Introduction to Git



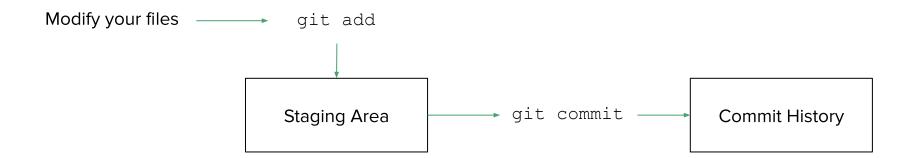
Jeriel Ng (@jerielng) Mobile Software Developer at NCR Auburn University 2019

What is Git?

- Version control system
- Developed by Linus Torvalds (Linux)
- Why is version control important?
 - Logged history of changes
 - Collaborative codebase



Commit Process



What is a commit?

A single point of time in your Git history that contains a log of changes

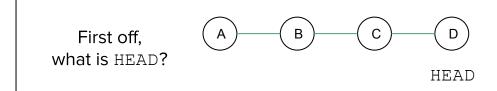
Commit History: Reviewing Your Changes

```
public void createInputDialog() {
                AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
                builder.setMessage(getString(R.string.enter_workout_name));
                final EditText editText = new EditText(getActivity());
                editText.setInputType(InputType.TYPE_CLASS_TEXT);
                builder.setView(editText);
                builder.setPositiveButton(getString(R.string.generate_button_text),
           public void createInputDialog(String previousInput) {
199 +
200 +
                mInputBuilder = new AlertDialog.Builder(getActivity());
                mInputBuilder.setMessage(getString(R.string.enter_workout_name));
201 +
202 +
                mInputField = new EditText(getActivity());
203 +
                mInputField.setInputType(InputType.TYPE CLASS TEXT);
204 +
                mInputBuilder.setView(mInputField);
205 +
               if (!TextUtils.isEmpty(previousInput)) {
206 +
                    mInputField.setText(previousInput);
207 +
208 +
                mInputBuilder.setPositiveButton(getString(R.string.generate button text),
                       new DialogInterface.OnClickListener() {
210
                   @Override
                   public void onClick(DialogInterface dialog, int which) {
                       Intent detailIntent = new Intent(getActivity(), WorkoutDetailActivity.class);
                       detailIntent.putExtra(getString(R.string.workout name extra),
                               editText.getText().toString());
214 +
                               mInputField.getText().toString());
```

Reviewing your diff:

- Additions green lines
- Deletions red lines
- Untouched white lines

Commit History: Undoing Your Changes



HEAD is the most recent commit of the branch you're currently on

Default

git reset --hard

Completely discard changes, whether staged or not

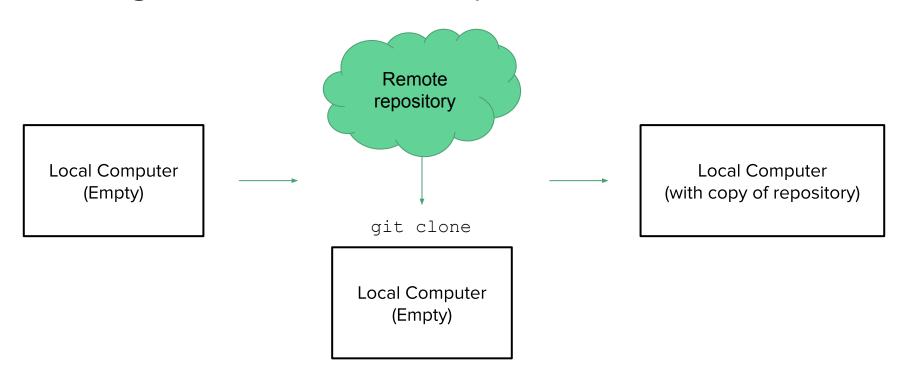
git reset --mixed

Removes files from staging area, but keeps changes

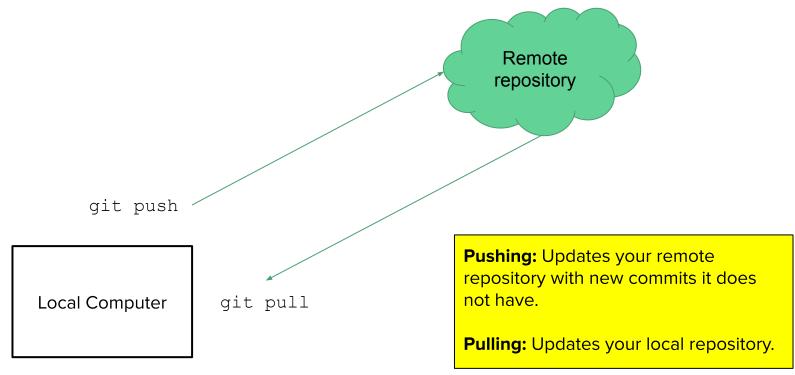
git reset --soft

Moves HEAD pointer, but keeps changes **in** the staging area

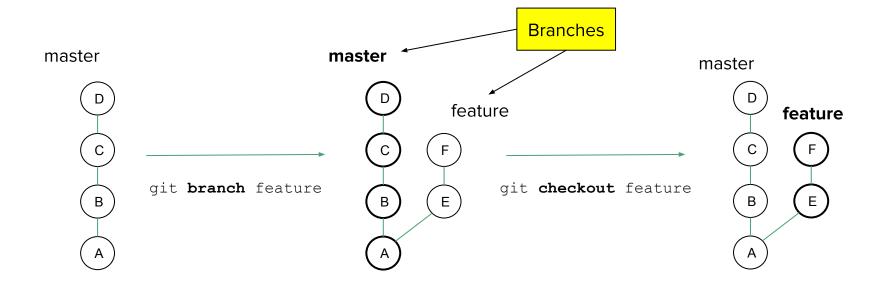
Working with Remotes: Setup



Working with Remotes: Updating

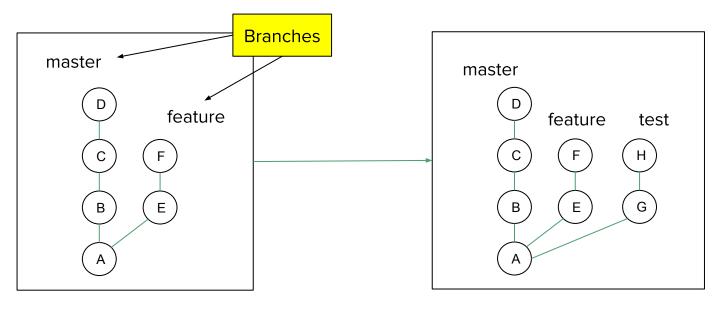


Alternate Timelines: Branches & Forks



You can also accomplish this with a single line:
git checkout -b feature

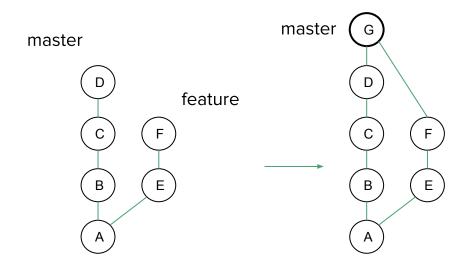
Alternate Timelines: Branches & Forks



awesome project

Fork of awesome project

Merging vs. Rebasing

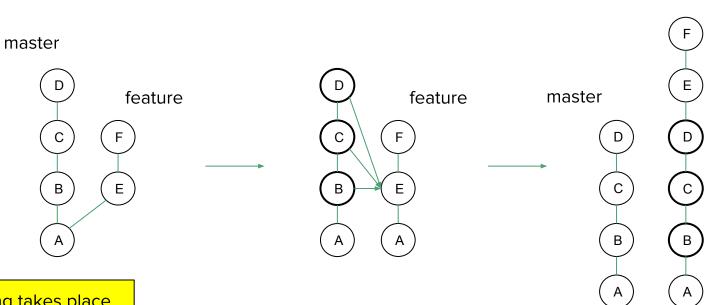


G is a newly created "merge commit" to combine the histories of both master and feature

(master): git merge feature

Notice merging takes place from the **master** branch, not the **feature** branch

Merging vs. **Rebasing**



feature

Notice rebasing takes place from the **feature** branch, not the **master** branch

(feature): git rebase master

Squashing Your Commits

```
feature
                          mick 144871a Create LICENSE
                            d62b33 Adjusted gitignore
                            531e55 Updated view controller
                                                                                                                       feature
  HEAD
                         # Rebase fa5c732..6531e55 onto fa5c732 (3 commands)
                           Commands:
HEAD~2
                           p, pick <commit> = use commit
                           r, reword <commit> = use commit, but edit the commit message
                          # e, edit <commit> = use commit, but stop for amending
                            s, squash <commit> = use commit, but meld into previous commit
HEAD~3
                            f, fixup <commit> = like "squash", but discard this commit's log message
                            b, break = stop here (continue rebase later with 'git rebase --continue')
                                                                                                                           В
                           d, drop <commit> = remove commit
                            1, label <label> = label current HEAD with a name
                           t, reset <label> = reset HEAD to a label
                           m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
                                   create a merge commit using the original merge commit's
                                   message (or the oneline, if no original merge commit was
                                   specified). Use -c <commit> to reword the commit message.
                           These lines can be re-ordered; they are executed from top to bottom.
```

(feature): git rebase -i HEAD~3

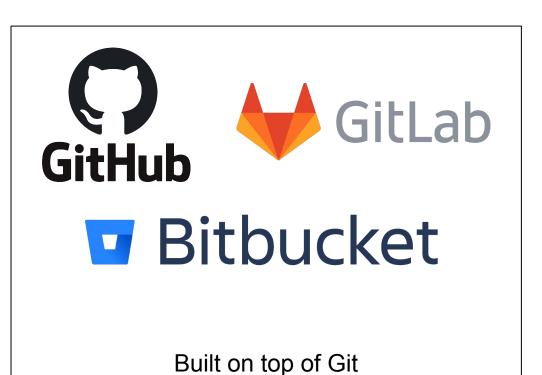
Merge Conflicts & How to Resolve Them

```
sample.txt
This is the first line
                                                     📴 🔲 Theirs: 🦞 merge-demo
                                                                                                 Ours: If master
Maybe a conflict may happen here
                                                       This is the first line
                                                                                                 This is the first line
                                                       I want to do something different here
                                                                                                 Maybe a conflict may happen here
            Coder A's changes
This is the first line
I want to do something different here
                                                       This is the first line
                                                       --- Merge Conflict ---
             Coder B's changes
```

Merge Conflicts & How to Resolve Them

- Best approach: Use a text editor or GUI tool (Especially for big files)
 - Visual Studio Code
 - GitKraken
 - Fork
 - SourceTree
- Have an understanding of who changed what and why
- Run your code afterwards to make sure it compiles

What is GitHub? I've heard of that



- GUI-based platforms
- Features:
 - Repository Hosting
 - Pull Requests
 - o CI/CD Integration

Further Resources

- Git Reference Manual
- Interactive Tutorials on Git
- GitHub Learning Lab
- Desktop GUI tools for Git
 - o <u>Fork</u>
 - o <u>GitKraken</u>
 - o <u>Sourcetree</u>

Questions