Git/GitHub

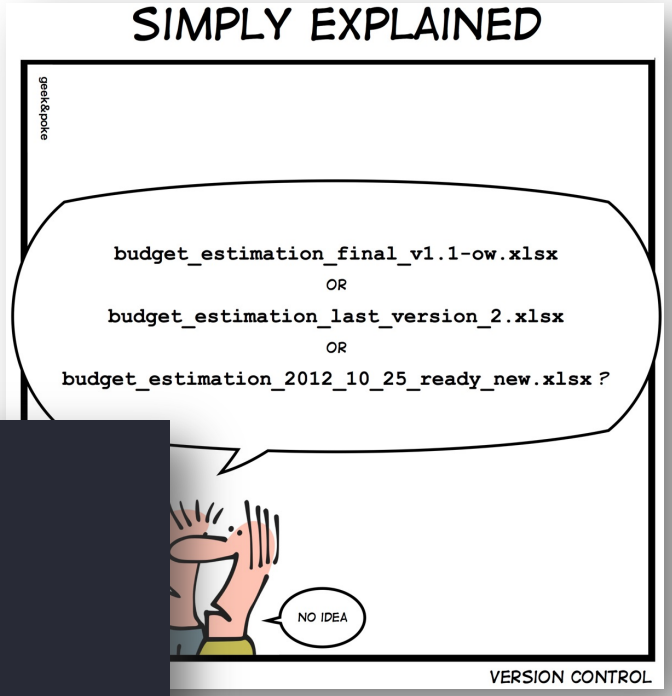
UNIVERSITY of WASHINGTON

What do Git and GitHub do?

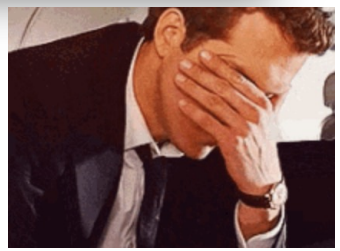
Changes

Manage Changes

Versions, so many versions




```
Main.java
Main_Tuesday.java
Main_Tuesday_ver1.java
Main_Tuesday_ver2.java
Main_Tuesday_ver2_11:59.java
Main_Wednesday_ver2_before_lunch.java
Main_Wednesday_ver2_before_lunch_doesntworkanymore.java
```



Source Control

Manage Changes

Version 1

```
int fib_space_inefficient(int n) {
    if (n <= 0) {
        return 0;
    }
    if (n == 1) {
        return 1;
    }

    int *data = new int[n + 1]; // why +1?
    data[0] = 0;
    data[1] = 1;

    for (int i = 2; i <= n; ++i) {
        data[i] = data[i - 1] + data[i - 2];
    }

    int ret = data[n];

    delete[]data;

    return ret;
}
```

• • • • • •

Version 2

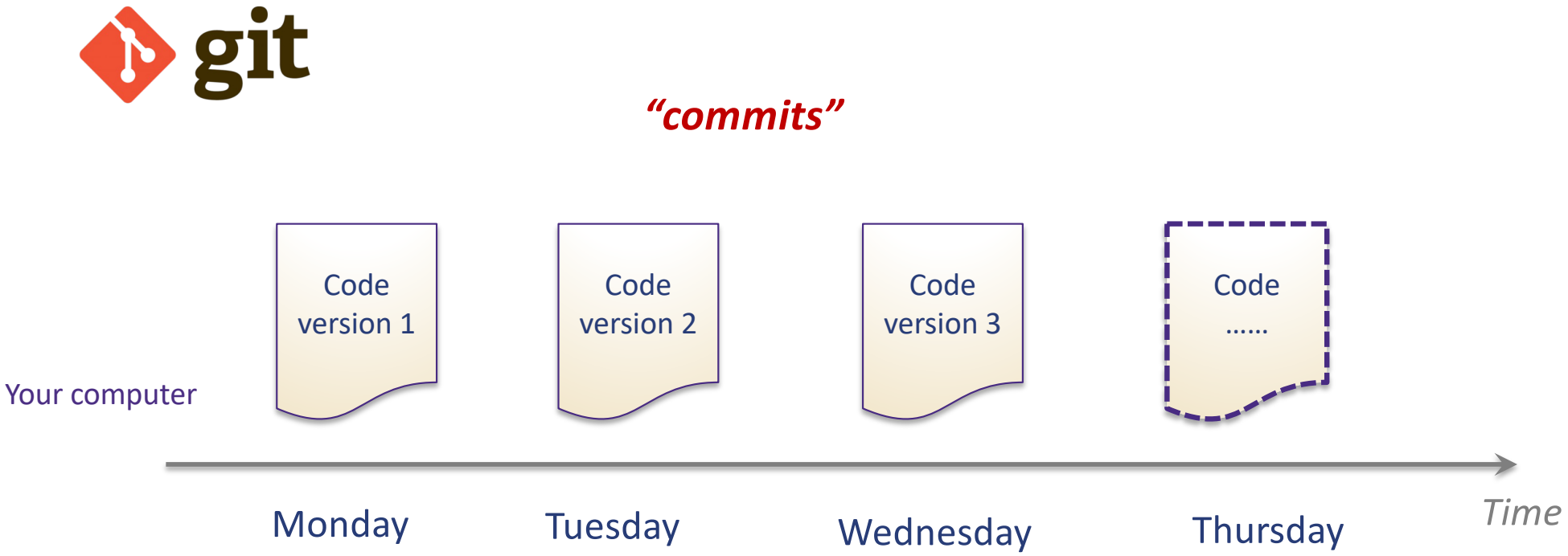
```
int fib_space_efficient(int n) {
    if (n <= 0) {
        return 0;
    }
    if (n == 1) {
        return 1;
    }
    int v1 = 0;
    int v2 = 1;

    // debug this i<n
    // use break point
    for (int i = 2; i <= n; ++i) {
        int v3 = v1 + v2;
        v1 = v2;
        v2 = v3;
    }

    return v2;
}
```

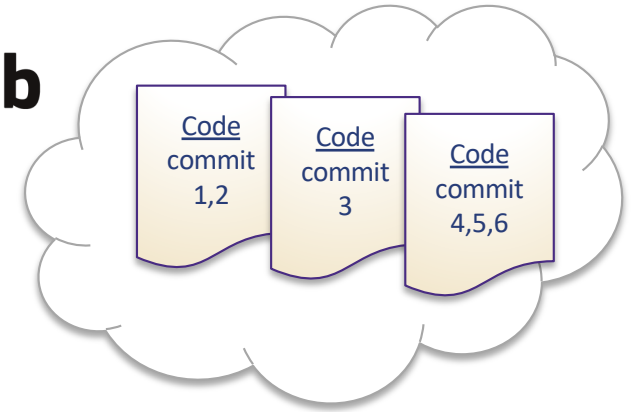
Monday Tuesday Wednesday Thursday

Why Git and GitHub

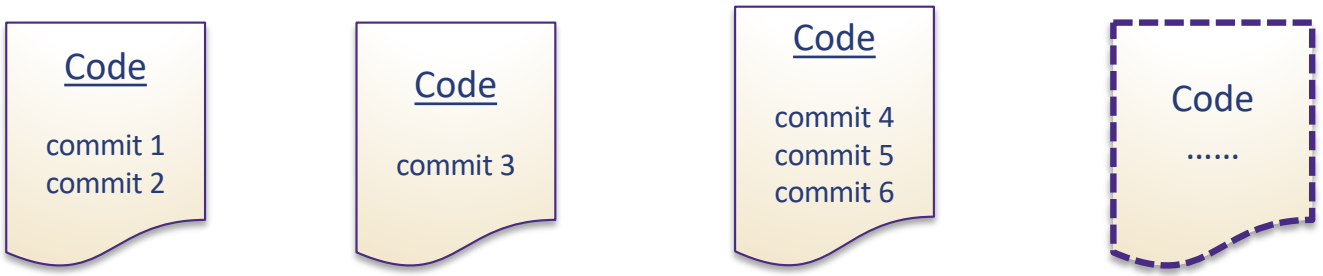
- Widely used in industry and open-source projects
- Easy to track **progress** (commits)
- Easy to track **features** (branch)
- Easy to **collaborate** with others (GitHub)

Source Control





Your computer



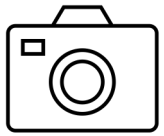
Git/GitHub Terms

Repository, or “Repo”

- Collection files from a project

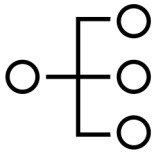
Commit

- A snapshot of the repo at the certain time



Branch

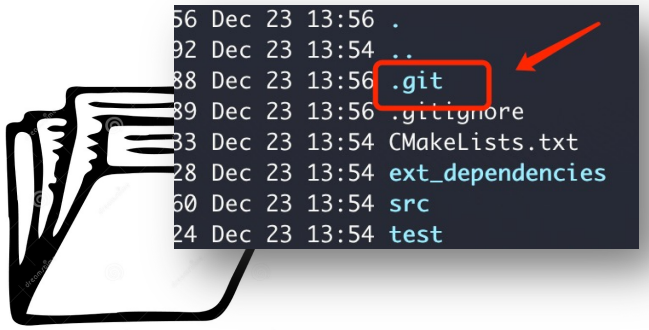
- A line of development, consisting of a series of commits



Find the start code here

```
.
├── .gitignore
├── CMakeLists.txt
├── ext_dependencies
│   ├── coverage
│   │   └── CodeCoverage.cmake
│   └── googletest
│       ├── googletest-download.cmake
│       └── googletest.cmake
├── src
│   ├── CMakeLists.txt
│   ├── fibonacci_sequence
│   │   ├── fib.cpp
│   │   └── fib.h
│   └── main.cpp
└── test
    ├── CMakeLists.txt
    ├── main.cpp
    ├── unit_test_fib.cpp
    ├── util.cpp
    └── util.h
```

Project folder



Git Repo

```
git init
```

"Hey Git! Manage this folder as a repo(sitory)!"

```
git add .
```

"Hey Git! Add all the files here to this repo"

```
git commit -am "my first commit"
```

"Hey Git! Make a commit (snapshot) of the repo"
"And use a commit message 'my first commit'"

Add unit tests

2

git diff

“Hey Git! Show me what’s changed but not committed since the last commit”

```
diff --git a/test/unit_test_fib.cpp b/test/unit_test_fib.cpp
index d8085be..c5f8843 100644
--- a/test/unit_test_fib.cpp
+++ b/test/unit_test_fib.cpp
@@ -3,9 +3,25 @@
 #include "util.h"

 TEST(fib, edge_cases) {
-    ASSERT_TRUE(false); // TODO: replace this with real test code
+
+    for (int i = -10; i < 0; i++) {
+        int actual = fib(i);
+
+        ASSERT_EQ(0, actual) << case_string(i);
+    }
 }

 TEST(fib, normal_cases) {
-    ASSERT_TRUE(false); // TODO: replace this with real test code
+    int expect[] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
+                    89, 144, 233, 377, 610, 987, 1597, 2584,
+                    4181, 6765, 10946, 17711, 28657, 46368, 75025,
+                    121393, 196418, 317811};
+
+    int size = sizeof(expect) / sizeof(expect[0]);
+
+    for (int i = 0; i < size; i++) {
+        int actual = fib(i);
+
+        ASSERT_EQ(expect[i], actual) << case_string(i);
+    }
```

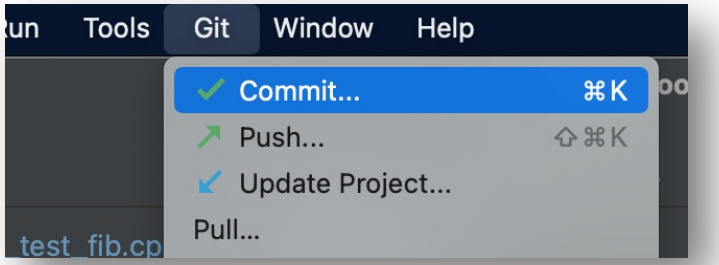
result in command line

```
#include "util.h"

TEST(fib, edge_cases) {
    ASSERT_TRUE(false); // TODO: replace this with real test code

    for (int i = -10; i < 0; i++) {
        int actual = fib(i);

        ASSERT_EQ(0, actual) << case_string(i);
    }
}

TEST(fib, normal_cases) {
    ASSERT_TRUE(false); // TODO: replace this with real test code
    int expect[] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
                    89, 144, 233, 377, 610, 987, 1597, 2584,
                    4181, 6765, 10946, 17711, 28657, 46368, 75025,
                    121393, 196418, 317811};

    int size = sizeof(expect) / sizeof(expect[0]);

    for (int i = 0; i < size; i++) {
        int actual = fib(i);

        ASSERT_EQ(expect[i], actual) << case_string(i);
    }
}
```

result in IDE (CLion)

```
git commit -am “add unit tests"
```

from in command line



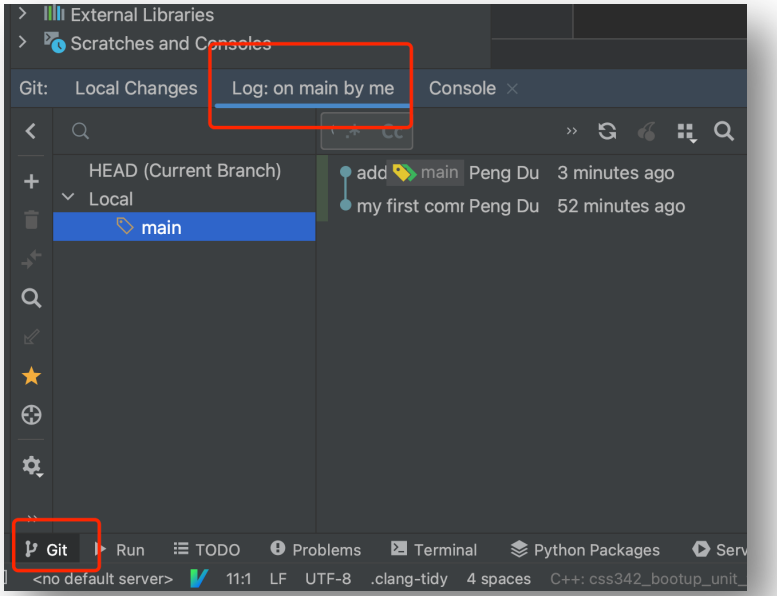
from IDE (CLion)

“Hey Git! Commit with message ’add unit tests’”

git log

from in command line

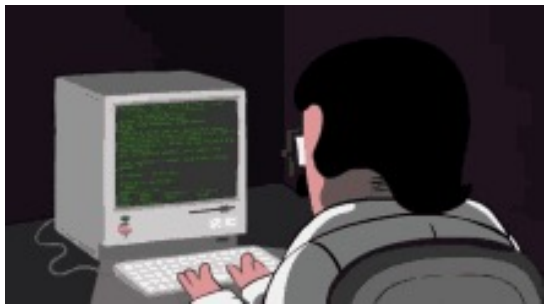
```
commit 172a207430d8073b8e9f7f4b5e731000fa44d564 (HEAD -> main)
Author: Peng Du <rick.peng.du@gmail.com>
Date:   Fri Dec 23 15:02:28 2022 -0800

    add unit tests

commit 782234bd01d63d35c552a6148ec4e921777d590b
Author: Peng Du <rick.peng.du@gmail.com>
Date:   Fri Dec 23 14:12:55 2022 -0800

    my first commit
```



from IDE (CLion)

"Hey Git! Shows me the list of commits so far'"

```
[==========] 2 tests from 1 test case ran. (1 ms total)
[  PASSED  ] 0 tests.
[  FAILED  ] 2 tests, listed below:
[  FAILED  ] fib.edge_cases
[  FAILED  ] fib.normal_cases
```



Write *fib(int n)*

```
[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
```

git commit -am ”finish fib"

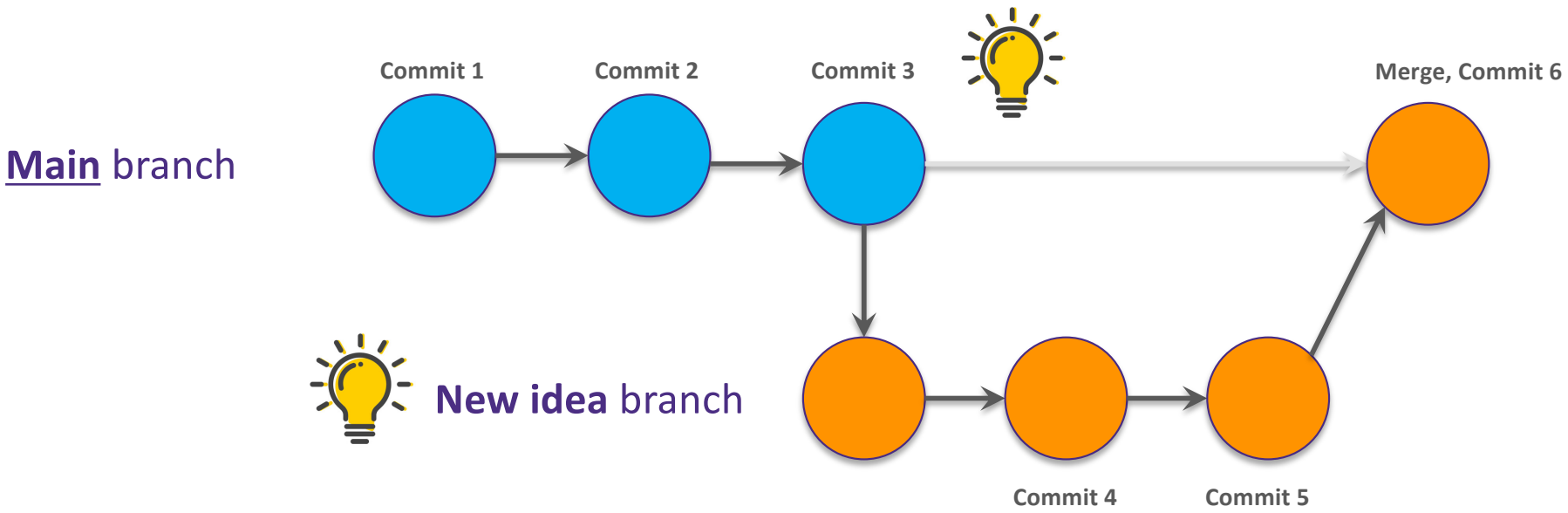
“Hey Git! Make a commit”

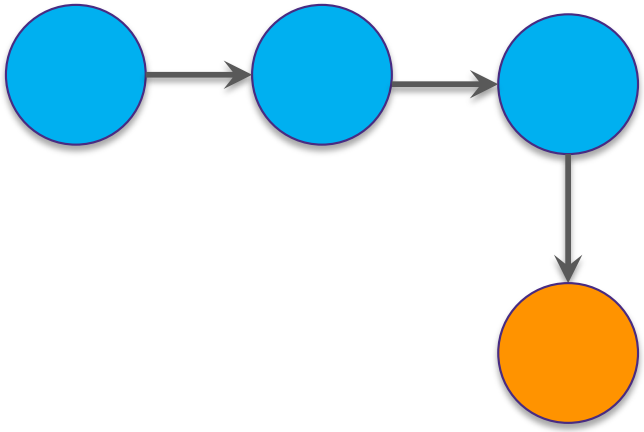
```
int fib(int n) {
    if (n <= 0) {
        return 0;
    }
    if (n == 1) {
        return 1;
    }

    int *data = new int[n + 1];
    data[0] = 0;
    data[1] = 1;

    for (int i = 2; i <= n; ++i) {
        data[i] = data[i - 1] + data[i - 2];
    }

    int ret = data[n];

    delete[]data;

    return ret;
}
```

0	1	1	2	3	5

New Idea: What if we don't use the array?

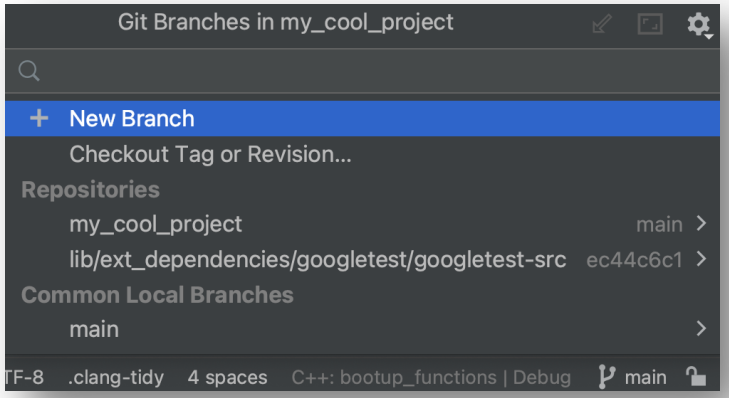
New idea? Test it in a new branch





```
git checkout -b new_idea
```

From command line



from IDE (CLion)

"Hey Git! Create and switch to a new branch called "new_idea"

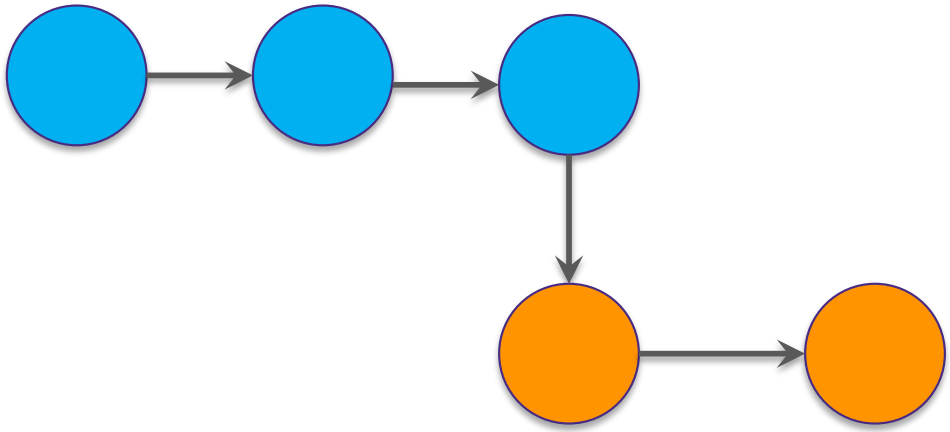
Write *fib(int n) with new idea*

```cpp
int fib(int n) {
    if (n <= 0) {
        return 0;
    }
    if (n == 1) {
        return 1;
    }

    int *data = new int[n + 1];
    data[0] = 0;
    data[1] = 1;

    for (int i = 2; i <= n; ++i) {
        data[i] = data[i - 1] + data[i - 2];
    }

    int ret = data[n];

    delete[]data;

    return ret;
}
```

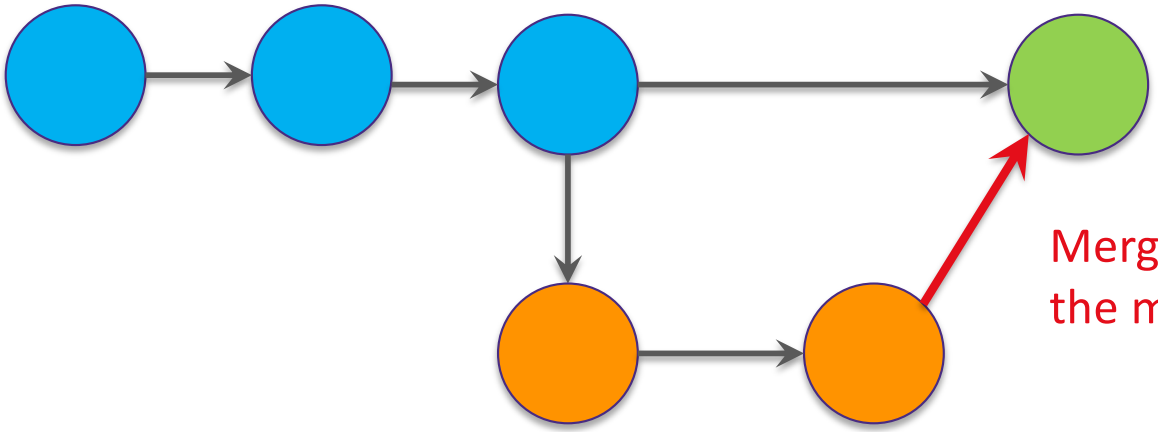
```
[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
```

```cpp
int fib(int n) {
    if (n <= 0) {
        return 0;
    }
    if (n == 1) {
        return 1;
    }
    int v1 = 0;
    int v2 = 1;

    for (int i = 2; i <= n; ++i) {
        int v3 = v1 + v2;
        v1 = v2;
        v2 = v3;
    }

    return v2;
}
```

```
[----------] Global test environment tear-down
[==========] 2 tests from 1 test case ran. (0 ms total)
[  PASSED  ] 2 tests.
```



```
git commit -am "finish fib with new idea"
```



Merge new idea into the main branch

git checkout main

"Hey Git! Switch back to the main branch"

git merge new_idea

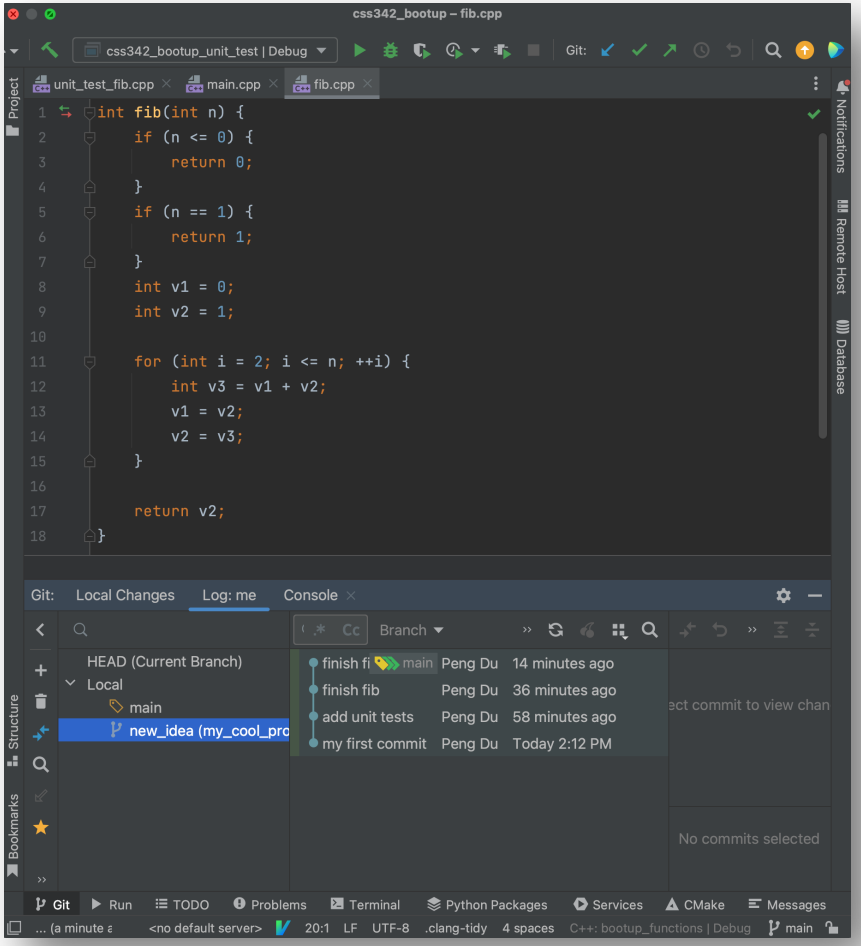
"Hey Git! Merge the code in new_idea branch to the current branch (main)"

```
Updating 025db6d..6e0ef0a
Fast-forward
 src/fibonacci_sequence/fib.cpp | 17 +++++++----------
 1 file changed, 7 insertions(+), 10 deletions(-)
```

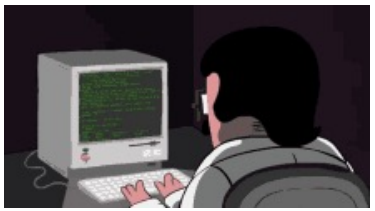
git log

```
commit 6e0ef0a2f26d38c96d049f4ac97b081efab4c043 (HEAD -> main, new_idea)
Author: Peng Du <rick.peng.du@gmail.com>
Date:   Fri Dec 23 15:46:10 2022 -0800

    finish fib with new idea

commit 025db6d5a8ee6db574d9d61621c7056ad20cc034
Author: Peng Du <rick.peng.du@gmail.com>
Date:   Fri Dec 23 15:24:14 2022 -0800

    finish fib

commit 172a207430d8073b8e9f7f4b5e731000fa44d564
Author: Peng Du <rick.peng.du@gmail.com>
Date:   Fri Dec 23 15:02:28 2022 -0800

    add unit tests

commit 782234bd01d63d35c552a6148ec4e921777d590b
Author: Peng Du <rick.peng.du@gmail.com>
Date:   Fri Dec 23 14:12:55 2022 -0800

    my first commit
(END)
```

from in command line



from IDE (CLion)



```
git init
```

```
git add .
```

```
git commit -am "commit message"
```

```
git diff
```

```
git log
```

```
git checkout -b new_idea
```

```
git merge new_idea
```


GitHub


git

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/a-teaching-goose/my_cool_project.git

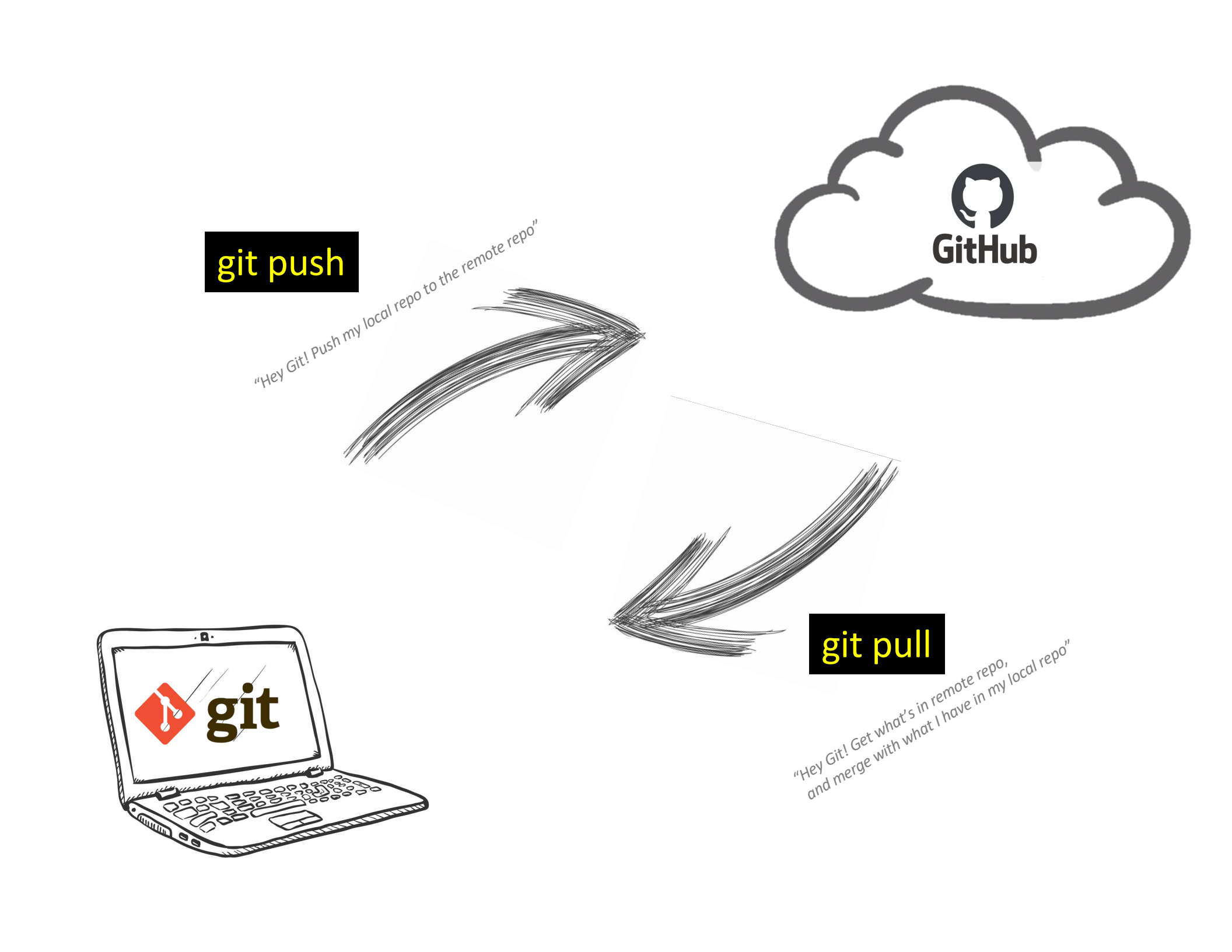
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

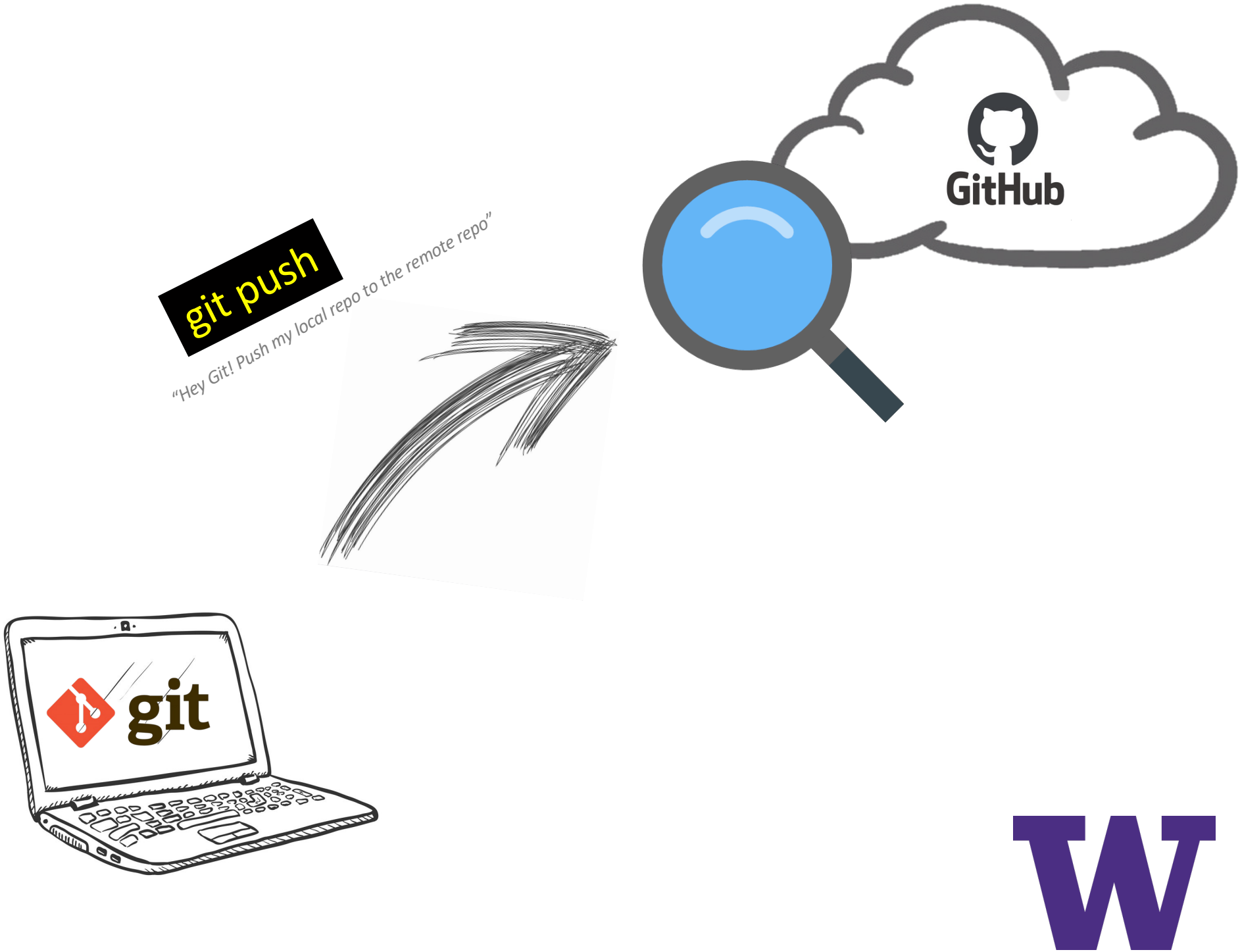
...or create a new repository on the command line

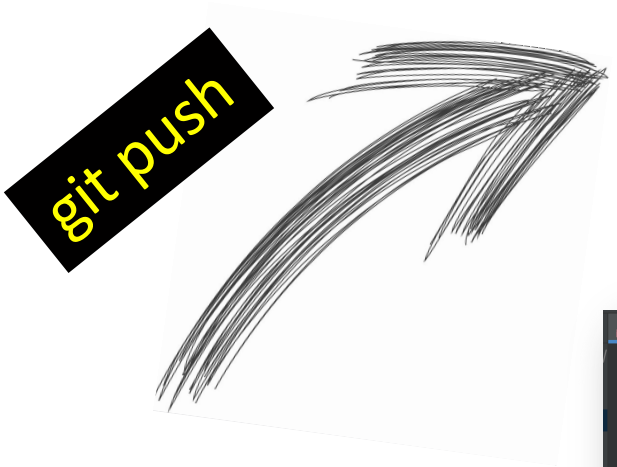
```
echo "# my_cool_project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/a-teaching-goose/my_cool_project.git
git push -u origin main
```

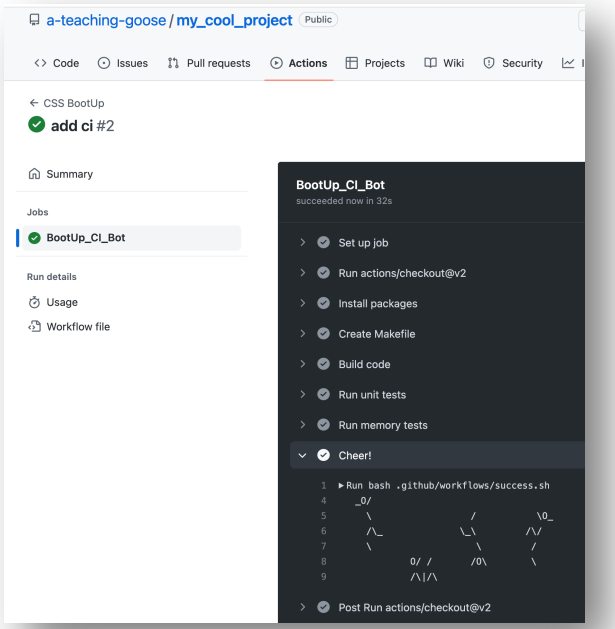

```
git remote add origin https://github.com/a-teaching-goose/my_cool_project.git
```

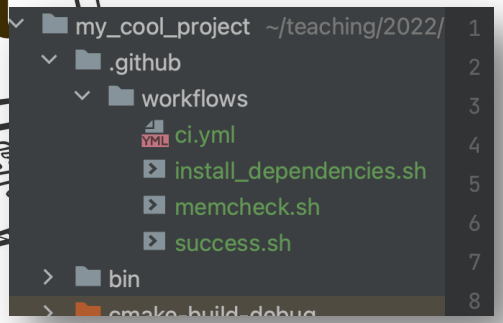
“Hey Git! Connect my local repo to a remote repo with the provided url, and I’ll use ‘origin’ on my laptop to refer to this remote repo”


GitHub
git push
"Hey Git! Push my local repo to the remote repo"
git pull
"Hey Git! Get what's in remote repo,
and merge with what I have in my local repo"
git

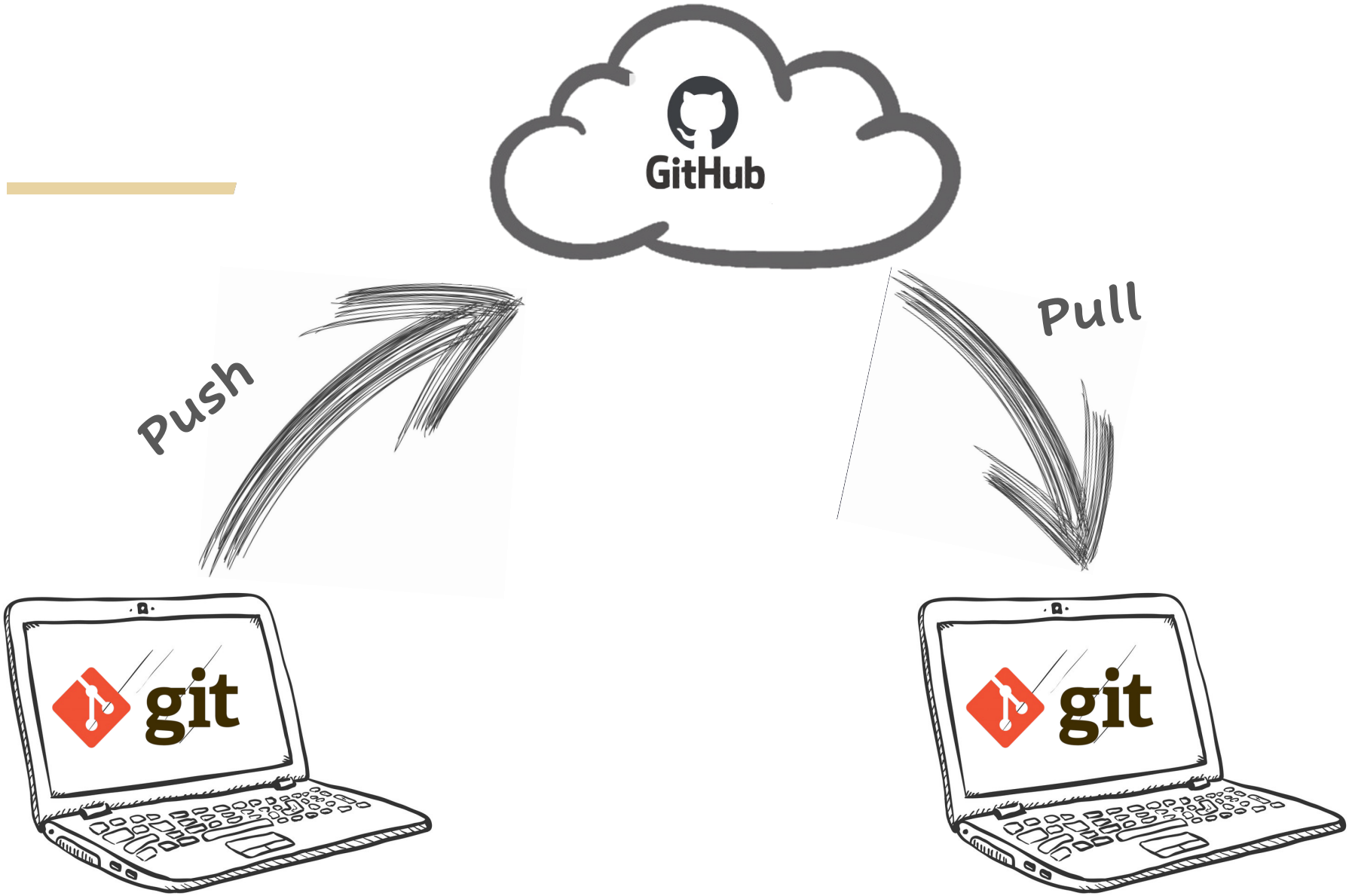

GitHub
git push
"Hey Git! Push my local repo to the remote repo"
git



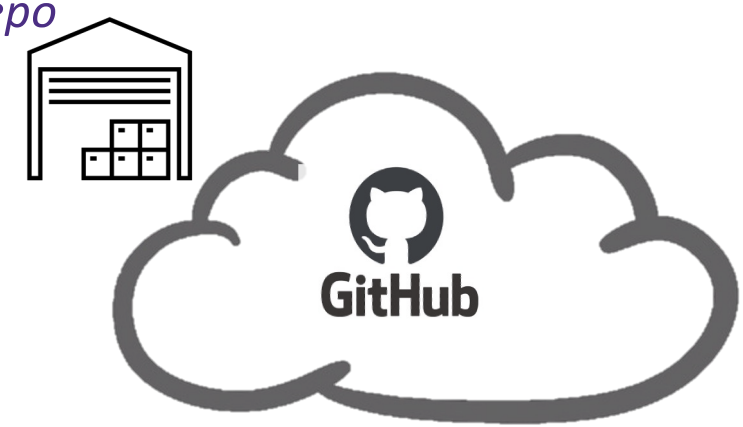





```yaml
# BootUp for CSS 342

name: CSS BootUp

on:
  push:
    branches: [ main ]

jobs:
  BootUp_CI_Bot:

    runs-on: ubuntu-18.04

    steps:
      - uses: actions/checkout@v2
      - name: Install packages
        run: .github/workflows/install_dependencies.sh
      - name: Create Makefile
        run: cmake -DCMAKE_BUILD_TYPE=Debug .
      - name: Build code
        run: make -j4
      - name: Run unit tests
        run: make function_tests # run all executables ending with "_test"
      - name: Run memory tests
        run: .github/workflows/memcheck.sh
      - name: Cheer!
        run: bash .github/workflows/success.sh
```


GitHub
Push
Pull
git
git

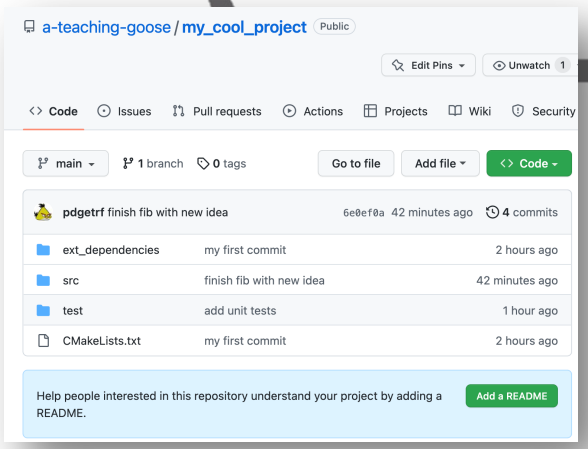
Remote repo



Exact Mirror

Local repo

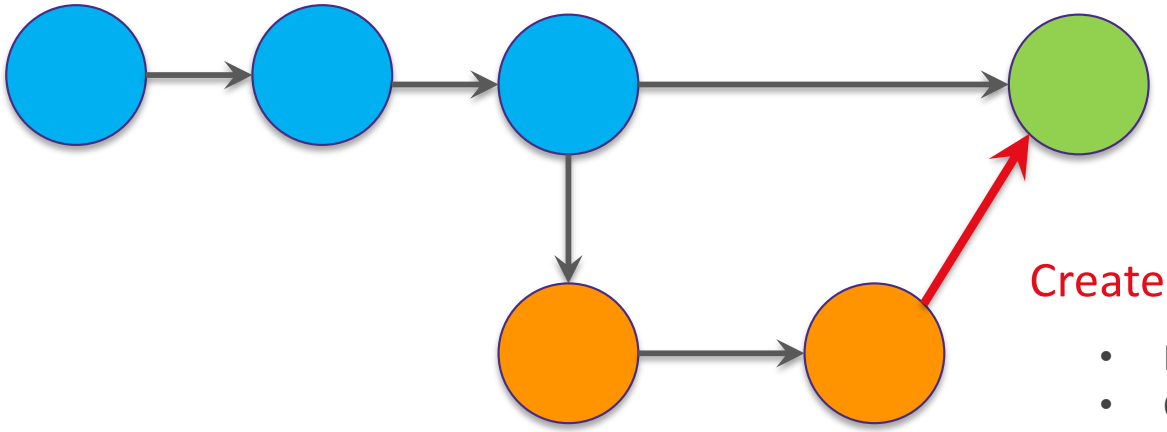

GitHub


a-teaching-goose / my_cool_project
Public
Edit Pins
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
main
1 branch
0 tags
Go to file
Add file
Code
pdgetrf finish fib with new idea
6e0ef0a 42 minutes ago
4 commits
ext_dependencies my first commit 2 hours ago
src finish fib with new idea 42 minutes ago
test add unit tests 1 hour ago
CMakeLists.txt my first commit 2 hours ago
Help people interested in this repository understand your project by adding a README.
Add a README


git
CLion File Edit View Navigate Code Refactor Build Run Tools Git
my_cool_project
Project
my_cool_project ~/teaching/2022/2022_UWB_BootUp/my_cool_project
bin
cmake-build-debug
ext_dependencies
lib
src
test
CMakeLists.txt
External Libraries
Scratches and Consoles


GIT MERGE

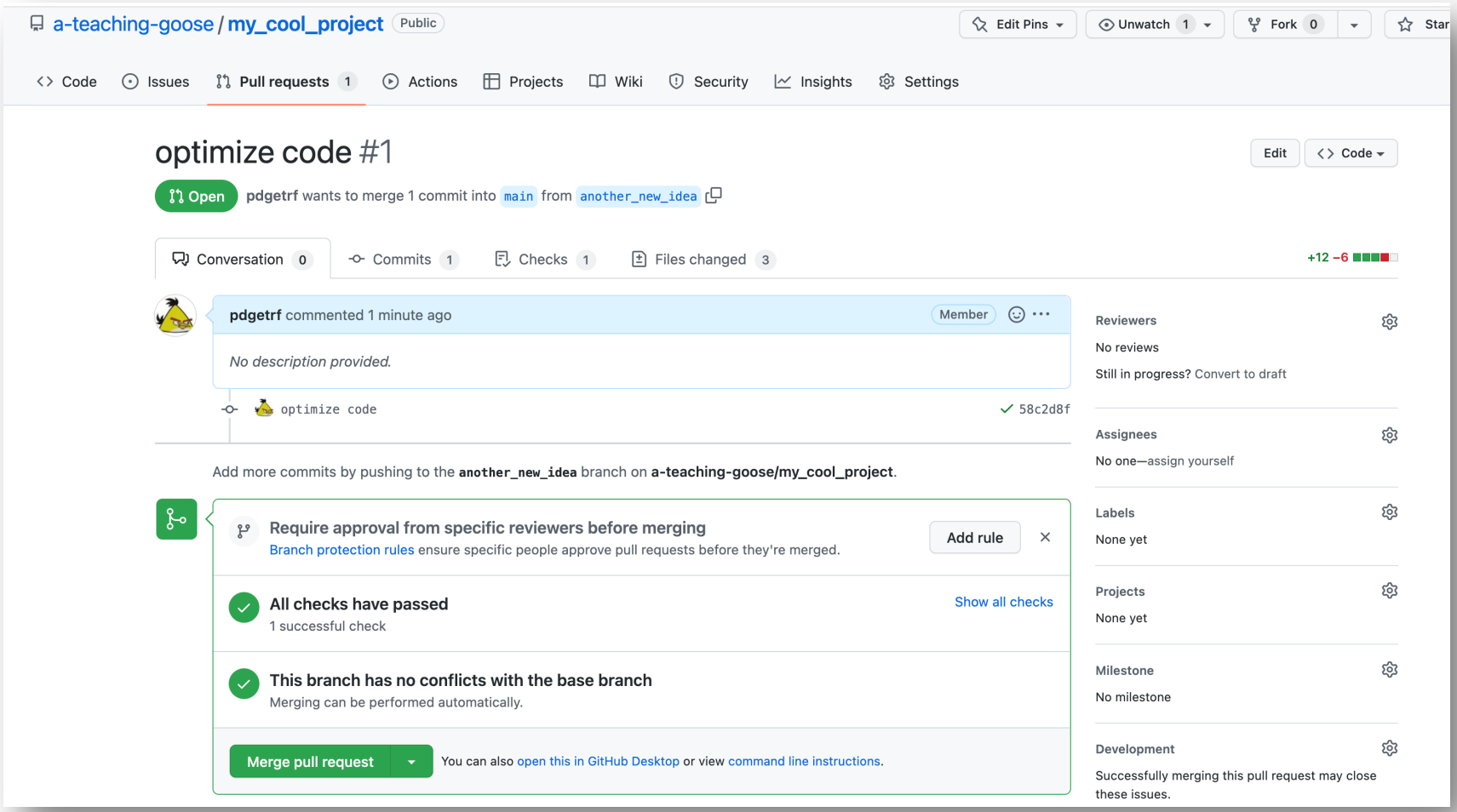
GitHub

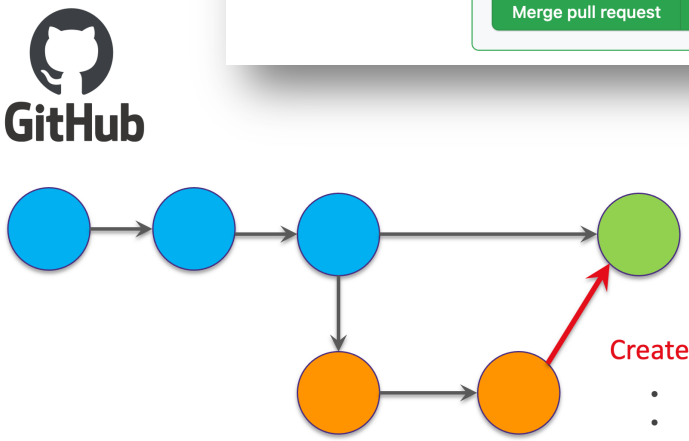


Create a pull request (for merge)

- Run all unit tests
- Code review by colleagues happens here
- Need someone else would approve this

Find the finished code here

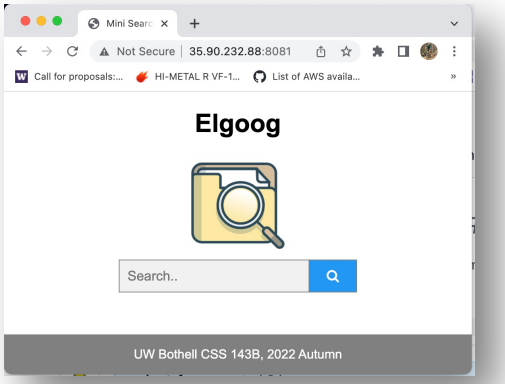




Create a pull request (for merge)

- Run all unit tests
- Code review by colleagues happens here
- Need someone else would approve this

CI/CD



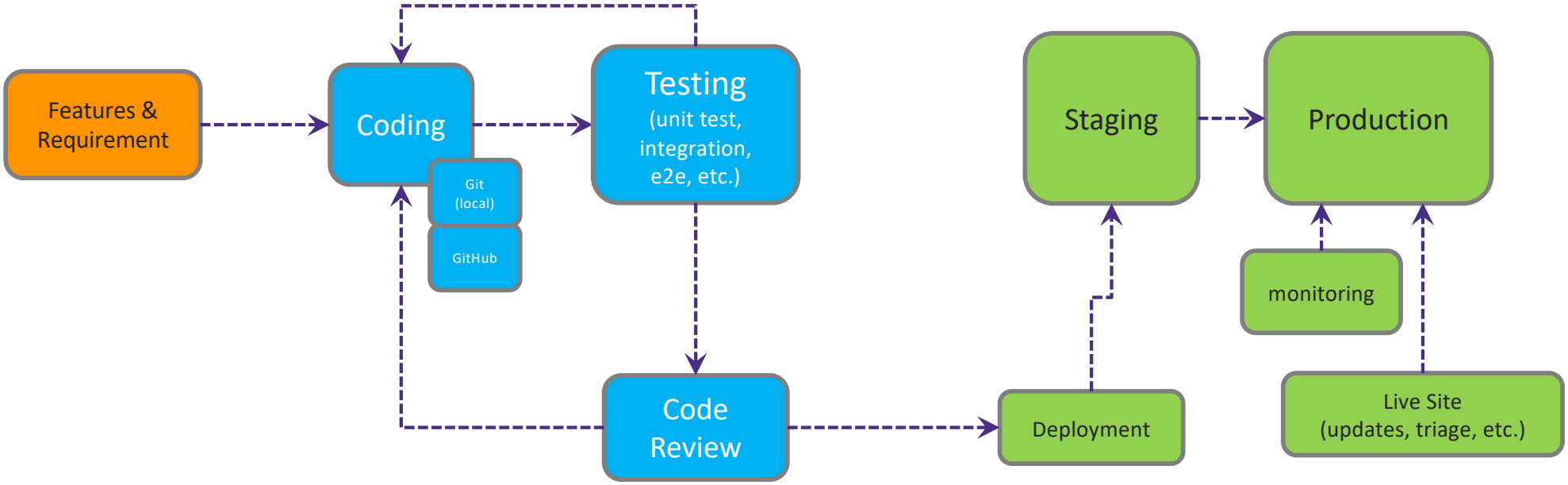
Clients/customers

”production site”

Tests (unit test, end-to-end test)

Your new code changes

Continuous Deployment Workflow (Generic)



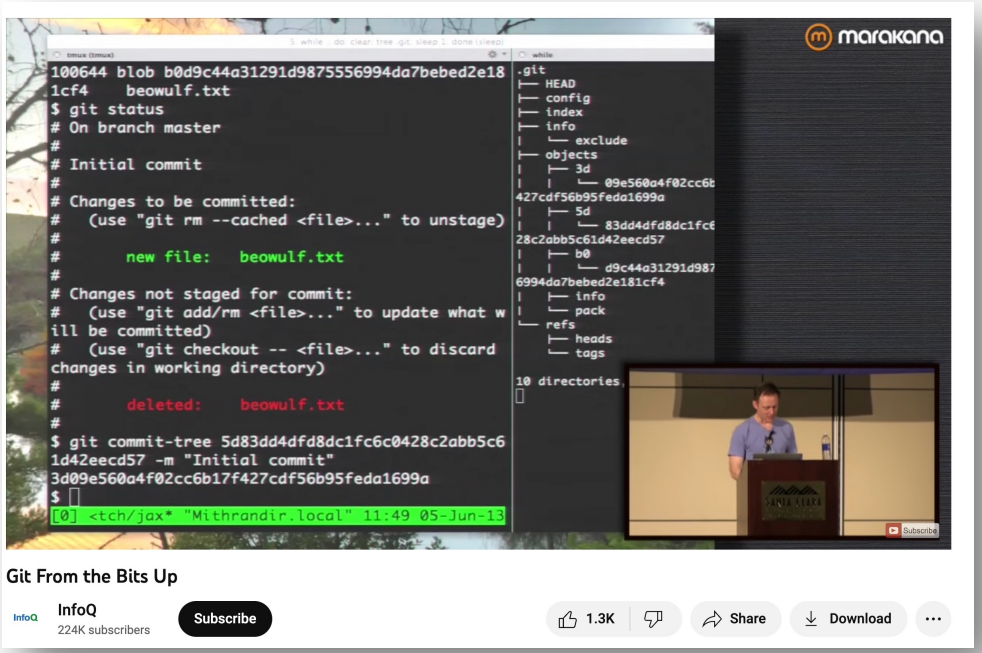

```
git push
git pull
```

```
git checkout -b new_idea
git merge new_idea
```

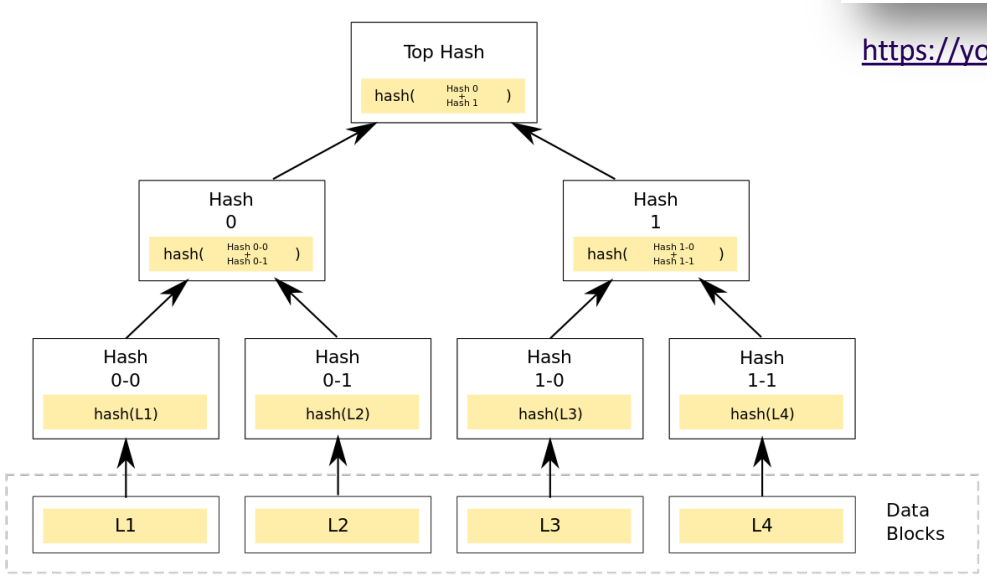
```
git init
git add .
git commit -am "commit message"
```

```
git diff
git log
```

Git Internals



https://youtu.be/MYP56QJpDr4



https://en.wikipedia.org/wiki/Merkle_tree

#CodeSafely

Who ate your homework?

“My laptop crashed last night, and my homework is gone”