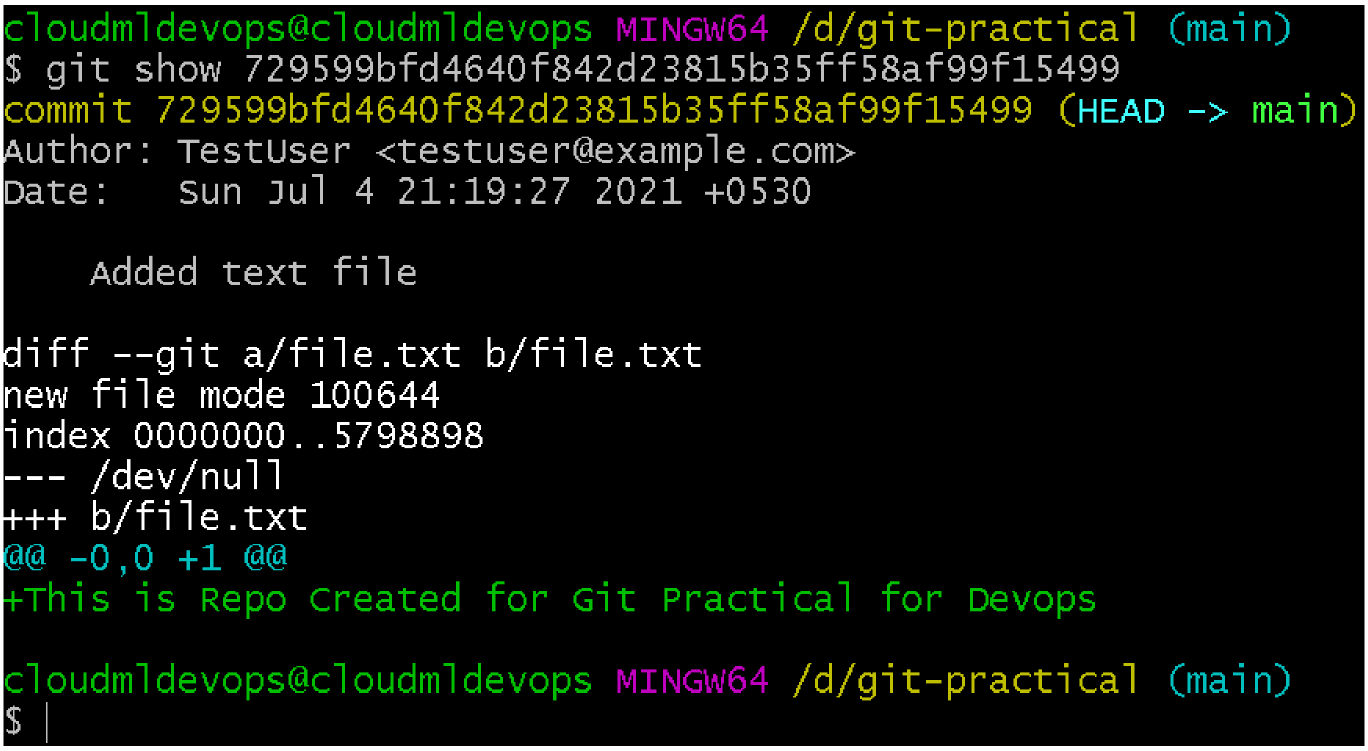
git commit | log | show

- `git commit` creates a commit, which is like a snapshot of your repository. These commits are snapshots of your entire repository at specific times. Commits include metadata in addition to the contents and message, like the `author, timestamp`
- To commit a change, there should be a `commit message` provided.
- Use below command to commit the changes from staging area into the repository.

```
git commit -m "Added text file"
git log
git log --oneline
git show 729599bfd4640f842d23815b35ff58af99f15499
```



```
cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git commit -m "Added text file"
[main (root-commit) 729599b] Added text file
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git log
commit 729599bfd4640f842d23815b35ff58af99f15499 (HEAD -> main)
Author: TestUser <testuser@example.com>
Date:   Sun Jul 4 21:19:27 2021 +0530

    Added text file

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git log --oneline
729599b (HEAD -> main) Added text file

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ |
```



```
cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ git show 729599bfd4640f842d23815b35ff58af99f15499
commit 729599bfd4640f842d23815b35ff58af99f15499 (HEAD -> main)
Author: TestUser <testuser@example.com>
Date:   Sun Jul 4 21:19:27 2021 +0530

    Added text file

diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..5798898
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+This is Repo Created for Git Practical for Devops

cloudmldevops@cloudmldevops MINGW64 /d/git-practical (main)
$ |
```

- Make some more changes in the file:

```
# Append some more
echo 'This is content written after 1st Commit' >> file.txt
git status
# Changes not staged for commit => File is changed in working area
git add file.txt
git diff --staged
# Changes to be committed: => File is in Staging Area and can be committed
```

```
git commit -m "added changes to same file for new commit"
git show <COMMIT_ID>
```

- `Changes not staged for commit` -> File is Working tree
- `Changes to be committed` -> File is in Staging area
- If you edit a file that is already staged, it will appear in `"Changes to be committed:"` and `"Changes not staged for commit:"`
- Viewing Your Staged and Unstaged Changes
- This command compares your staged changes to your last commit:

```
git diff --staged
```

- make some changes in the file that is already staged, below command will show the differences

```
git diff
```

- Make a commit
- Viewing the Commit History and view commit details in one line

```
git log
git log --oneline
```

- To view the only last two entries

```
git log -p -2
```

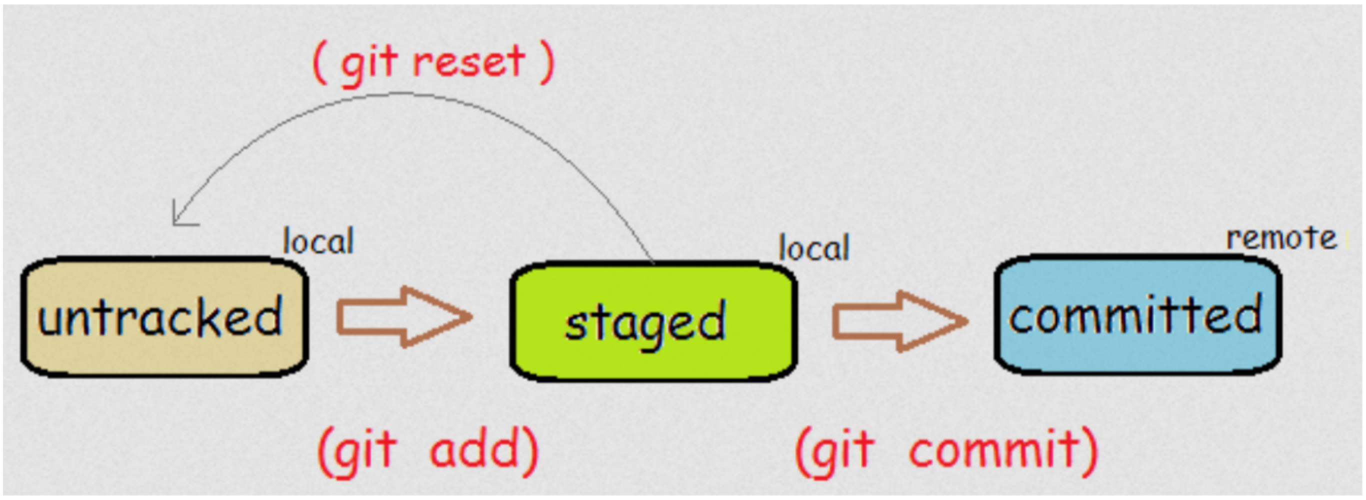
- More common options for `git log`

```
git log --stat
git log --pretty=oneline
```

- To change the commit message of an existing commit

```
echo "this is testing file" >> newfile.txt
git add newfile.txt
git commit -m 'newfile123.txt commit'
git log -p -2
git commit --amend -m "newfile.txt commit"
git log -p -2
```

- Unstaging a Staged File



```
echo "adding some content to unstage changes" >> newfile.txt
# add all files in working tree into staging area
git add * OR git add .
git reset newfile.txt
git commit -m "added new change"
# git will never commit any change directly from working tree
# working tree -> (git add) -> Staging Area -> (git commit) -> File is Commit in
repo
```

Github Account Remote Repository.

Create a Remote Repository

- Sign Up into www.github.com and verify email id.
- Create a New empty repository in Github using browser.
- Navigate to a directory in local using `Git Bash` that is not a git directory.

git clone

- The `git clone` command is used to create a copy of a specific repository or branch within a repository.
- When you clone a repository in Git, you don't get one file, you get the entire repository - `all files, all branches, and all commits`.

```
git clone <REMOTE_GIT_SSH_URL>
OR
git clone <REMOTE_GIT_HTTPS_URL>
```

- The git remote command lists the names of all the remote repositories and the -v parameter (verbose) will display the full URL of the remote repository for each remote name listed.

```
git remote -v
```

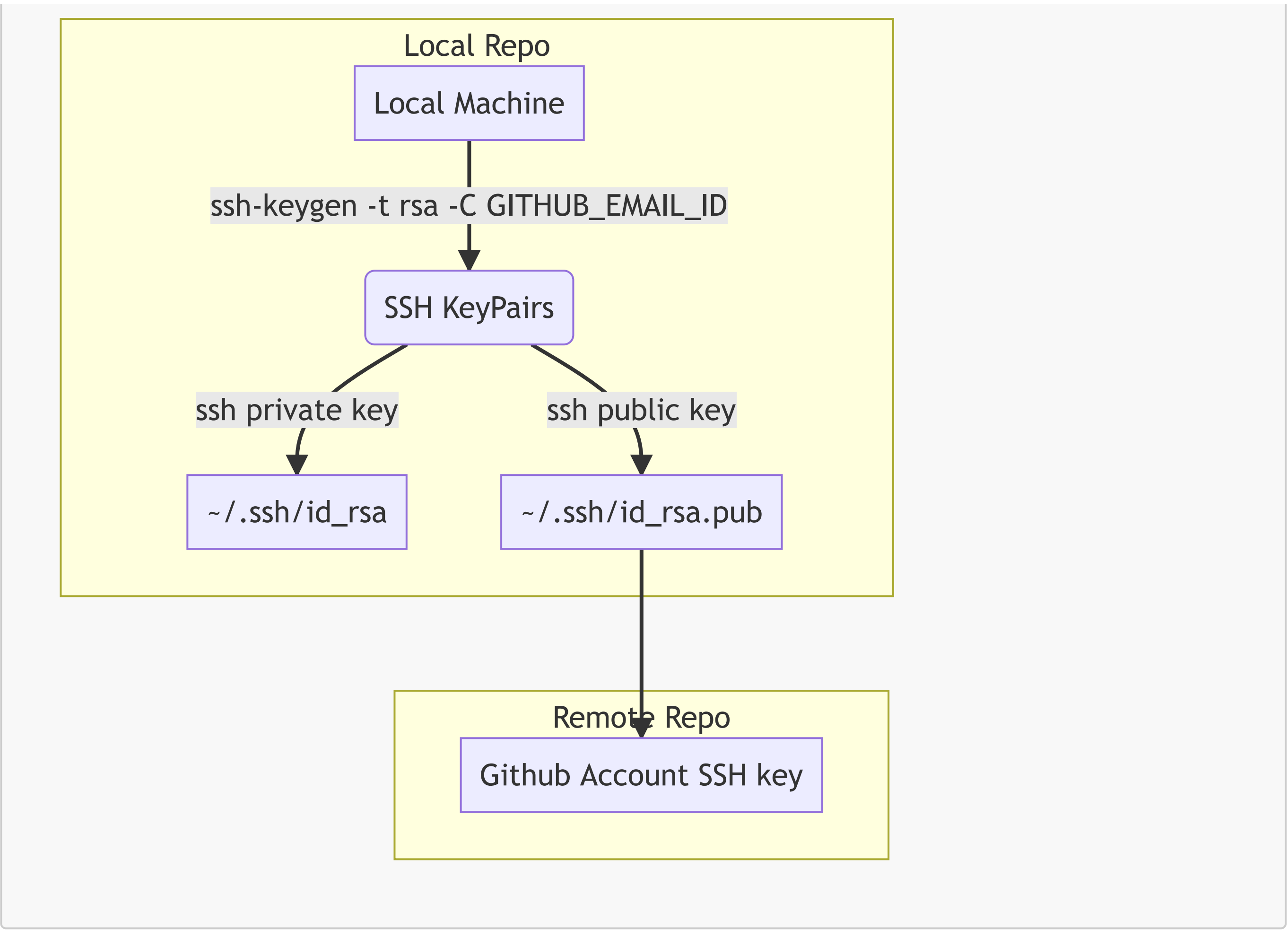
- To have multiple commits created, add or modify some files -> commit and push it to GitHub.

```
git config --local user.name "TestUser"
git config --local user.email "testuser@example.com"
echo "this is file created in local repo" >> file.txt
git add <FILENAME>
git status
git commit -m "Message for the commit"
git log
```

- Git Clone URL are of two types :
  - https: Requires username and password
  - ssh : Requires ssh private key and public key
- Connection to your GitHub Account using SSH
- Generating an SSH Key, Use the github sign-in email address in the below command.

```
ssh-keygen -t rsa -C "<GITHUB_EMAIL_ID>@gmail.com"
ssh-keygen -t ed25519 -C "<GITHUB_EMAIL_ID>@gmail.com"

Generating public/private ed25519 key pair.

# create id_ed25519 ( Private Key) and a id_ed25519.pub ( Public)
# private key -> ~/.ssh/id_ed25519
# public key -> ~/.ssh/id_ed25519.pub

cat ~/.ssh/id_ed25519
cat ~/.ssh/id_ed25519.pub
```

- Above command will create a Public (`~/.ssh/id_ed25519.pub`) and Private Key(`~/.ssh/id_ed25519`) Pair.
- Add the Public Key file content into your Github Account Settings under: `Settings` > `SSH and GPG keys.` > `New SSH key` > `Paste the Public Key Content` > `Save`



- Verify SSH authentication

```
ssh -T git@github.com
# Hi <GITHUB_USER> You've successfully authenticated, but GitHub does not provide
shell access.
OR
ssh -i <PRIVATE_KEY_PATH> -T git@github.com
# Hi GITHUB_USERNAME! You've successfully authenticated, but GitHub does not
provide shell access.
```

- You should get above similar message if connection to github account is successful using SSH

git push

- `git push` uploads all local branch commits to the corresponding remote branch.
- First time push command use below `-u` parameter

```
git push -u origin master

git diff origin/main..main
```

- To Push changes from local Repo to Remote repo for main branch using ssh, add a SSH remote origin URL for Local Repository.

> To change the remote URL click here

- Using git remote add command allows us to associate a remote repository. Normally, you want to paste in the full URL for the remote repository given to you by your Git host (GitHub). By convention, the first or primary remote repository is named origin.

```
git remote add origin git@github.com:<GITHUB_USERNAME>/git-practical.git
```

- If there are any changes made in the Remote Repository, to have those changes present in Local Repository, use below command:

```
git pull origin master
```

- Push changes in GitHub

```
git push origin master
```

- To get or display the content of the file as per particular commit.

```
git show e4b71efa7f76c0fc0875e0562d5fb6d7dadbff9c:newfile.txt
```

Reference information

Changing a git remote URL

1. Switching remote URLs from SSH to HTTPS

- List your existing remotes in order to get the name of the remote you want to change.

```
git remote -v
origin  git@hostname:USERNAME/REPOSITORY.git (fetch)
origin  git@hostname:USERNAME/REPOSITORY.git (push)
```

- Change your remote's URL from SSH to HTTPS with the git remote set-url command.

```
git remote set-url origin https://hostname/USERNAME/REPOSITORY.git
```

- Use below command to clone the Repository with SSH URL.

```
git clone git@github.com:<GITHUB_USERNAME>/<REPO_NAME>.git
git remote -v
```

- The next time you `git pull, or git push` to the remote repository, you'll be asked for your GitHub username and password.

2. Switching remote URLs from HTTPS to SSH

- Change your remote's URL from HTTPS to SSH with the git remote set-url command.

```
git remote set-url origin git@github.com:USERNAME/REPOSITORY.git
```

- Send Changes to Remote

```
git push -u remote-name branch-name

git push remote-name branch-name
```

- The git push sends all your local changes (commits) on branch `branch-name` to the remote named `remote-name`.
- The `-u` parameter is needed the first time you push a branch to the remote.
- Receive Changes from Remote

```
git pull remote-name branch-name
```

- The git pull receives all your remote changes (commits) from the remote named remote-name and on branch branch-name.

> Getting help from git

```
git help <verb>
git help config
git add -h
```

Related Terms

- `git status`: Always a good idea, this command shows you what branch you're on, what files are in the working or staging directory, and any other important information.
- `git commit -m "descriptive message"`: Records file snapshots permanently in version history.
- `git push`: Uploads all local branch commits to the remote.

Reference:

- Download Github for Desktop from https://desktop.github.com/