

# Introduction to Parallel Processing

Lecture 1 : Tools for Parallel Computing

Professor Amanda Bienz

# Github

- Version control
- Useful for programming in general
  - Stored online, can work on multiple computers
  - Useful for team programming (large projects)
    - Teammates and I all make changes
  - **Useful for parallel computing**
    - **Work between local machine and supercomputers**

# Github

- <https://github.com/>
  - If you don't have an account, you will need to make one for this course
  - Check out GitHub education! Great resource for students

# Github Accounts

- Your GitHub password is used to log in on [GitHub.com](https://GitHub.com)
- It is not used to push/pull/clone/checkout/merge etc.
- You can setup your public key on GitHub. Great for local machine. No need to use a password with git commands
- When sshing into a machine, you likely want to use a GitHub token instead

# Public Key

The screenshot shows a GitHub user profile for "Amanda Bienz". The left sidebar lists various account settings, with "SSH and GPG keys" highlighted by a red oval. The main content area is titled "SSH keys / Add new" and contains fields for "Title" and "Key type" (set to "Authentication Key"). A large text box for the "Key" is partially visible, showing the beginning of an SSH key. A green "Add SSH key" button is at the bottom right.

Amanda Bienz  
Your personal account [Switch to another account](#) [Go to your personal profile](#)

Public profile Account Appearance Accessibility Notifications

Access Billing and plans Emails

>Password and authentication **SSH and GPG keys** Organizations

Code, planning, and automation

Repositories Codespaces Packages GitHub Copilot Pages Saved replies

**SSH keys / Add new**

Title

Key type

Authentication Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

# Github Tokens

The screenshot shows the GitHub 'Personal access tokens' page. At the top, there is a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the page title is 'Settings / Developer settings'. On the left side, there is a sidebar with three tabs: 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens'. The 'Personal access tokens' tab is highlighted with a blue border. A large red circle is drawn around this tab. In the main content area, the heading is 'Personal access tokens'. Below it, a sub-heading says 'Tokens you have generated that can be used to access the GitHub API.' There are five entries listed:

Token Name	Scopes	Last Used	Action
Coding	admin:org, delete:packages, delete_repo, project, repo, workflow, write:discussion, write:packages	Last used within the last week	Delete
hoc_pass	delete:packages, repo, workflow, write:packages	Last used within the last 2 weeks	Delete
Pass	repo	Last used within the last 5 months	Delete
sep_token	delete:packages, repo, workflow, write:packages	Last used within the last 9 months	Delete
AugToken	repo	Last used within the last 12 months	Delete

Each entry includes the token name, a list of scopes, the last usage date, and a 'Delete' button. Above the token list, there are two buttons: 'Generate new token' (highlighted with a red circle) and 'Revoke all'. At the bottom of the page, a note states: 'Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#)'.

 © 2022 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

# Github Tokens

Settings / Developer settings

GitHub Apps    OAuth Apps    Personal access tokens

## New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

Github Token for Basic Git Commands

What's this token for?

**Expiration \***

30 days    The token will expire on Thu, Sep 22 2022

**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

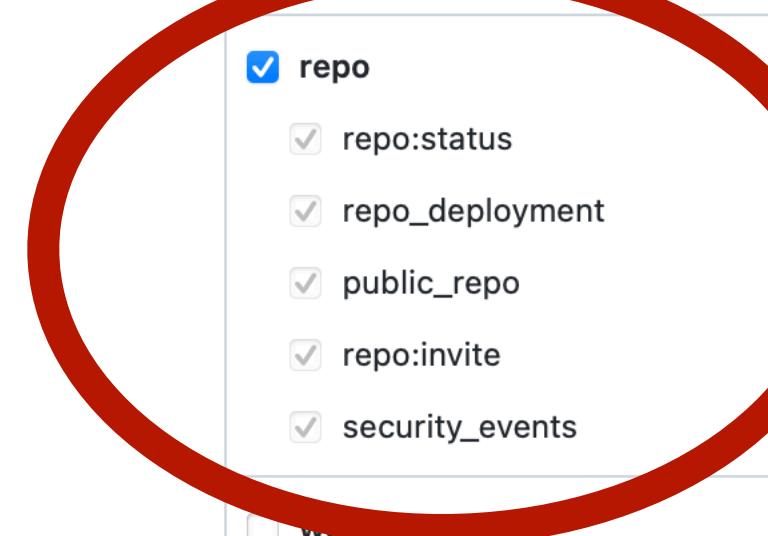
**repo** Full control of private repositories  
   repo:status Access commit status  
   repo\_deployment Access deployment status  
   public\_repo Access public repositories  
   repo:invite Access repository invitations  
   security\_events Read and write security events

**workflow** Update GitHub Action workflows

**write:packages** Upload packages to GitHub Package Registry  
 read:packages Download packages from GitHub Package Registry

**delete:packages** Delete packages from GitHub Package Registry

**admin:org** Full control of orgs and teams, read and write org projects  
 write:org Read and write org and team membership, read and write org projects



# Creating a Repository

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

### Repository template

Start your repository with a template repository's contents.

No template ▾

Owner \*



ProfessorBienz ▾

Repository name \*



Great repository names are short and memorable. Need inspiration? How about [fictional-rotary-phone](#)?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

License: None ▾

# Github Classroom

- You will not create your own repository
- Instead, follow the link posted in the assignment on Canvas

The image shows two side-by-side screenshots of the GitHub Classroom interface. Both screenshots have a dark header bar with the GitHub Classroom logo and navigation links.

**Left Screenshot:** The title is "Join the classroom: CS442/542 Fall 2023". It instructs users to select their GitHub account from a list of identifiers. The list includes:

- Aguilera, Ester
- Anwaar-Maximo, Danial F
- Bickel, Charles Michael
- Biernz, Amanda
- Caffrey, Elektra
- Denlinger, Althea R
- Dominguez, David M

**Right Screenshot:** The title is "CS442/542 Fall 2023 Accept the assignment — CMake and GitHub Tutorial". It informs users that accepting the assignment grants access to the "cmake-and-github-tutorial-biernz2" repository in the "ProfessorBiernz" organization. A green button at the bottom right says "Accept this assignment".

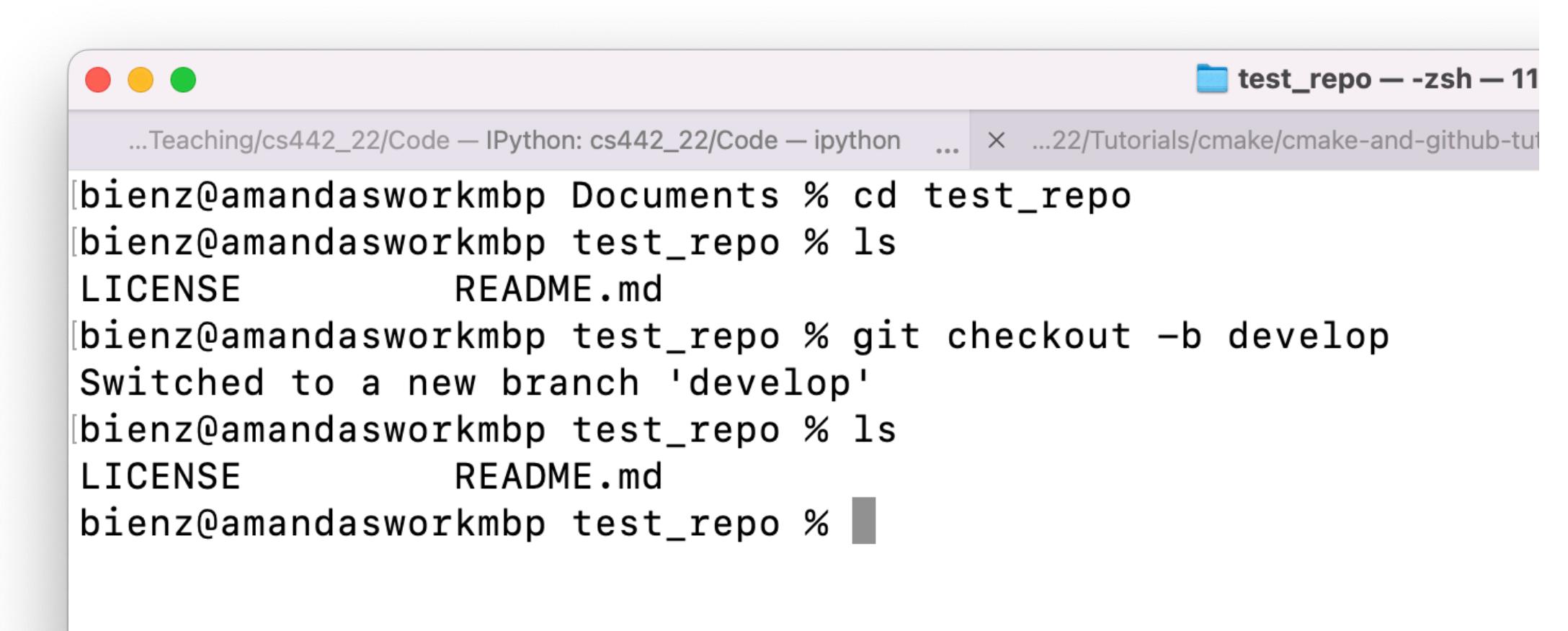
# Cloning a Repository

The image shows a composite screenshot illustrating the process of cloning a GitHub repository. On the left, a screenshot of a GitHub repository page for 'ProfessorBienz/test\_repo' is displayed. The 'Code' tab is selected, showing the main branch and a list of files: '.gitignore', 'LICENSE', and 'README.md'. A 'Clone' modal window is open over the code area, providing options for cloning via HTTPS, SSH, or GitHub CLI. The HTTPS URL is highlighted as `https://github.com/ProfessorBienz/test`. Below the URL, instructions state 'Use Git or checkout with SVN using the web URL.' Further down, there are links to 'Open with GitHub Desktop' and 'Download ZIP'. On the right, a terminal window titled 'Documents -- zsh -- 112x45' shows the command being run: `bienz@amandasworkmbp Documents % git clone git@github.com:ProfessorBienz/test_repo.git`. The terminal output details the cloning process, including object enumeration, counting, compressing, and receiving, all completed at 100%.

```
bienz@amandasworkmbp Documents % git clone git@github.com:ProfessorBienz/test_repo.git
Cloning into 'test_repo'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
bienz@amandasworkmbp Documents %
```

# Creating a Branch

- If others are using your repo, you want the master ('main') branch to always be working
- When making changes, work in a different branch (e.g. 'develop')
- When you have tested changes and they are working, merge into master



```
...Teaching/cs442_22/Code - IPython: cs442_22/Code - ipython ... x ...22/Tutorials/cmake-and-github-tut
[bienz@amandasworkmbp Documents % cd test_repo
[bienz@amandasworkmbp test_repo % ls
[LICENSE README.md
[bienz@amandasworkmbp test_repo % git checkout -b develop
Switched to a new branch 'develop'
[bienz@amandasworkmbp test_repo % ls
[LICENSE README.md
[bienz@amandasworkmbp test_repo % ]
```

# Doing Some Work

```
[abienz@Amandas-MBP Documents % cd test_repo  
[abienz@Amandas-MBP test_repo % git checkout -b develop  
[Switched to a new branch 'develop'  
abienz@Amandas-MBP test_repo % ls  
[LICENSE README.md  
abienz@Amandas-MBP test_repo % vi test.cpp  
[abienz@Amandas-MBP test_repo % gcc -o test test.cpp  
[abienz@Amandas-MBP test_repo % ./test
```

# Checking Status

```
abienz@Amandas-MBP test_repo % vi test.cpp
[abienz@Amandas-MBP test_repo % gcc -o test test.cpp
[abienz@Amandas-MBP test_repo % ./test
[abienz@Amandas-MBP test_repo % git status
[On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test
    test.cpp
```

# Committing and Pushing Updates

```
abienz@Amandas-MBP test_repo % git add test.cpp
[abienz@Amandas-MBP test_repo % git commit -m "Added test file to develop branch"
[develop 6beabee] Added test file to develop branch
Committer: Amanda Bienz <abienz@Amandas-MBP.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:
```

```
git config --global --edit
```

After doing this, you may fix the identity used for this commit with:

```
git commit --amend --reset-author
```

```
1 file changed, 7 insertions(+)
create mode 100644 test.cpp
```

# Committing and Pushing Updates

```
[abienz@Amandas-MBP test_repo % git push --set-upstream origin develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 362 bytes | 362.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
```

# Merging a Branch

```
[abienz@Amandas-MBP test_repo % git merge develop
Updating ab165fe..6beabee
Fast-forward
  test.cpp | 7 ++++++
  1 file changed, 7 insertions(+)
  create mode 100644 test.cpp
abienz@Amandas-MBP test_repo %
```

```
[abienz@Amandas-MBP test_repo % git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:ProfessorBienz/test_repo.git
  ab165fe..6beabee  main -> main
abienz@Amandas-MBP test_repo %
```

# File : .gitignore

- All files that should not be committed and pushed to GitHub



A screenshot of a Mac OS X desktop showing a Vim window titled "test\_repo — Vim .gitignore — 168x51". The window contains a .gitignore file with the following content:

```
# Prerequisites
*.d

# Compiled Object files
*.slo
*.lo
*.o
*.obj

# Precompiled Headers
*.gch
*.pch

# Compiled Dynamic libraries
*.so
*.dylib
*.dll

# Fortran module files
*.mod
*.smod

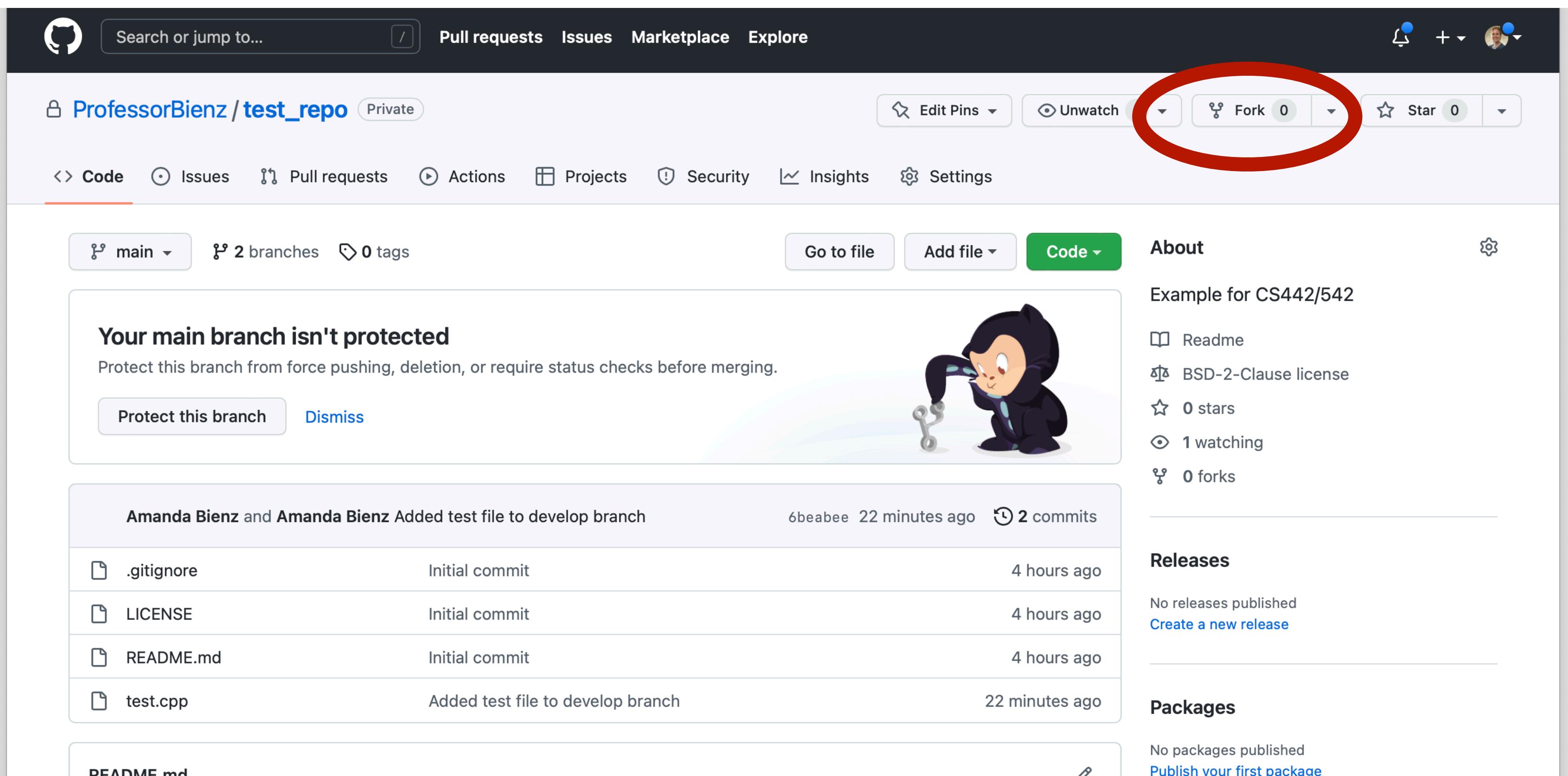
# Compiled Static libraries
*.lai
*.la
*.a
*.lib

# Executables
*.exe
*.out
*.app

# Binary Folder
build
~
```

# Forking a Repo

- Create a new repo from an existing repo



# Submodules

- Include a git repository within your git repository

```
[abienz@Amandas-MBP test_repo % git submodule add git@github.com:ProfessorBienz/tutorial_src.git
Cloning into '/Users/bienz/Documents/test_repo/tutorial_src'...
remote: Enumerating objects: 206, done.
remote: Counting objects: 100% (206/206), done.
remote: Compressing objects: 100% (140/140), done.
remote: Total 206 (delta 64), reused 195 (delta 57), pack-reused 0
Receiving objects: 100% (206/206), 601.85 KiB | 1.28 MiB/s, done.
Resolving deltas: 100% (64/64), done.
[abienz@Amandas-MBP test_repo % ls
LICENSE           README.md        test          test.cpp      tutorial_src
```

# Submodules

- Include a git repository within your git repository

```
:abienz@Amandas-MBP test_repo % git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)  
modified:   .gitignore  
new file:   .gitmodules  
new file:   tutorial_src
```

Untracked files:

```
(use "git add <file>..." to include in what will be committed)  
test
```

# File : .gitmodules



A screenshot of a Vim window titled "test\_repo — Vim .gitmodules — 168x51". The window displays the following text:

```
[submodule "tutorial_src"]
  path = tutorial_src
  url = git@github.com:ProfessorBienz/tutorial_src.git
```

# Submodules

- Include a git repository within your git repository

```
[abienz@Amandas-MBP test_repo % git commit -m "Added submodule"
[main e05c3eb] Added submodule
Committer: Amanda Bienz <abienz@Amandas-MBP.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:
```

```
git config --global --edit
```

After doing this, you may fix the identity used for this commit with:

```
git commit --amend --reset-author
```

```
3 files changed, 7 insertions(+)
create mode 100644 .gitmodules
create mode 160000 tutorial_src
```

# Submodules

- Include a git repository within your git repository

```
[abienz@Amandas-MBP test_repo % git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 559 bytes | 559.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:ProfessorBienz/test_repo.git
```

# Submodules

Screenshot of a GitHub repository page for **ProfessorBienz/test\_repo** (Private).

The repository has **2 branches** and **0 tags**.

A message on the main branch states: **Your main branch isn't protected**. It suggests protecting the branch from force pushing, deletion, or requiring status checks before merging.

The commit history shows:

- Amanda Bienz Added submodule (e05c3eb, 3 minutes ago)
- tutorial\_src @ 4a5fdc5 (3 minutes ago)
- .gitignore (3 minutes ago)
- .gitmodules (3 minutes ago)
- LICENSE (4 hours ago)
- README.md (4 hours ago)
- test.cpp (28 minutes ago)

A red circle highlights the commit **tutorial\_src @ 4a5fdc5**.

**About** section:

- Example for CS442/542
- Readme
- BSD-2-Clause license
- 0 stars
- 1 watching
- 0 forks

**Releases**: No releases published. [Create a new release](#)

**Packages**: No packages published. [Publish your first package](#)

**Languages**

# Submodule : Folder is Empty? !

- If I clone my branch (say on a new computer), the submodule is an empty folder...

```
[abienz@Amandas-MBP Documents % git clone git@github.com:ProfessorBienz/test_repo.git
Cloning into 'test_repo'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 15 (delta 4), reused 11 (delta 3), pack-reused 0
Receiving objects: 100% (15/15), done.
Resolving deltas: 100% (4/4), done.
```

```
[abienz@Amandas-MBP Documents % cd test_repo
[abienz@Amandas-MBP test_repo % ls tutorial_src
abienz@Amandas-MBP test_repo % █
```

# Initialize and Update Submodules

- One fix : 'git clone --recurse-submodules <repo-name>
- Otherwise if already cloned and have empty submodule:

```
[abienz@Amandas-MBP test_repo % git submodule update --init --recursive
Submodule 'tutorial_src' (git@github.com:ProfessorBienz/tutorial_src.git) registered for path 'tutorial_src'
Cloning into '/Users/bienz/Documents/test_repo/tutorial_src'...
ls Submodule path 'tutorial_src': checked out '4a5fdc53a2cc2b7ad276bf01a1c4af1ca30bed7c'
[abienz@Amandas-MBP test_repo % ls tutorial_src
CMakeLists.txt  LICENSE          tests           tutorial.h
abienz@Amandas-MBP test_repo %
```

# CMake : <https://cmake.org>

- Tool for building and testing code
- Helpful if you want users to run your code on various machines
- Necessary for users to be able to run your code on supercomputers
  - *Also, makes it easier for you to run your code on multiple supercomputers*

# Install CMake

- Linux : `sudo apt-get -y install cmake`
- Mac : `brew install cmake`
- Supercomputer : should already exist
  - If not, or need different version, `spack` (**to be discussed in a later class**)
- Binaries also available on [cmake.org](http://cmake.org)

# CMakeLists.txt

- File that ‘cmake’ command looks for
- This tells cmake how to compile your programs

```
cmake_minimum_required(VERSION 3.10)

# Project Name
project(test_project)

# Executable
add_executable(test test.cpp)
~
```

# Compiling Program with CMake

```
[abienz@Amandas-MBP test_repo % mkdir build
[abienz@Amandas-MBP test_repo % cd build
[abienz@Amandas-MBP build % cmake ..
-- The C compiler identification is AppleClang 13.1.6.13160021
-- The CXX compiler identification is AppleClang 13.1.6.13160021
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/bienz/Documents/test_repo/build
[abienz@Amandas-MBP build % make
[ 50%] Building CXX object CMakeFiles/test.dir/test.cpp.o
[100%] Linking CXX executable test
[100%] Built target test
abienz@Amandas-MBP build % ]
```

# Enable Languages

- Tell cmake I want to be able to compile programs with these languages
- Requires cmake to find these compilers

```
cmake_minimum_required(VERSION 3.10)

# Project Name
project(test_project)

# Enable Languages
enable_language(C)
enable_language(CXX)
enable_language(Fortran)

# Executable
add_executable(test test.cpp)
~
```

# Specify Language Standards

```
cmake_minimum_required(VERSION 3.10)

# Project Name
project(test_project)

# Enable Languages
enable_language(C)
enable_language(CXX)
enable_language(Fortran)

# Require C++11, C99
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_C_STANDARD 99)

# Executable
add_executable(test test.cpp)
~
```

# Finding a Package

```
cmake_minimum_required(VERSION 3.10)

# Project Name
project(test_project)

# Enable Languages
enable_language(C)
enable_language(CXX)
enable_language(Fortran)

# Require C++11, C99
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_C_STANDARD 99)

# Find PythonLibs package, and require it be found
find_package(PythonLibs REQUIRED)

# Executable
add_executable(test test.cpp)
~
```

# Include Directories

- To use Python, must include the following directories

```
cmake_minimum_required(VERSION 3.10)

# Project Name
project(test_project)

# Enable Languages
enable_language(C)
enable_language(CXX)
enable_language(Fortran)

# Require C++11, C99
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_C_STANDARD 99)

# Find PythonLibs package, and require it be found
find_package(PythonLibs REQUIRED)

# Include directories so that I can run Python
include_directories(${PYTHON_INCLUDE_DIRS})
include_directories(SYSTEM ${MPI_INCLUDE_PATH})

# Executable
add_executable(test test.cpp)
```

# Create Library

---

```
enable_language(Fortran)

# Require C++11, C99
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_C_STANDARD 99)

# Find PythonLibs package, and require it be found
find_package(PythonLibs REQUIRED)

# Include directories so that I can run Python
include_directories(${PYTHON_INCLUDE_DIRS})
include_directories(SYSTEM ${MPI_INCLUDE_PATH})

# Create Library
add_library(testlib STATIC test.cpp)
```

# Link Libraries

---

```
enable_language(Fortran)

# Require C++11, C99
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_C_STANDARD 99)

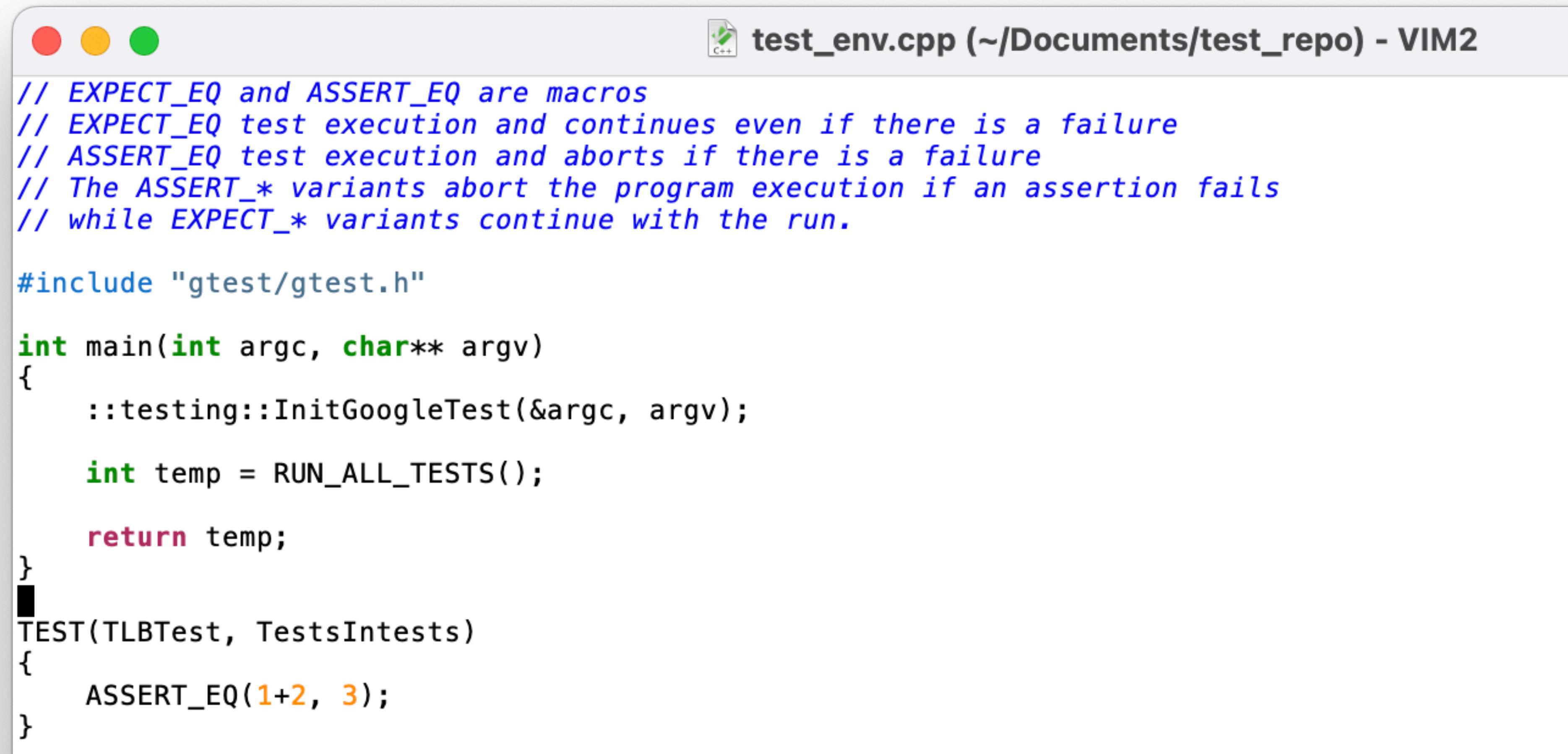
# Find PythonLibs package, and require it be found
find_package(PythonLibs REQUIRED)

# Include directories so that I can run Python
include_directories(${PYTHON_INCLUDE_DIRS})
include_directories(SYSTEM ${MPI_INCLUDE_PATH})

# Create Library
add_library(testlib STATIC test.cpp)

# Link libraries needed by testlib
target_link_libraries(testlib m)
```

# Googletest



The screenshot shows a Vim 2 window titled "test\_env.cpp (~/Documents/test\_repo) - VIM2". The Vim status bar at the top indicates three marks: a red dot, a yellow dot, and a green dot. The code editor displays the following C++ code:

```
// EXPECT_EQ and ASSERT_EQ are macros
// EXPECT_EQ test execution and continues even if there is a failure
// ASSERT_EQ test execution and aborts if there is a failure
// The ASSERT_* variants abort the program execution if an assertion fails
// while EXPECT_* variants continue with the run.

#include "gtest/gtest.h"

int main(int argc, char** argv)
{
    ::testing::InitGoogleTest(&argc, argv);

    int temp = RUN_ALL_TESTS();

    return temp;
}

TEST(TLBTest, TestsIntests)
{
    ASSERT_EQ(1+2, 3);
}
```

# CMakeLists.txt Gooletest

```
# Fetch GoogleTest content
FetchContent_Declare(
    googletest
    GIT_REPOSITORY https://github.com/google/googletest.git
    GIT_TAG        5376968f6948923e2411081fd9372e71a59d8e77
)

# Make content available
FetchContent_MakeAvailable(googletest)

# Enable testing
enable_testing()
```

# Testing Environment

```
# Enable Testing (Command 'make test' runs testing environment)
enable_testing()
add_executable(test_env test_env.cpp)
target_link_libraries(test_env testlib gtest_main)
add_test(EnvironmentTest ${CMAKE_BINARY_DIR}/test_env)
~
```

# Testing Environment

```
[abienz@Amandas-MBP build % make test
Running tests...
Test project /Users/bienz/Documents/test_repo/build
  Start 1: EnvironmentTest
1/1 Test #1: EnvironmentTest ..... Passed    0.15 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.16 sec
abienz@Amandas-MBP build %
```

# Remember Back to the Submodule

```
[abienz@Amandas-MBP test_repo % git submodule add git@github.com:ProfessorBienz/tutorial_src.git
Cloning into '/Users/bienz/Documents/test_repo/tutorial_src'...
remote: Enumerating objects: 206, done.
remote: Counting objects: 100% (206/206), done.
remote: Compressing objects: 100% (140/140), done.
remote: Total 206 (delta 64), reused 195 (delta 57), pack-reused 0
Receiving objects: 100% (206/206), 601.85 KiB | 1.28 MiB/s, done.
Resolving deltas: 100% (64/64), done.
[abienz@Amandas-MBP test_repo % ls
LICENSE           README.md        test          test.cpp      tutorial_src
```

# Subdirectory CMakeLists.txt

```
cmake-and-github-tutorial-bienz2 ~]$ ls tutorial_src  
CMakeLists.txt LICENSE tests tutorial.h  
cmake-and-github-tutorial-bienz2 ~]$ mvim tutorial_src/CMakeLists.txt  
cmake-and-github-tutorial-bienz2 ~]$
```

CMakeLists.txt (~/Library/CloudBees/CloudBees CloudBees CloudBees)

```
set(CMAKE_INCLUDE_CURRENT_DIR ON)  
  
set(src_SOURCES  
    tutorial_src/tutorial.h  
    PARENT_SCOPE  
)  
~  
~  
~  
~  
~
```

# Adding Subdirectories

```
# Add subdirectory
add_subdirectory(tutorial_src)

# Create library
add_library(tutorial STATIC ${src_SOURCES} C/tutorial.c)

# Link math library
target_link_libraries(tutorial m)

#set_target_properties(tutorial PROPERTIES LINKER_LANGUAGE CXX)

# Add tests subdirectory
add_subdirectory(tutorial_src/tests)
~
```

# Subdirectory CMakeLists.txt

```
bienz@Amandas-MacBook-Pro-2 cmake-and-github-tutorial-bienz2 % ls tutorial_src/tests
CMakeLists.txt      tutorial_test.cpp
[bienz@Amandas-MacBook-Pro-2 cmake-and-github-tutorial-bienz2 % mvim tutorial_src/tests/CMakeList]
s.txt
[bienz@Amandas-MacBook-Pro-2 cmake-and-github-tutorial-bienz2 % mvim tutorial_src/tests/tutorial_
test.cpp
bienz@Amandas-MacBook-Pro-2 cmake-and-github-tutorial-bienz2 % ]
```

```
include_directories(${tutorial_INCDIR})
include_directories(${CMAKE_SOURCE_DIR})

add_executable(tutorial_test tutorial_test.cpp)
target_link_libraries(tutorial_test tutorial gtest_main)
add_test(TutorialTest ${CMAKE_BINARY_DIR}/tutorial_src/tests/tutorial_test)

~
~
~
```

```
// EXPECT_EQ and ASSERT_EQ are macros
// EXPECT_EQ test execution and continues even if there is a failure
// ASSERT_EQ test execution and aborts if there is a failure
// The ASSERT_* variants abort the program execution if an assertion fails
// while EXPECT_* variants continue with the run.

#include "gtest/gtest.h"
#include "tutorial_src/tutorial.h"

int main(int argc, char** argv)
{
    ::testing::InitGoogleTest(&argc, argv);

    int temp = RUN_ALL_TESTS();

    return temp;
}

TEST(TLBTest, TestsIntests)
{
    ASSERT_EQ(0, tutorial());
}
```

# Other Useful CMake Links

- **Finding a package** : [https://cmake.org/cmake/help/latest/command/find\\_package.html](https://cmake.org/cmake/help/latest/command/find_package.html)
- **Debug vs Release** : <https://cmake.org/cmake/help/latest/guide/tutorial/Packaging%20Debug%20and%20Release.html>
- **Running ccmake** : <https://cmake.org/cmake/help/latest/manual/ccmake.1.html>
- **Online cmake tutorial** : <https://cmake.org/cmake/help/latest/guide/tutorial/index.html>
- **Messages** : <https://cmake.org/cmake/help/latest/command/message.html>