# Key Term

**Version control** - Tracking and managing changes to source code over time. Allows coordinating work between multiple people.

**Repository (repo)** - A folder for a project that is tracked by version control. Contains project files and revision history.

**Commit** - An individual change to a project, along with a commit message describing the update. Commits capture a snapshot of the repo at a point in time.

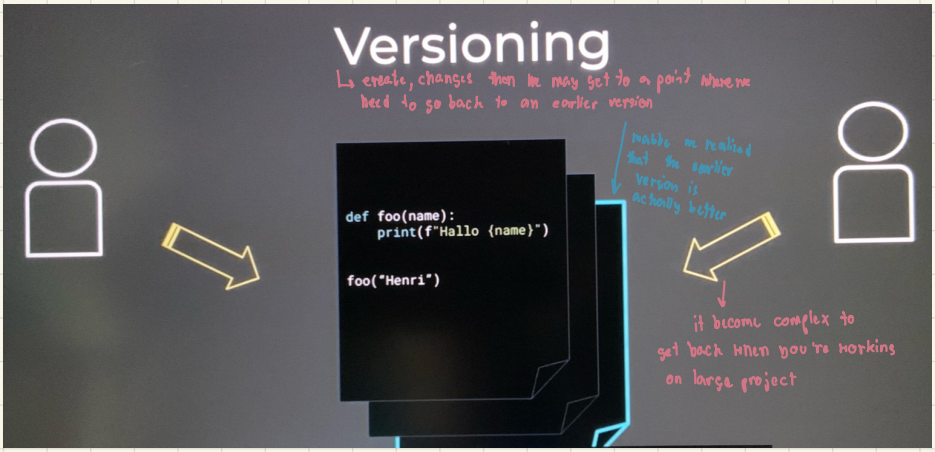
**Branch** - A parallel version of a repo for isolated development. Useful for experimenting without impacting the main code.

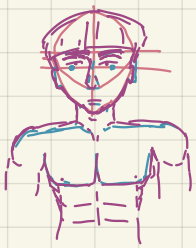
**Merge** - Combining changes from separate branches. Used to bring divergent branches back together.

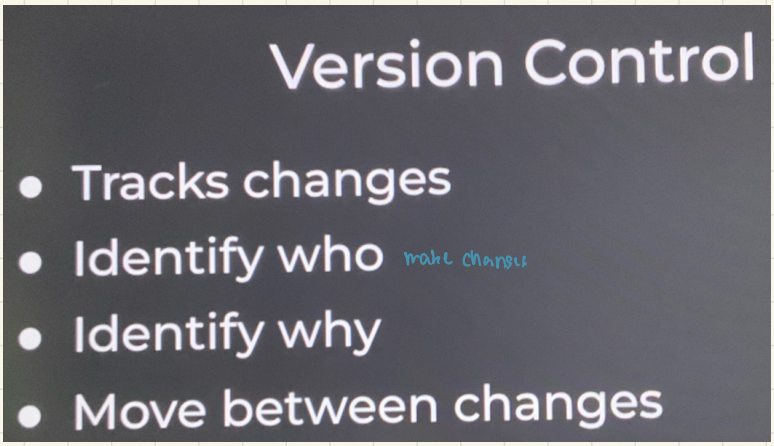
**Pull request -** A request to merge code from one branch into another branch. Allows discussion and review before merging.

```
# Clone an existing repo from GitHub
git clone https://github.com/user/repo.git

# Create a new local repo
git init my-new-repo

# Check status of changes in repo
git status

# Stage files to commit
git add file1.txt file2.txt

# Commit changes with message
git commit -m "Updated feature X"

# Create a new branch
git branch new-feature

# Switch to another branch
git checkout another-branch

# Merge new-feature branch into main
git checkout main
git merge new-feature
```

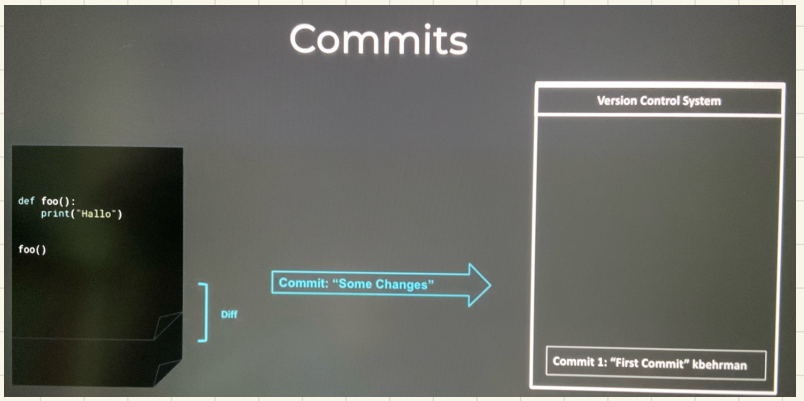
# What is Version Control?

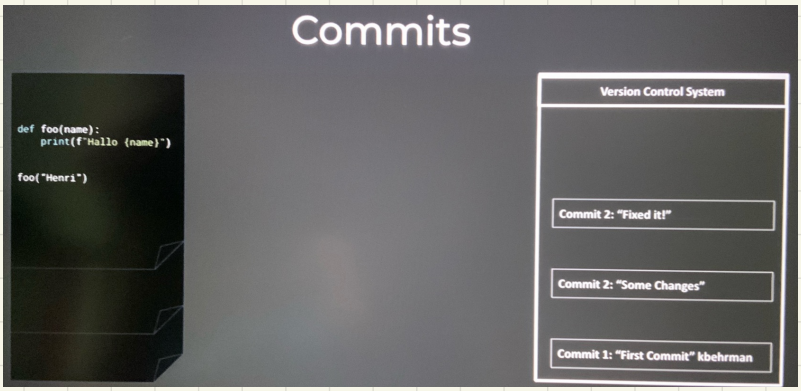


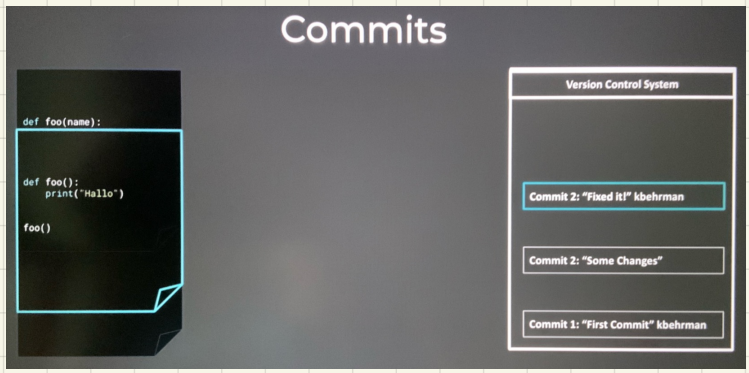









# Introduction to Git and Git Concepts

## Git Commands

- init - setup repository
- status - report on status of repo contents
- add - include file in commit
- commit - record changes and pushes them into the GIT system
- log - see commit history
- diff - see changes the differents between what you have and the last commit
- checkout - change branch
- stash - store changes in temporary cache
- merge - apply changes from another branch

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % ls
hello.py
kennedyrobertbehrman@MacBook-Pro-2 git_example % cat hello.py
print("Hello")
kennedyrobertbehrman@MacBook-Pro-2 git_example % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/kennedyrobertbehrman/Google Drive/projects/python.and.pandas.f
or.data.engineering.Duke.Coursera/git_example/.git/
kennedyrobertbehrman@MacBook-Pro-2 git_example % git status
On branch master

No commits yet

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

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

data was already set

first to use GIT

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git add hello.py
kennedyrobertbehrman@MacBook-Pro-2 git_example % git status
On branch master

No commits yet

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

use "add" to track, add to Git Repository

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git commit -m "adding new file"
[master (root-commit) b8dd624] adding new file
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
kennedyrobertbehrman@MacBook-Pro-2 git_example % git status
On branch master
nothing to commit, working tree clean
kennedyrobertbehrman@MacBook-Pro-2 git_example %
```

→ to always supply a commit message

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git log
commit b8dd6248ff37c1fa3241731cf9dc945273f3f41c (HEAD -> master)
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 15:58:26 2022 -0800

    adding new file
```

→ see the log, history of our commit

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % vim hello.py
kennedyrobertbehrman@MacBook-Pro-2 git_example % git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.py

no changes added to commit (use "git add" and/or "git commit -a")
kennedyrobertbehrman@MacBook-Pro-2 git_example % git diff
diff --git a/hello.py b/hello.py
index 2f9a147..f76682a 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1 @@
-print("Hello")
+print("Hello Paul")
```

→ to modify

old ver.

new ver

# Introduction to Git and Git Concepts

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git stash
Saved working directory and index state WIP on master: b8dd624 adding new file
kennedyrobertbehrman@MacBook-Pro-2 git_example % git status
On branch master
nothing to commit, working tree clean
kennedyrobertbehrman@MacBook-Pro-2 git_example % git stash pop
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.py

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (f12c7c86514a1d3c78cf5013bec2bb5a6a11c609)
```

git stash to move back to the last commited ver.

get changes back from stash

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git commit hello.py -m "added name"
[master c0b90a2] added name
 1 file changed, 1 insertion(+), 1 deletion(-)
kennedyrobertbehrman@MacBook-Pro-2 git_example % git checkout -b new-branch
Switched to a new branch 'new-branch'
kennedyrobertbehrman@MacBook-Pro-2 git_example % git branch
  master
* new-branch
kennedyrobertbehrman@MacBook-Pro-2 git_example % vim hello.py
kennedyrobertbehrman@MacBook-Pro-2 git_example % git status
On branch new-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.py

no changes added to commit (use "git add" and/or "git commit -a")
```

commit this changed

create new branch

see which branch we are working on

```
no changes added to commit (use "git add" and/or "git commit -a")
kennedyrobertbehrman@MacBook-Pro-2 git_example % git diff
diff --git a/hello.py b/hello.py
index f76682a..8b8f074 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1 @@
-print("Hello Paul")
+print("Hello Sue")
kennedyrobertbehrman@MacBook-Pro-2 git_example % git commit hello.py -m "changed name to Sue"
[new-branch 3ba5b8a] changed name to Sue
 1 file changed, 1 insertion(+), 1 deletion(-)
kennedyrobertbehrman@MacBook-Pro-2 git_example % git log
commit 3ba5b8a25d16a7d50555869abaf05ccc25be8cf3 (HEAD -> new-branch)
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 16:02:05 2022 -0800

    changed name to Sue

commit c0b90a2b12ef06822882bc68338380a40c9e2984 (master)
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 16:00:32 2022 -0800

    added name

commit b8dd6248ff37c1fa3241731cf9dc945273f3f41c
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 15:58:26 2022 -0800

    adding new file
```

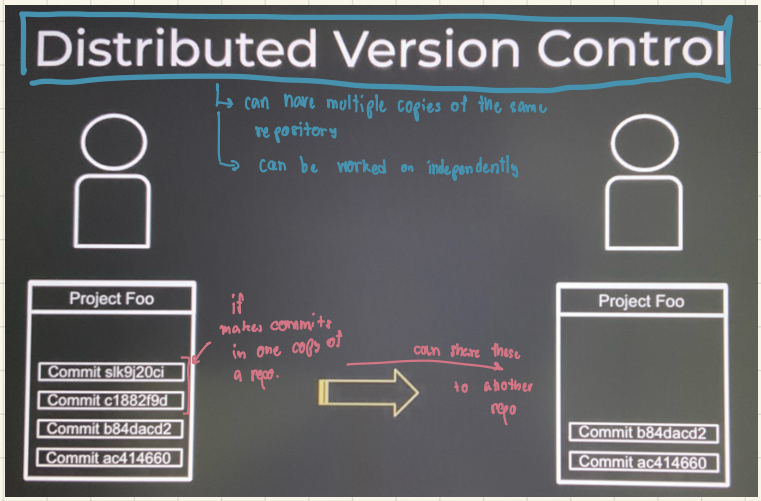
```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git diff master
diff --git a/hello.py b/hello.py
index f76682a..8b8f074 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1 @@
-print("Hello Paul")
+print("Hello Sue")
kennedyrobertbehrman@MacBook-Pro-2 git_example % git checkout master
Switched to branch 'master'
kennedyrobertbehrman@MacBook-Pro-2 git_example % git log
commit c0b90a2b12ef06822882bc68338380a40c9e2984 (HEAD -> master)
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 16:00:32 2022 -0800

    added name

commit b8dd6248ff37c1fa3241731cf9dc945273f3f41c
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 15:58:26 2022 -0800

    adding new file
```

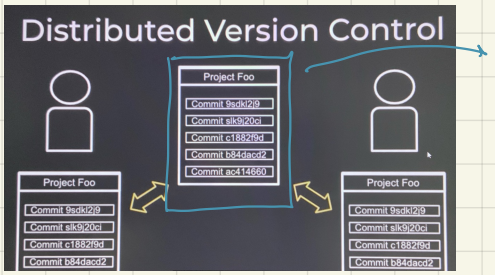
go back to master branch

```
kennedyrobertbehrman@MacBook-Pro-2 git_example % git merge new-branch
Updating c0b90a2..3ba5b8a
Fast-forward
 hello.py | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
kennedyrobertbehrman@MacBook-Pro-2 git_example % git log
commit 3ba5b8a25d16a7d50555869abaf05ccc25be8cf3 (HEAD -> master, new-branch)
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 16:02:05 2022 -0800

    changed name to Sue

commit c0b90a2b12ef06822882bc68338380a40c9e2984
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 16:00:32 2022 -0800

    added name

commit b8dd6248ff37c1fa3241731cf9dc945273f3f41c
Author: kbehrman <kr.behrman@gmail.com>
Date:   Sun Feb 6 15:58:26 2022 -0800

    adding new file
```

brings the changes from other branch to master

# Version Control with GitHub




```
$ git clone https://github.com/kbehrman/demo-project.git
Cloning into 'demo-project'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
$ cd demo-project
$ ls
README.md
$ vim README.md
```

clone

specify dir

modify

```
1 # demo-project
2
3 ## Some real work is going on here
```

modified

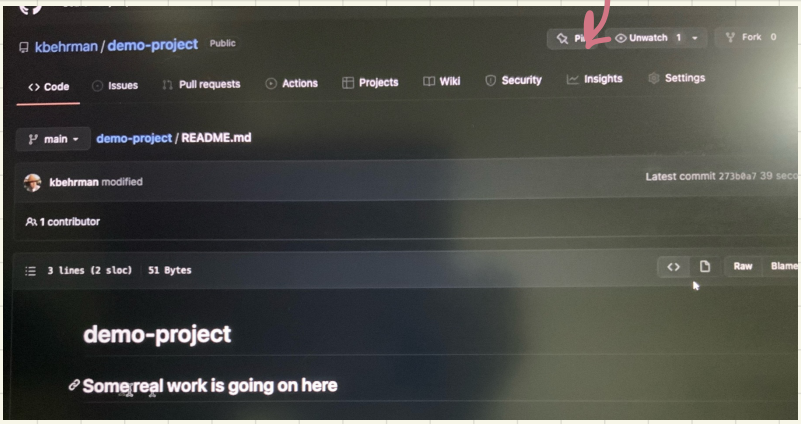
```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

no changes added to commit (use "git add" and/or "git commit -a")
$ git commit -a -m "modified"
[main 273b0a7] modified
 1 file changed, 3 insertions(+), 1 deletion(-)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kbehrman/demo-project.git
   f9ccab3..273b0a7  main -> main
```

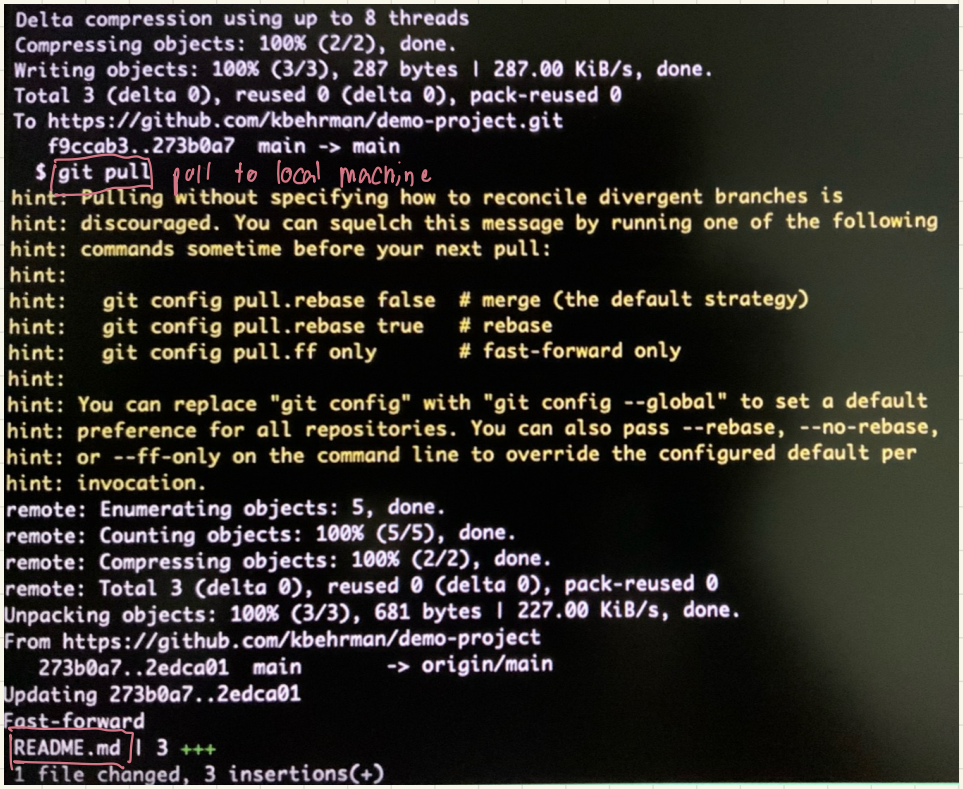
check

commit

push to Github

already changed






**Key Points**

- Version control with Git allows tracking changes to code over time and coordinating work between multiple developers
- Repositories contain the files for a project as well as its entire revision history
- Commits capture snapshots of changes, along with descriptive messages explaining the updates
- Branches allow working on independent streams of changes before merging back together
- GitHub provides remote hosting of repositories for collaboration and sharing

**Reflection Questions**

1. What types of projects would benefit from using version control with Git?
2. How can topic branches help improve coordination when multiple developers work on the same codebase?
3. What best practices should you follow when crafting commit messages?
4. How can you leverage pull requests during code reviews before merging new features?
5. What Git commands seem most valuable as part of your typical development workflow?

**Challenge Exercises**

1. Initialize a local Git repo and make some commits changing multiple files
2. Push the repo to a GitHub remote and open pull requests between branches
3. Explore the commit history of an active open source GitHub project
4. Set up SSH keys for simplified authentication with remote servers
5. Configure your identity globally and per-project with git config