DevOps
Git Version Control

Trainer: Abhijith V G – AWS, eCommerce, Mobile & DevOps Architect

GIT: Introduction: Version Control Systems

Why?

1. Easily Track Changes
2. Consistent File and Folder Names! – No need to rename files/folders
3. Easily Do/UnDo Changes – Ability to undo many files/folders together
4. Use as a Communication Tool – Add comments to changes
5. Accountability – Track who made what changes
6. States or Checkpoints – Branches and Tags for production, staging etc.

GIT: Introduction: Local, Centralized and distributed version controls

1. **Local (No server)**
   A single user manages the changes locally. Though such a single user local version control does not really exist, Git which is a distributed version control works perfectly well as a local isolated version control system.

2. **Centralized (Easier to understand, Controlled Access, Older hence GUI)**
   The main concept of a centralized system is that it works in a client and server relationship. Eg: SubVersion

3. **Distributed (No server, Faster, Reliable)**
   Distributed systems are a newer option. In distributed version control, each user has their own copy of the entire repository, not just the files but the history as well.
   Eg: Git, Mercurial

GIT: The Inventor

Linus Torvalds

The Legacy of Linus Torvalds: Linux and Git.

1. Linux, which now runs vast swathes of the internet, including Google and Facebook.

2. Git, software that's now used by developers across the net to build new applications of all kinds.

GIT: Installation on Linux

1. **Debian/Ubuntu**

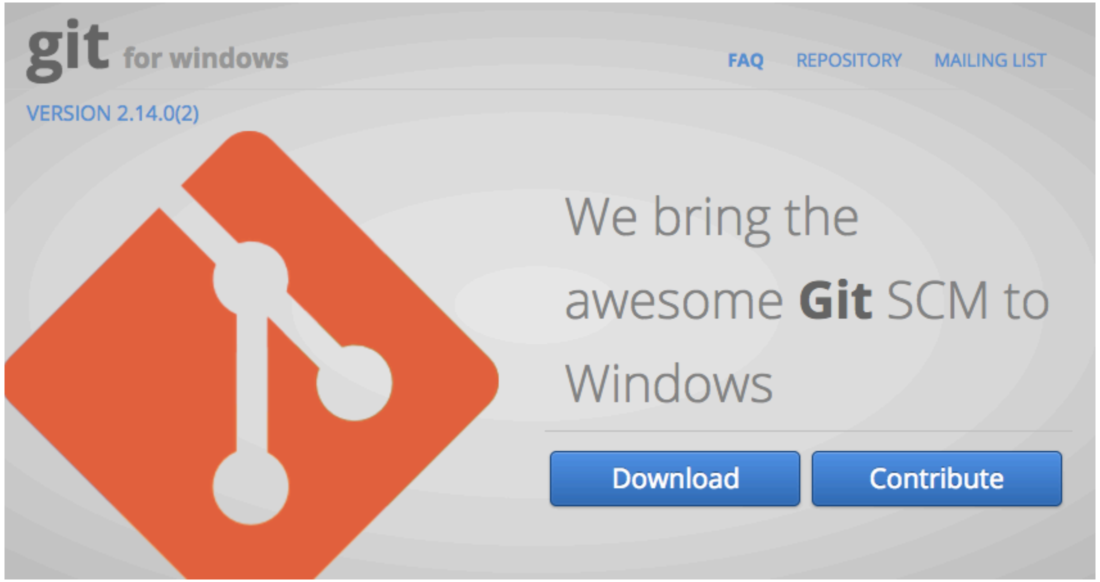
```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git
```

2. **RedHat/Centos**

```
sudo yum upgrade
sudo yum install git
```

GIT: Installation on Windows

1. https://git-for-windows.github.io



GIT: Installation: Initial Setup

1. **Your Identity**
   git config --global user.name ”YourName”
   git config --global user.email youremail@domain.com

2. **Your Editor**
   git config --global core.editor notepad
   git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession”

3. **Checking**
   git config --list or git config user.name

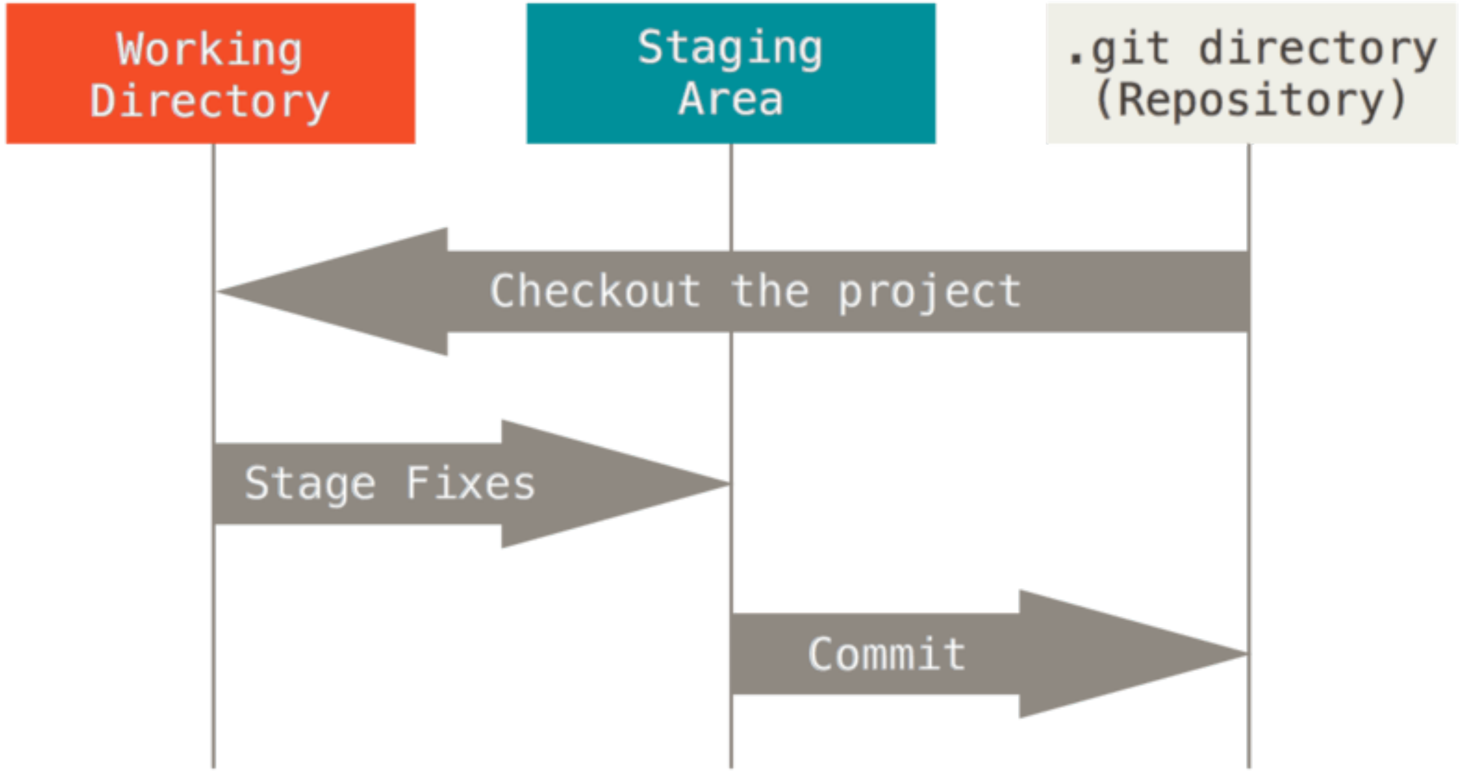
GIT: Essentials: Creating repository

Create or CD into a folder to be converted to a Git Repo(sitory)

```
git init
```

This creates a new hidden folder .git that contains all of your necessary repository files.
(At this point, nothing in your project is tracked yet)

GIT: Essentials: Internals



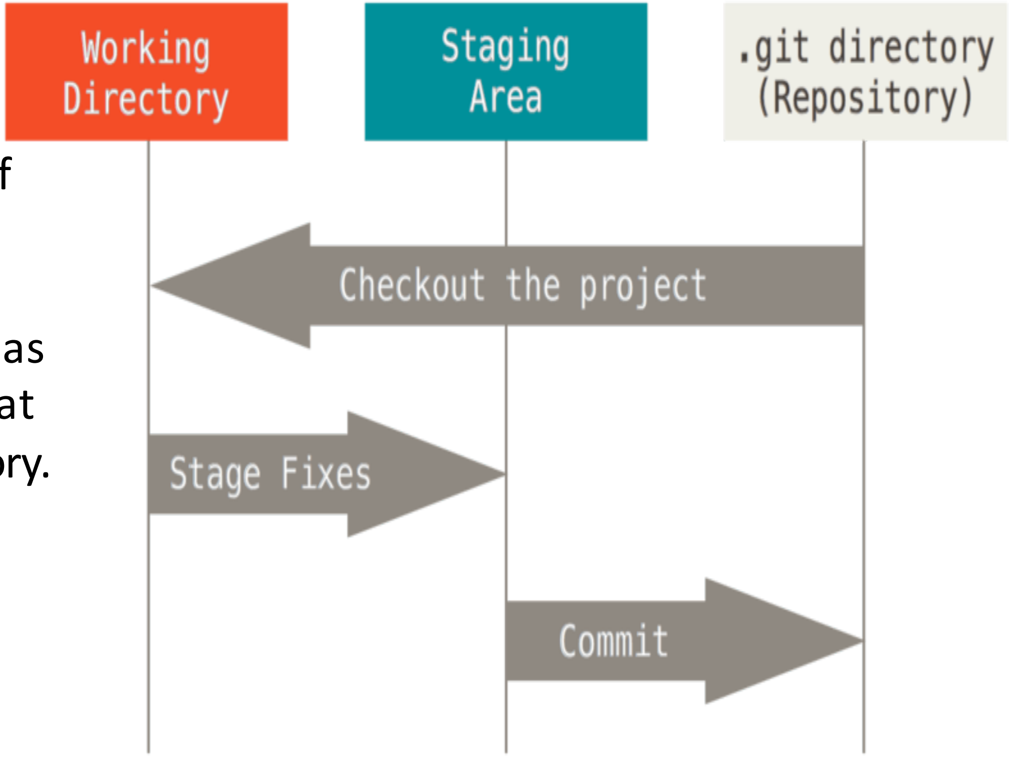
GIT: Essentials: Internals

1. You modify files in your working tree.

2. You stage the files, adding snapshots of them to your staging area.

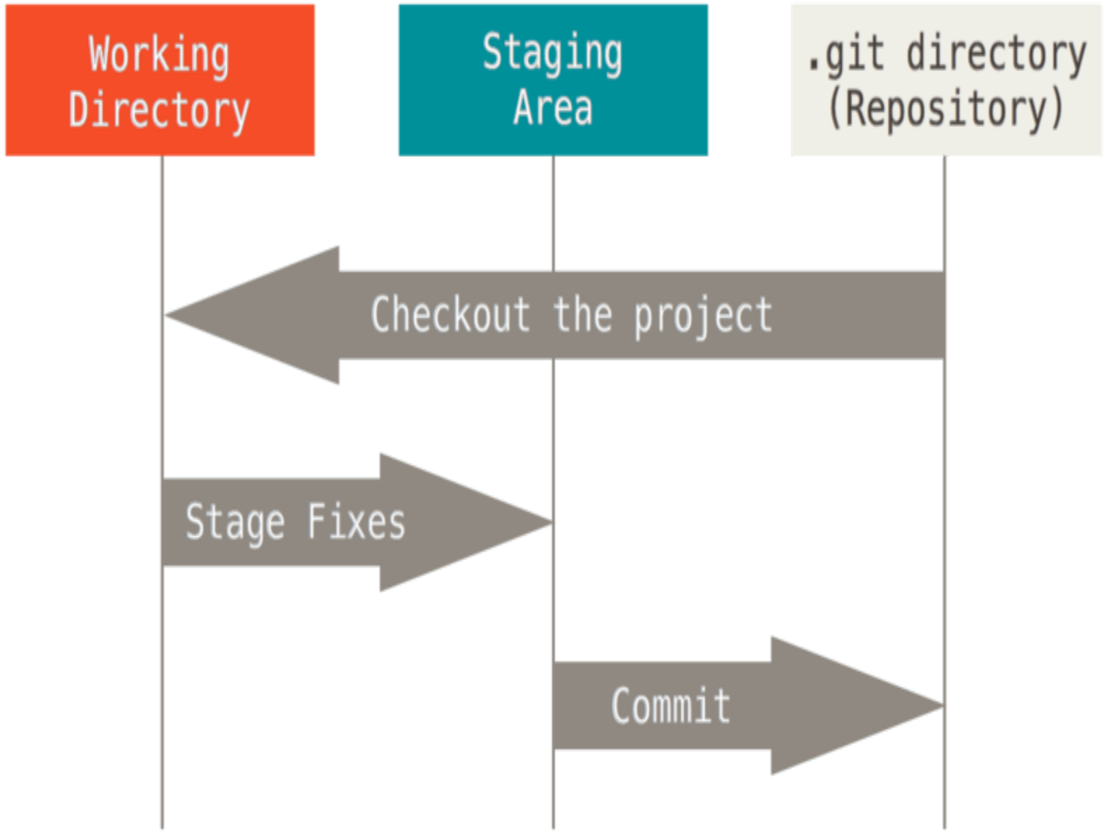
3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

A TRACKED file can be in 1.staged, 2. committed and 3. modified states.
All other files are in un-tracked state.



GIT: Essentials: Begin Tracking files

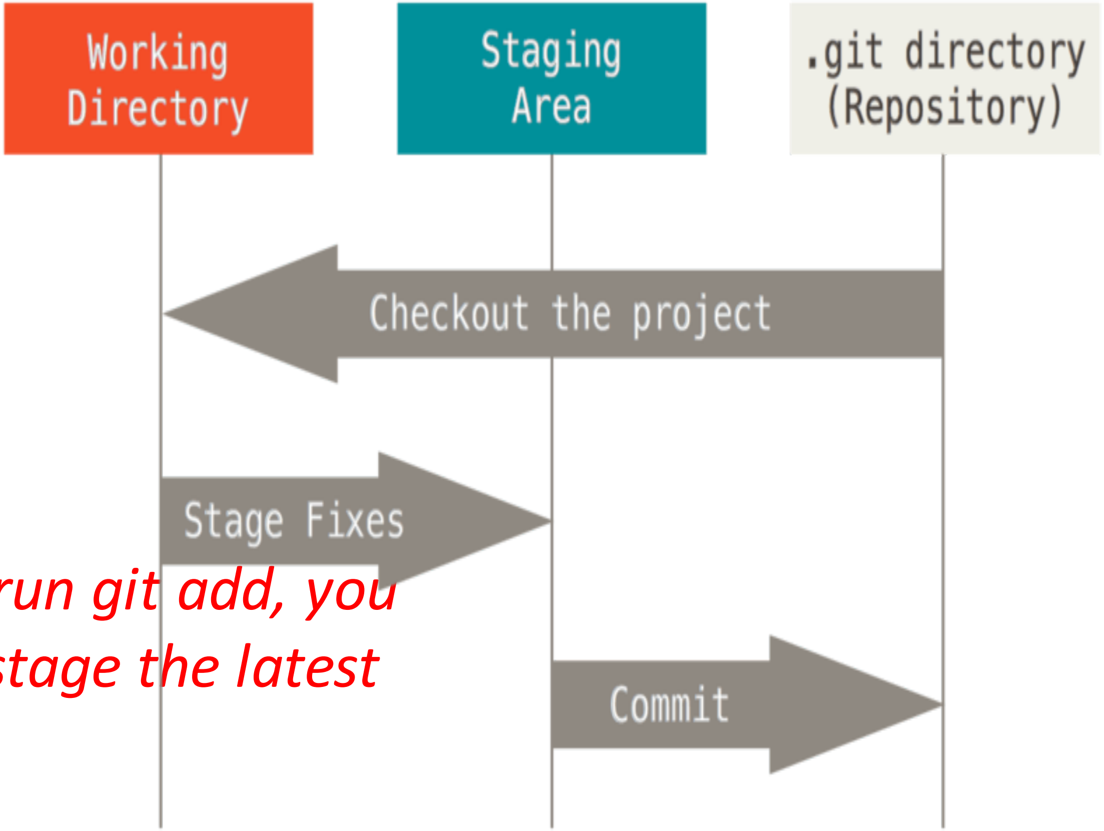
```
git status

git add *.txt

git add -A
```



GIT: Essentials: Check-in (Staging changes)

echo “Hello world” > test.txt

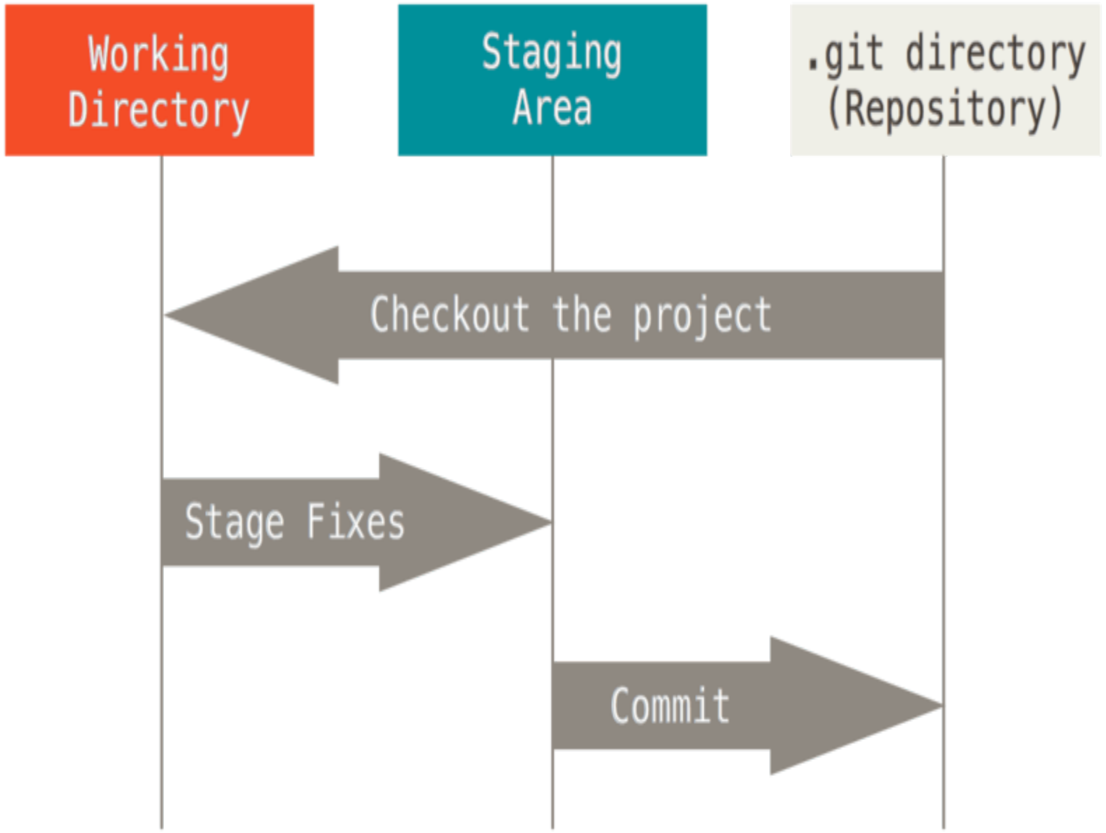
git add test.txt



If you modify a file after you run git add, you have to run git add again to stage the latest version of the file!

GIT: Essentials: Undo Check-in (Un-Staging changes)

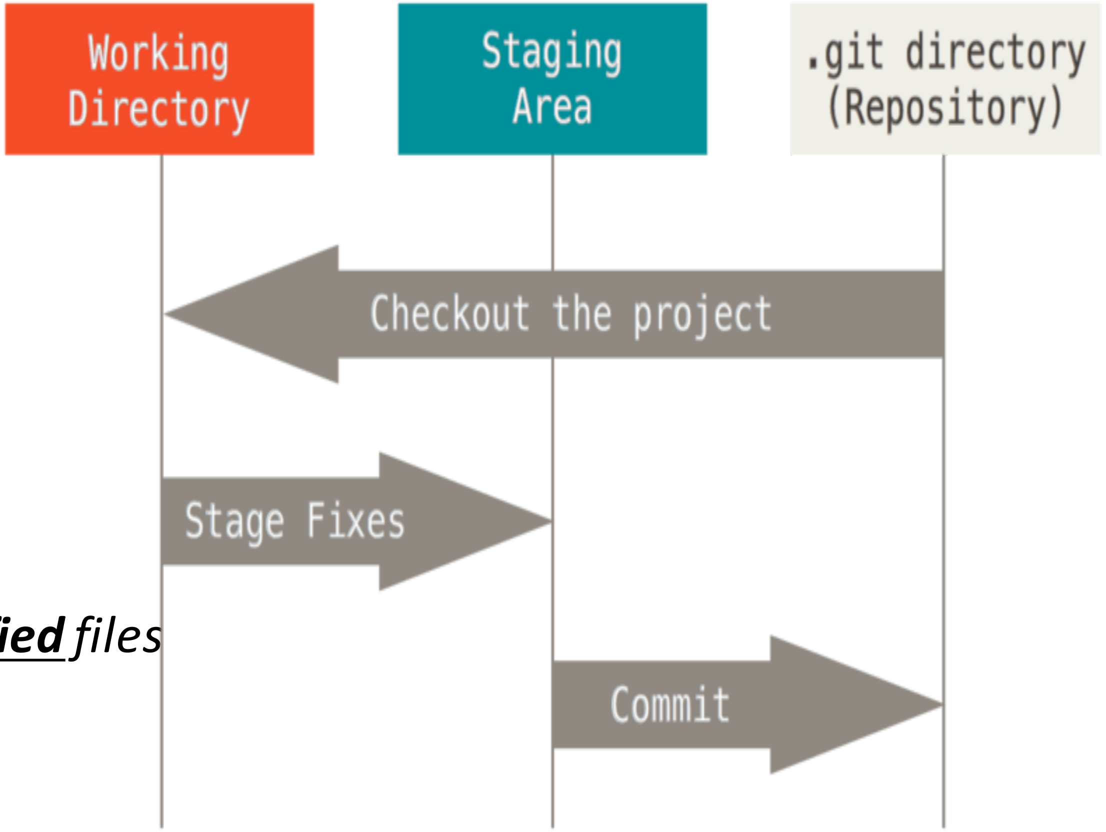
```
git reset test.txt
```



GIT: Essentials: Committing Changes

Commit from stage to repo

git commit -m "Added text.txt"

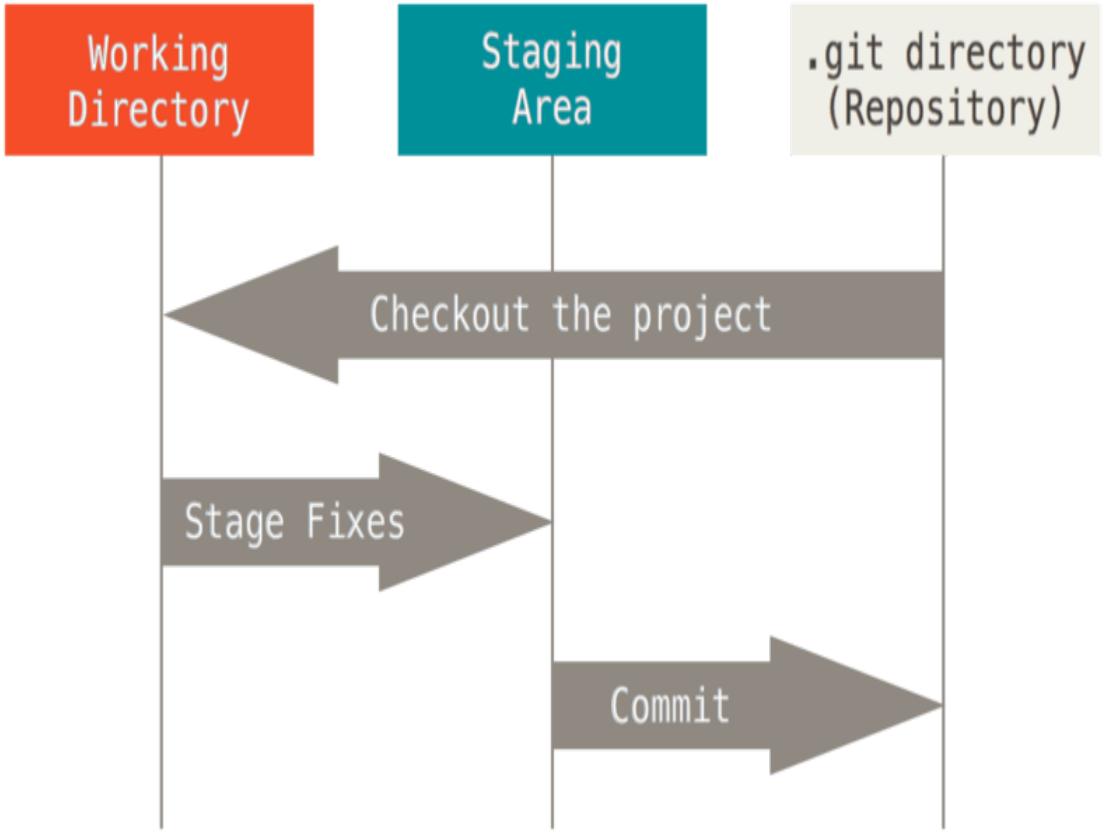


*Directly commit **<u>tracked but modified</u>** files*

git commit -a -m 'made a change'

GIT: Essentials: Check-Out

```
git checkout -- test.txt
```

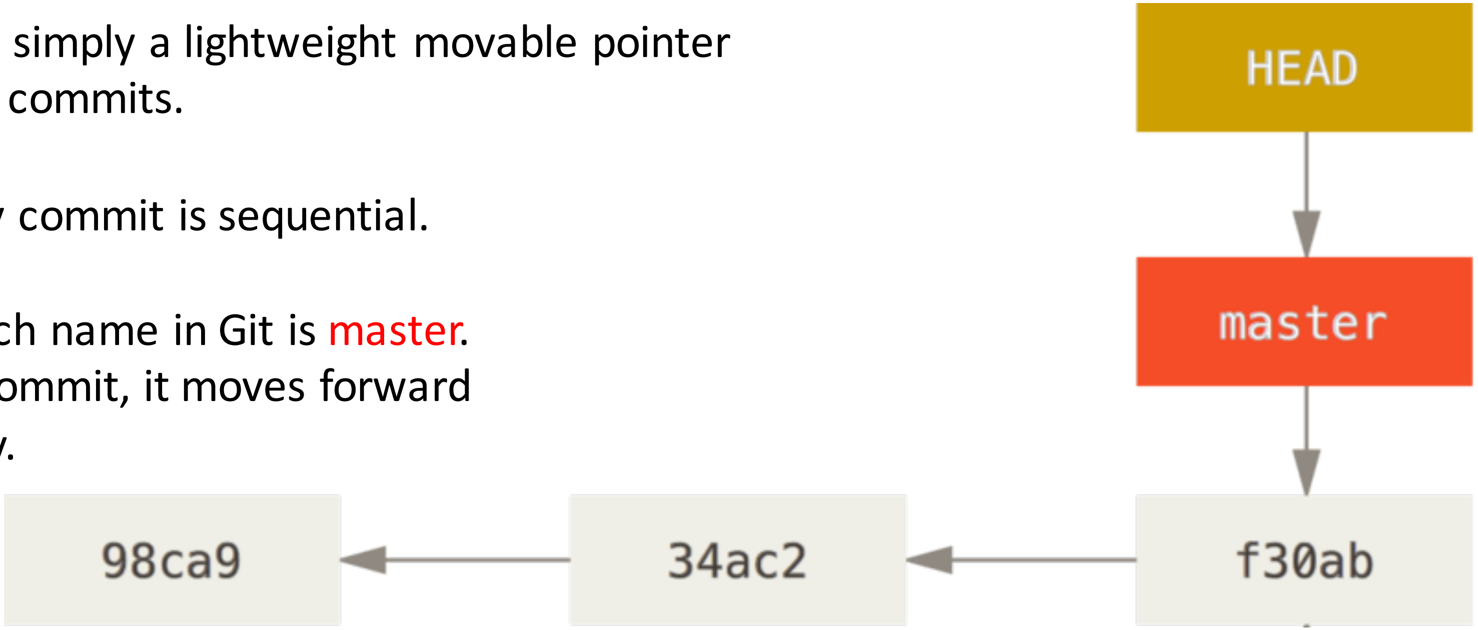


GIT: Essentials: Branching

A branch in Git is simply a lightweight movable pointer to one of the commits.

Remember every commit is sequential.

The default branch name in Git is master.
Every time you commit, it moves forward automatically.



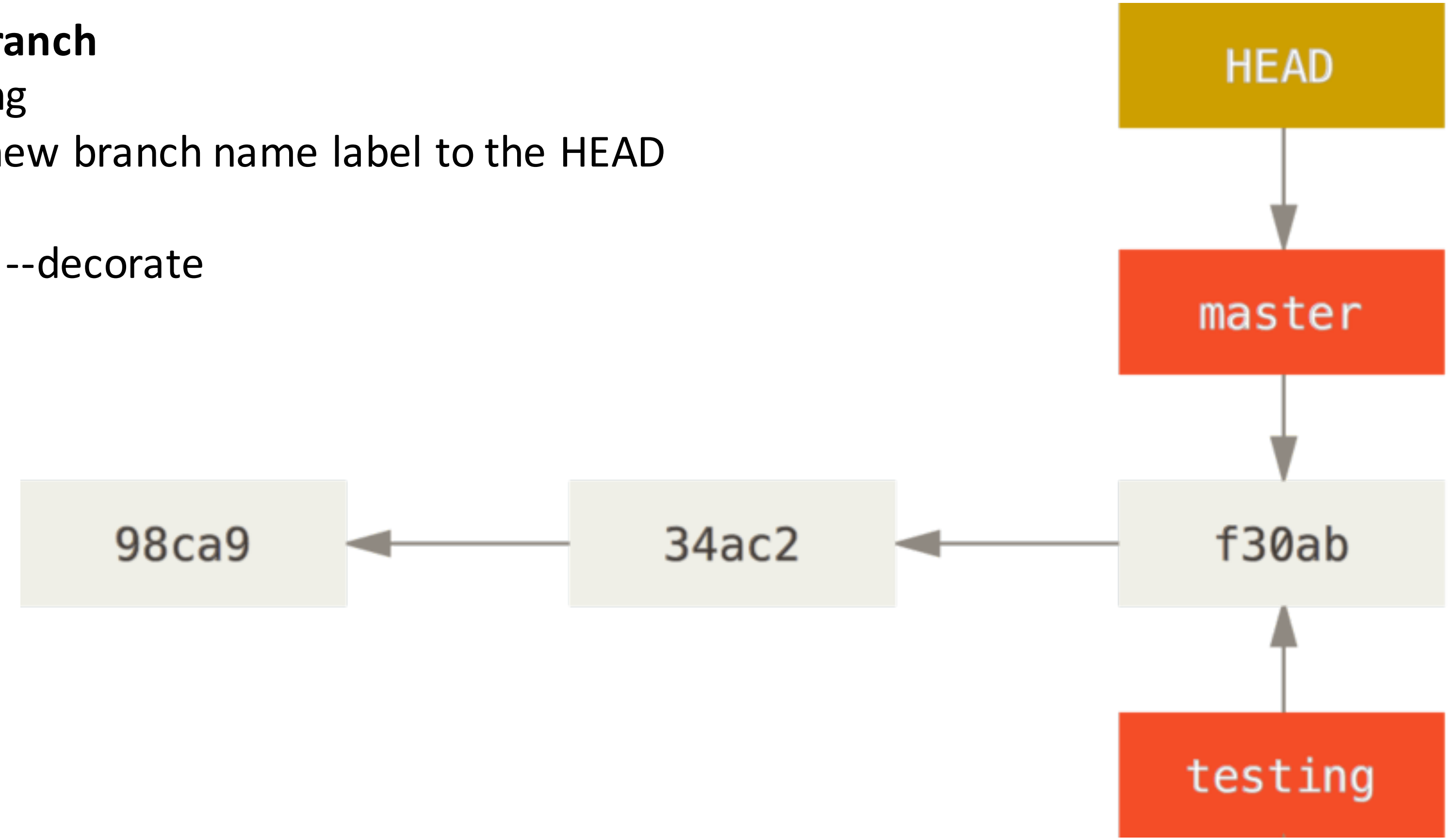
GIT: Essentials: Branching

Create a new branch

git branch testing

just adds a new branch name label to the HEAD

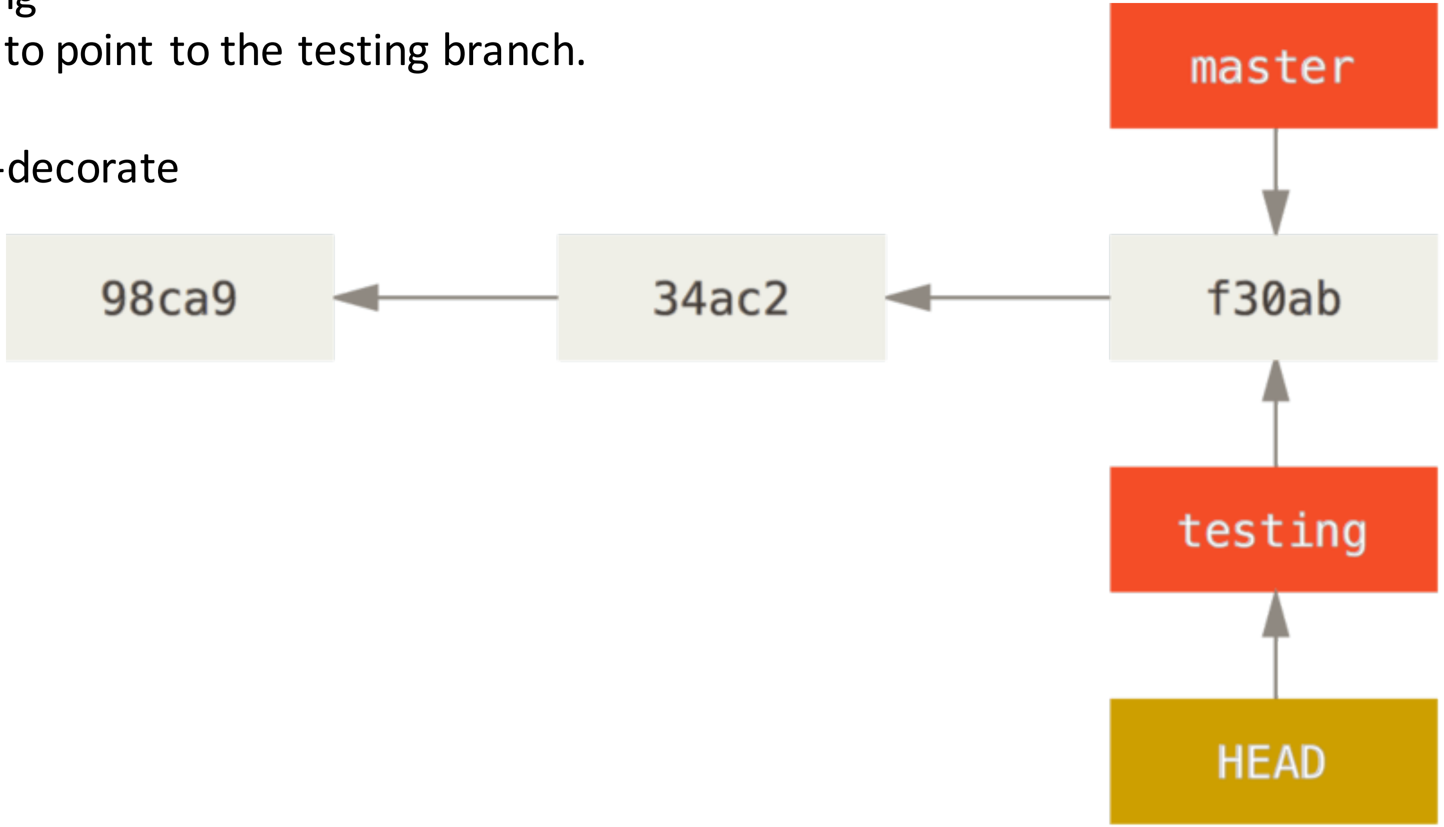
git log --oneline --decorate



GIT: Essentials: Branching

git checkout testing
Just moves HEAD to point to the testing branch.

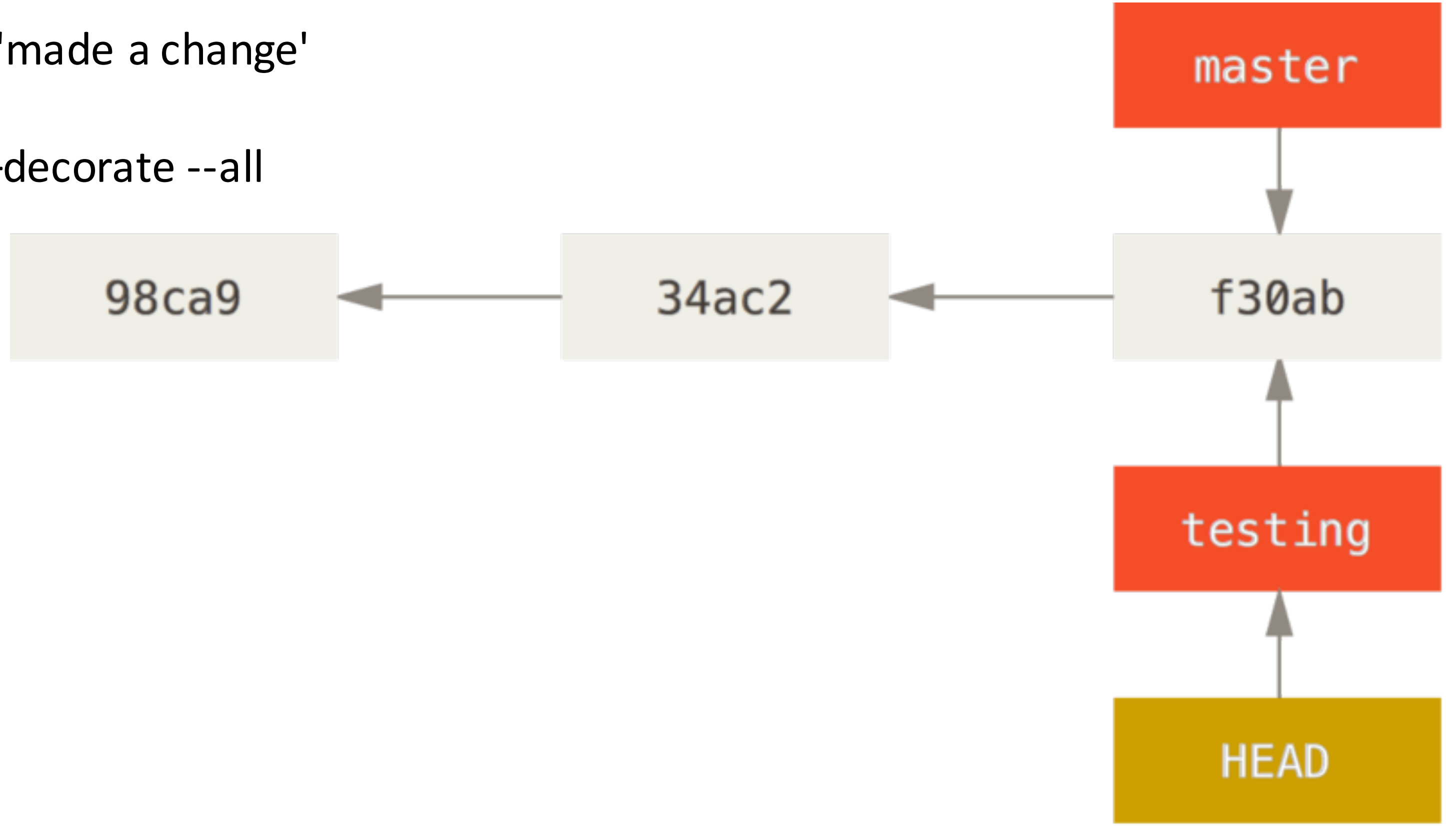
git log --oneline --decorate



GIT: Essentials: Branching

edit a file
git commit -a -m 'made a change'

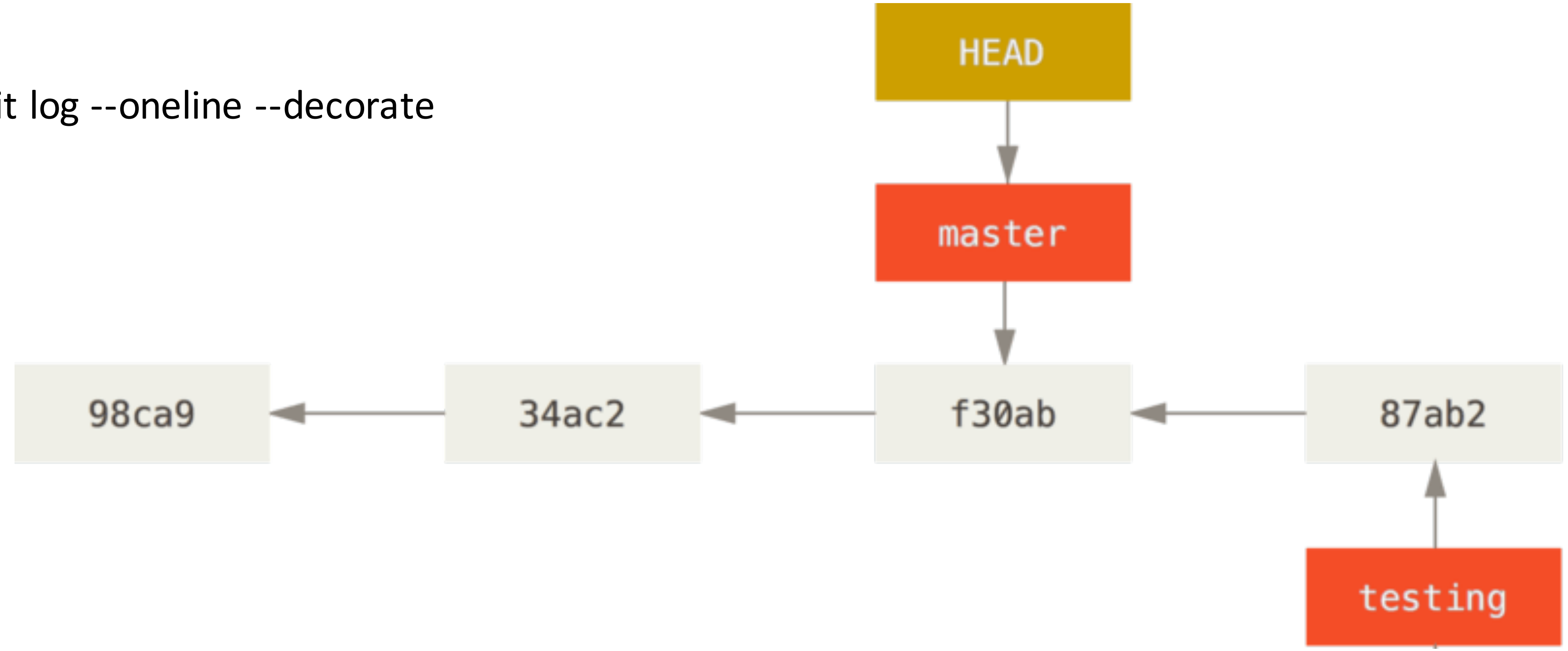
git log --oneline –decorate --all



GIT: Essentials: Branching

git checkout master

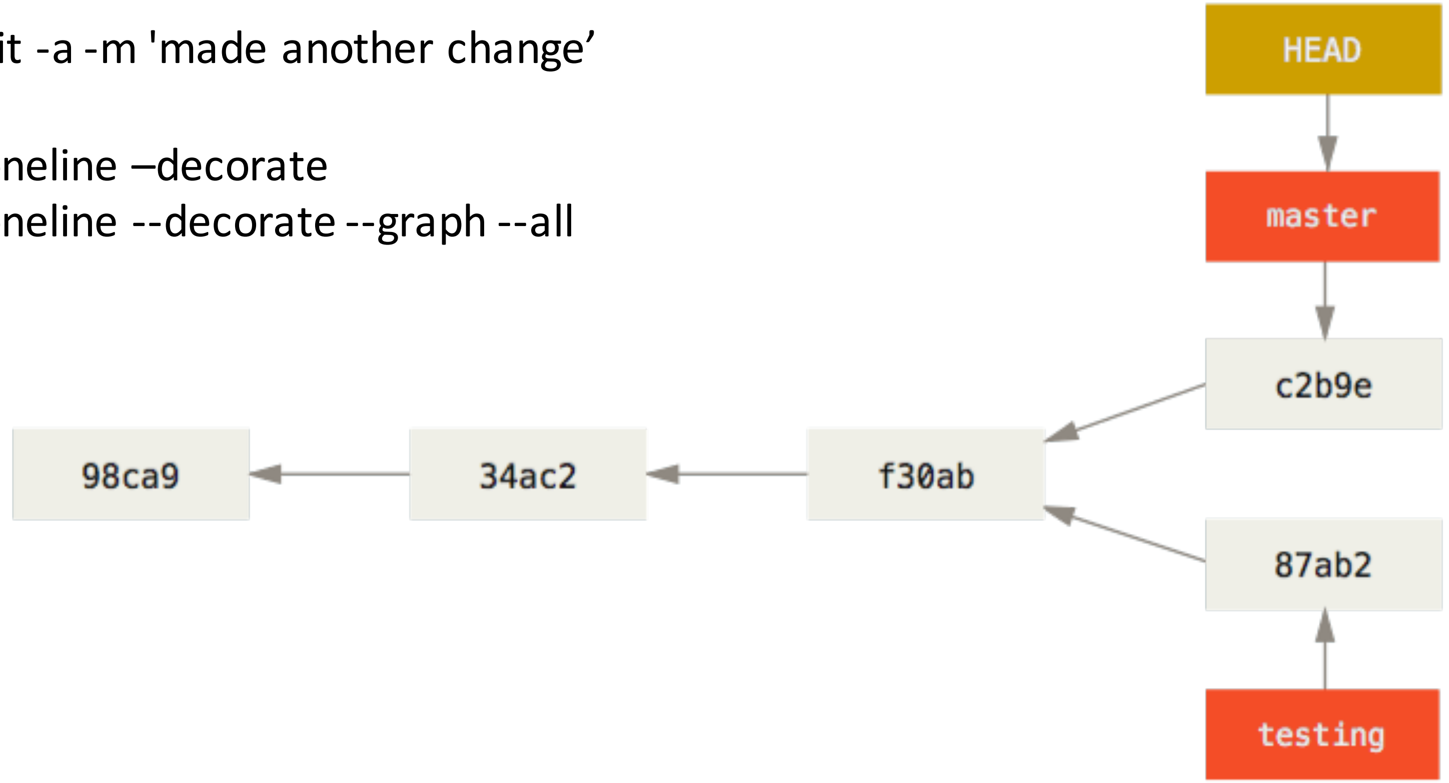
git log --oneline --decorate



GIT: Essentials: Branching

edit a file
git commit -a -m 'made another change’

git log --oneline –decorate
git log --oneline --decorate --graph --all



GIT: Essentials: Viewing Changes: Git Status

Viewing Your Staged and Unstaged Changes

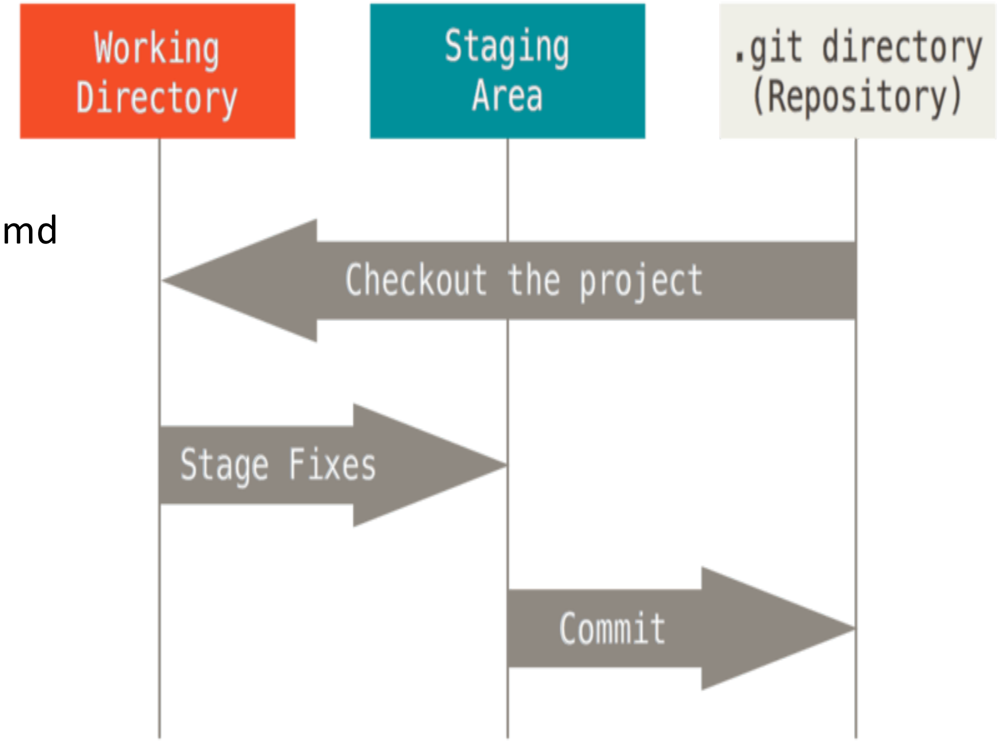
```
git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

GIT: Essentials: Viewing Changes: Git Diff

Compare Staged and Unstaged Changes

git diff

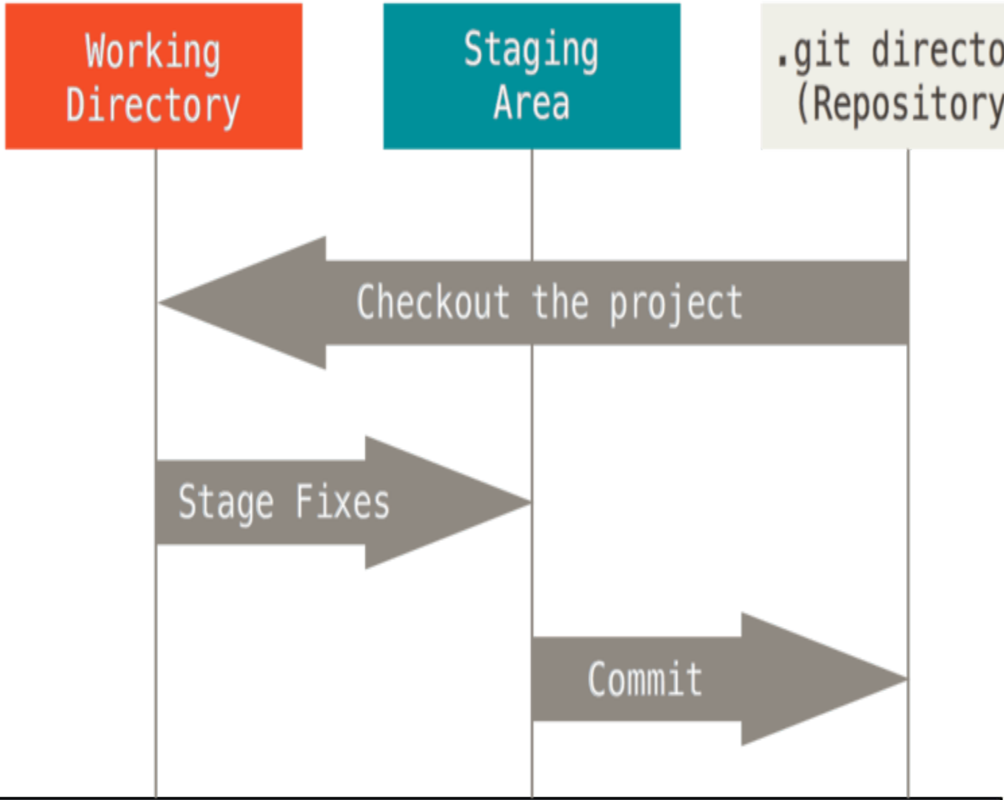
```
diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md
index 8ebb991..643e24f 100644
--- a/CONTRIBUTING.md
+++ b/CONTRIBUTING.md
```



GIT: Essentials: Viewing Changes: Git Diff

Viewing Your Staged and Committed (last) Changes

git diff –staged OR
git diff --cached



GIT: Essentials: Viewing Changes: Git Log

```
git log
git log -p -2 #-p shows the difference introduced in each commit
git log --stat
git log --pretty=oneline
git log --pretty=format:"%h - %an, %ar : %s"
git log --pretty=format:"%h %s" --graph
git log --since=2.weeks
git log -S function_name
git log --graph --decorate --oneline --all
```

GIT: Essentials: Working with Remotes, Fetch & Push

```
git clone https://github.com/schacon/ticgit
cd ticgit
git remote
git remote -v
git remote add pb https://github.com/paulboone/ticgit
git fetch pb
```

```
git push [remote-name] [branch-name]
git push origin master
```

GIT: Essentials: Cloning a Repository

If you want to get a copy of an existing Git repository

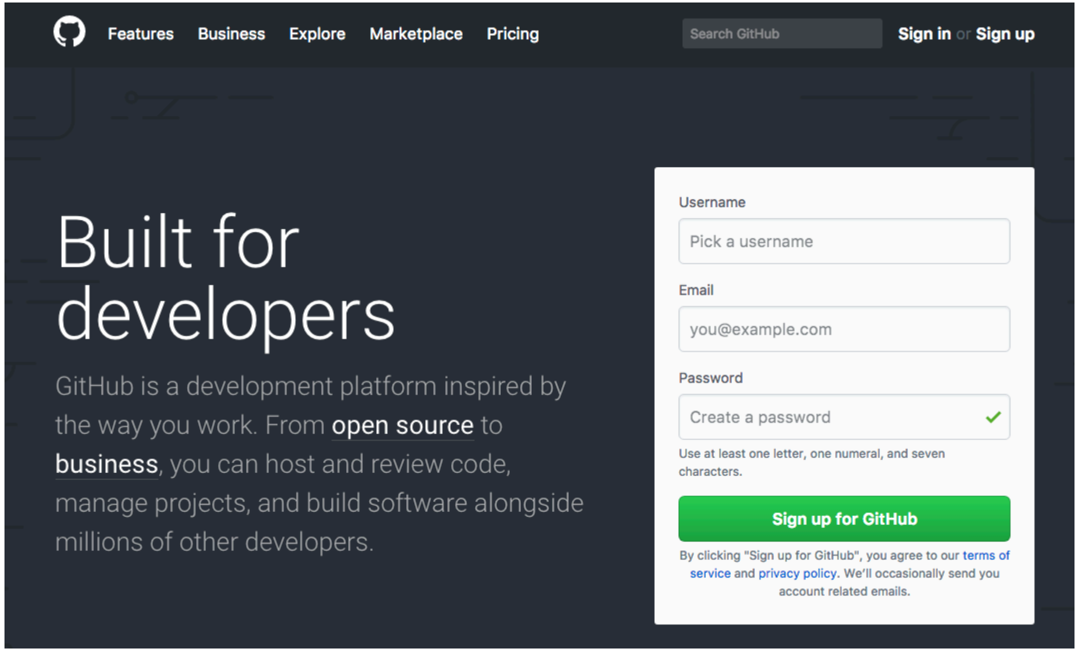
git clone https://github.com/libgit2/libgit2

That creates a directory named "libgit2", initializes a .git directory inside it, pulls down **all the data for that repository**, and **checks out** a working copy of the latest version. If you go into the new libgit2 directory, you'll see the project files in there, ready to be worked on or used.

Git has a number of different transfer protocols you can use, you may also see git:// or user@server:path/to/repo.git, which uses the SSH transfer protocol.

GIT: Essentials: GitHub

You MUST Create Your Online GitHub Account and Repositories!
https://github.com



Built for developers

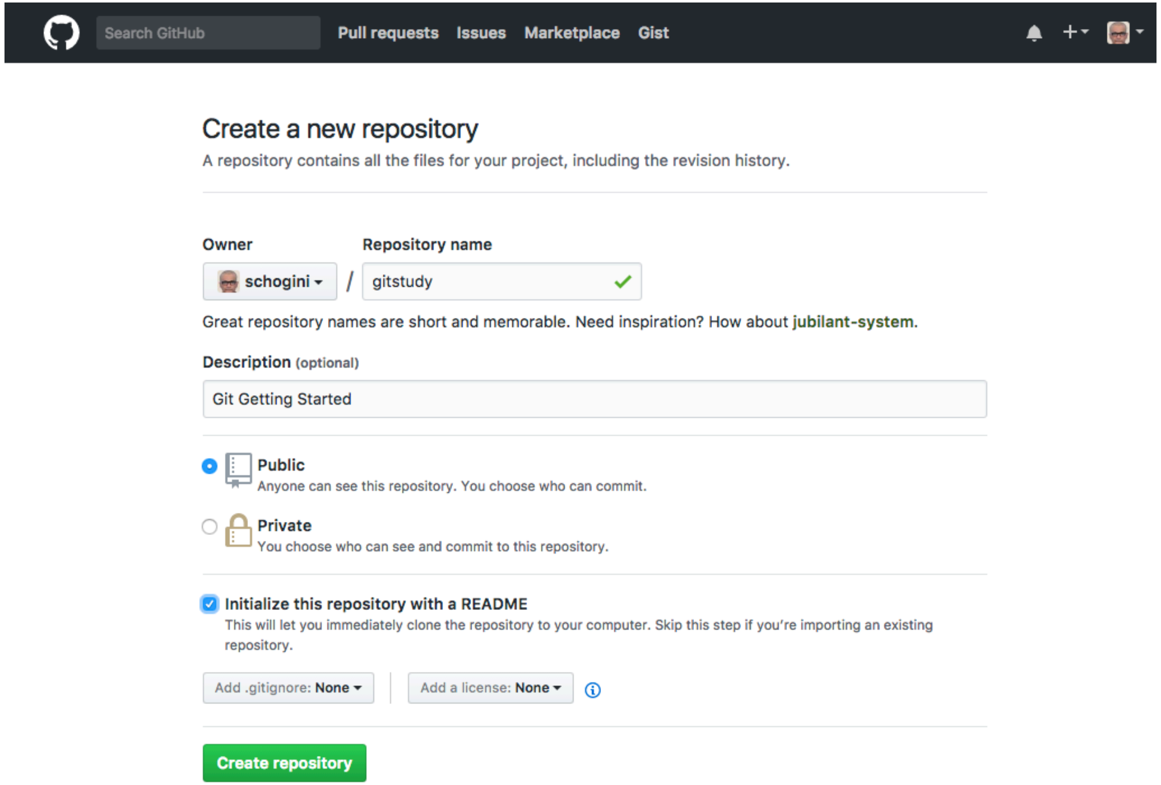
GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside millions of other developers.

Username
Pick a username
Email
you@example.com
Password
Create a password
Use at least one letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.

GIT: Essentials: GitHub

You MUST Create Your Online GitHub Account and Repositories!



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: schogini / Repository name: gitstudy ✓

Great repository names are short and memorable. Need inspiration? How about jubilant-system.

Description (optional): Git Getting Started

● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

[✓] Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None
Add a license: None

GIT: Essentials: GitHub

You MUST Create Your Online GitHub Account and Repositories!

Create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/schogini/test.git
git push -u origin master
```

HTTPS URL https://github.com/schogini/test.git or
SSH URL git@github.com:schogini/test.git

GIT: Essentials: GitHub

Cloning a GitHub repository on the command line

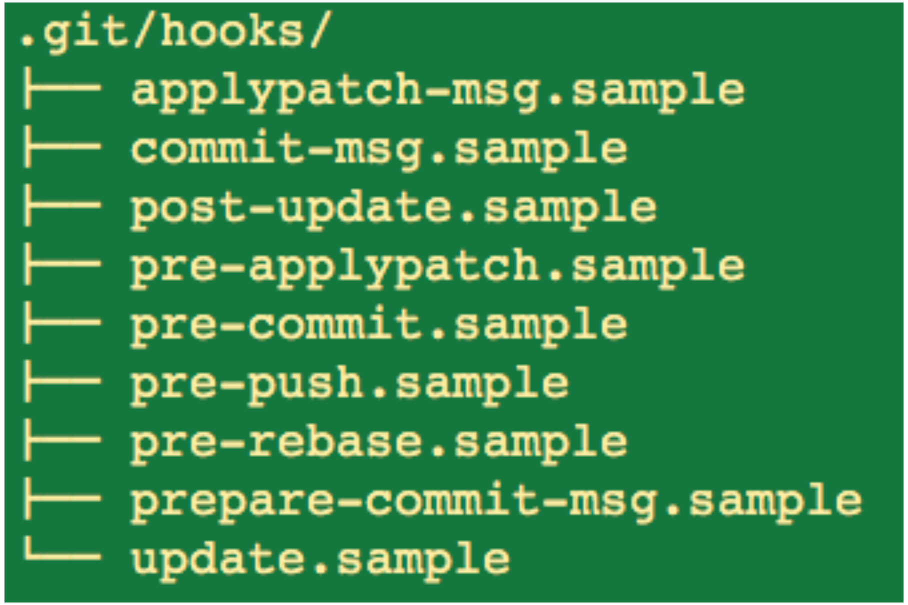
```
git clone https://github.com/schogini/test.git
```

GIT: Essentials: GitHub

Push to an existing repository from the command line to GitHub

```
git remote add origin https://github.com/schogini/test.git
git push -u origin master
```

GIT: Essentials: Automation Local



```
.git/hooks/
├── applypatch-msg.sample
├── commit-msg.sample
├── post-update.sample
├── pre-applypatch.sample
├── pre-commit.sample
├── pre-push.sample
├── pre-rebase.sample
├── prepare-commit-msg.sample
└── update.sample
```

GIT: Essentials: Extras-1

1. **Removing a file from staging**
   git reset <filename>

2. **Discard/Undo**
   git checkout -- <file>…

3. **Short Status**
   git status –s

4. **Ignoring Files**
   Create a ***.gitignore*** file
   *.[oa]
   *~

GIT: Essentials: Extras-2

1. **Stage the file to be removed from the repo and also remove it from the working directory**

   ```
   git rm <file>
   ```

2. **Stage a file to be removed from the repo without deleting it from the working directory**

   ```
   git rm --cached <file>
   ```

1. **Amending a commit**

   ```
   git commit --amend
   ```

GIT: Essentials: Everyday Use

LOCAL

git init

git add –A

git commit -m “Message”

--- INSTEAD--

git commit –a -m “Message”

REMOTE REPO

git clone <URL>

  --- above local commands –

git fetch origin master

(and git push origin master – if you have write permissions)

GIT: Essentials: Home Work

1. Create a local repository called test1 and
2. Create a file called file1 and add and commit. Try out the commands from this presentation.
3. Create a GitHub account and a Repo and clone it locally
4. Add a file to it and commit and push it back to GitHub and see that it is present online

For technical support:

Call : +91 75730 27611

E-mail : tactsupport@collabera.com

Visit us at : http://www.collaberatact.com