

Code Archeology

17-313: Foundations of Software Engineering

<https://cmu-313.github.io>

Michael Hilton and Josh Sunshine

Spring 2026

Administrivia

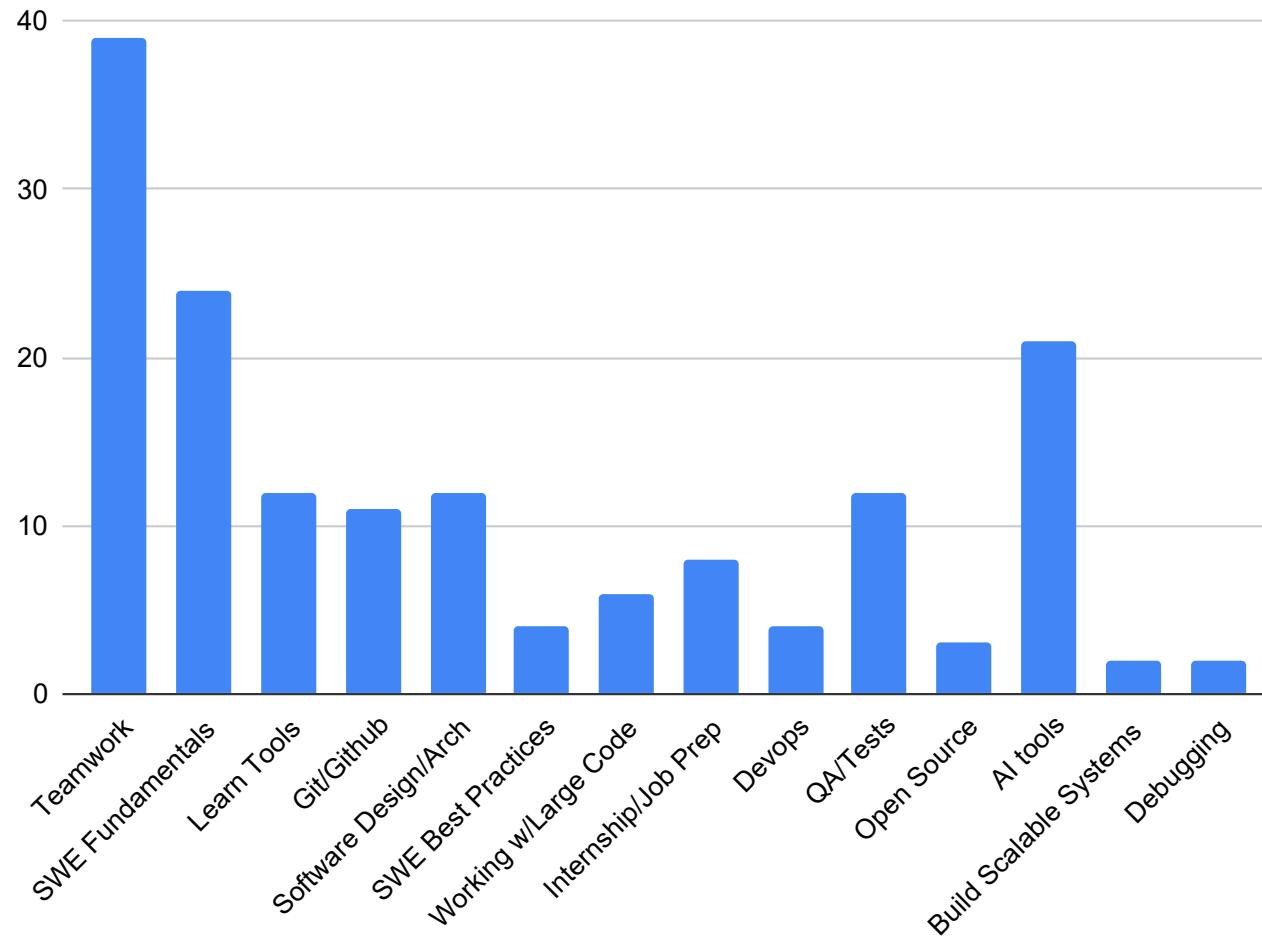
- **Project 1A - Build Checkpoint**
Due Friday, January 16, 2026 @ 11:59PM
- If you haven't: **PLEASE FILL OUT TEAMWORK SURVEY!**
 - Posted to slack and canvas
 - Get started early, ask for help, and check the #technical-support channel; chances are your questions have been asked by others!
 - NO RECITATION MONDAY (MLK Day)

Smoking Section

- Last full row



Hope to learn (from last class)

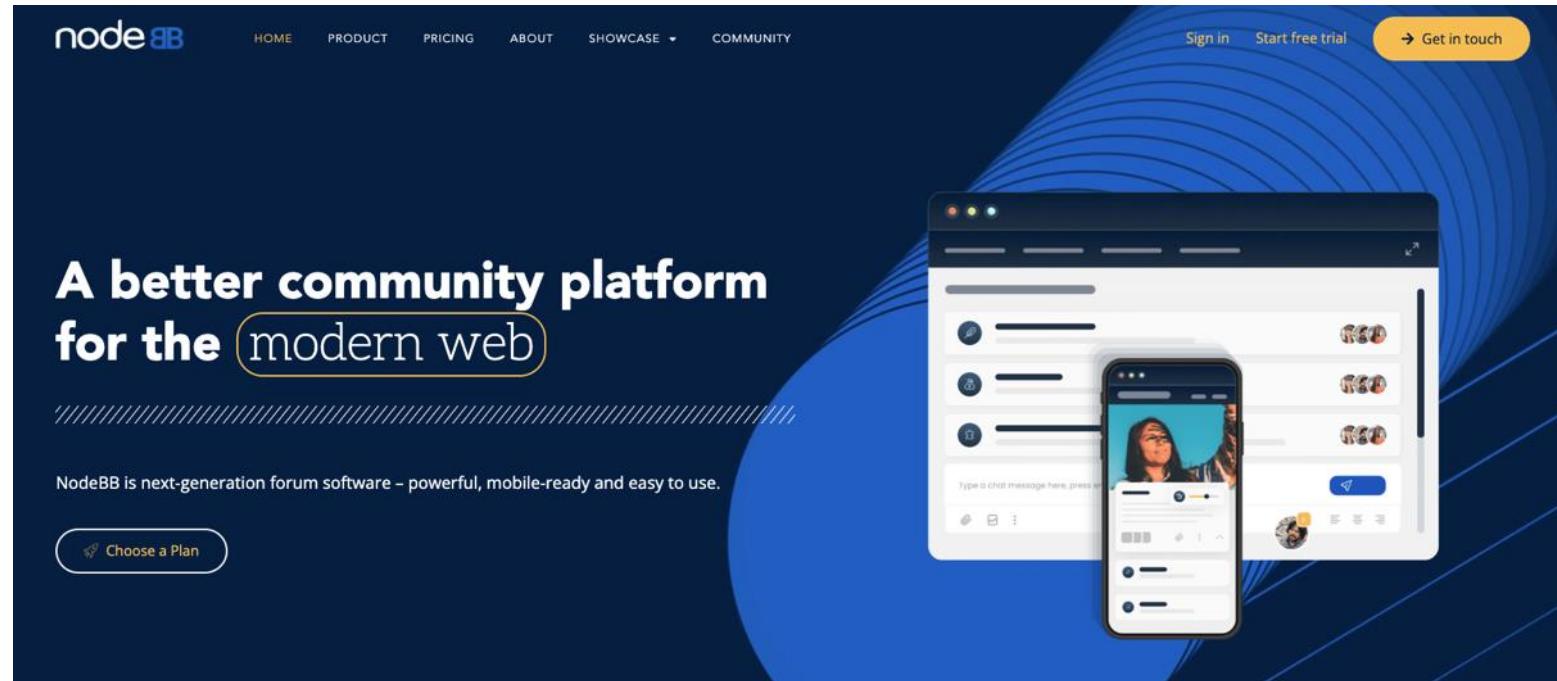


Slack Usage:

- Lots of great help for each other on #technicalsupport, keep up the good work!
 - use  emoji to signal thread is answered
- Please Search before asking new questions
- Please put a picture of your face!!
- We don't guarantee round the clock availability

Context: big ole pile of code

- ... do something with it!



Step 1: Get Code

Step 1: ~~Get Code~~ Git Code

Version Control



Git Fundamentals

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

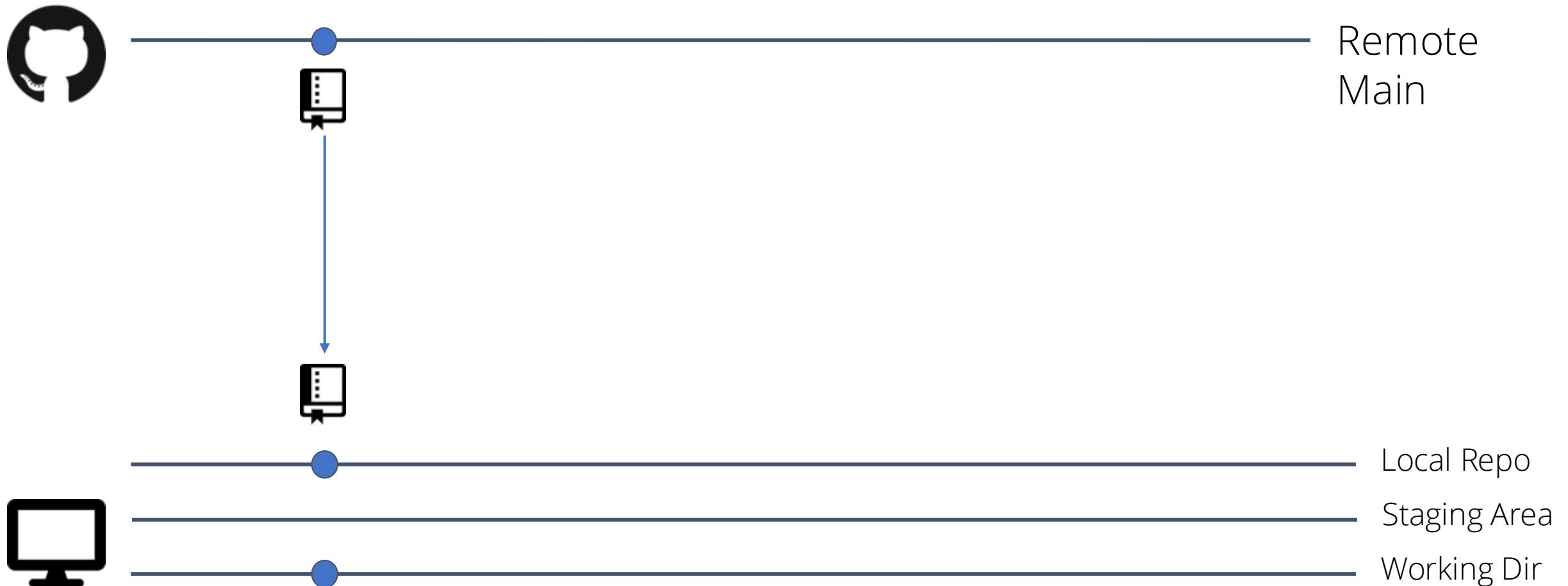
NO IDEA. JUST MEMORIZIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



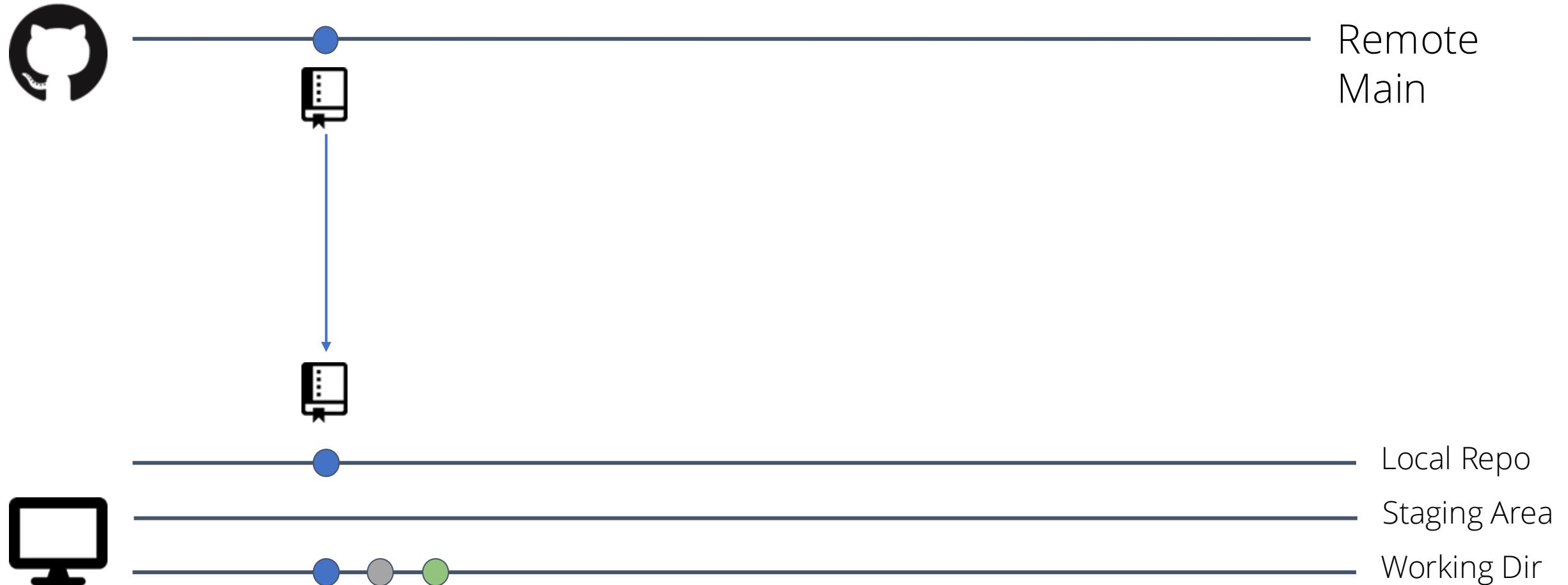
> git clone REPO



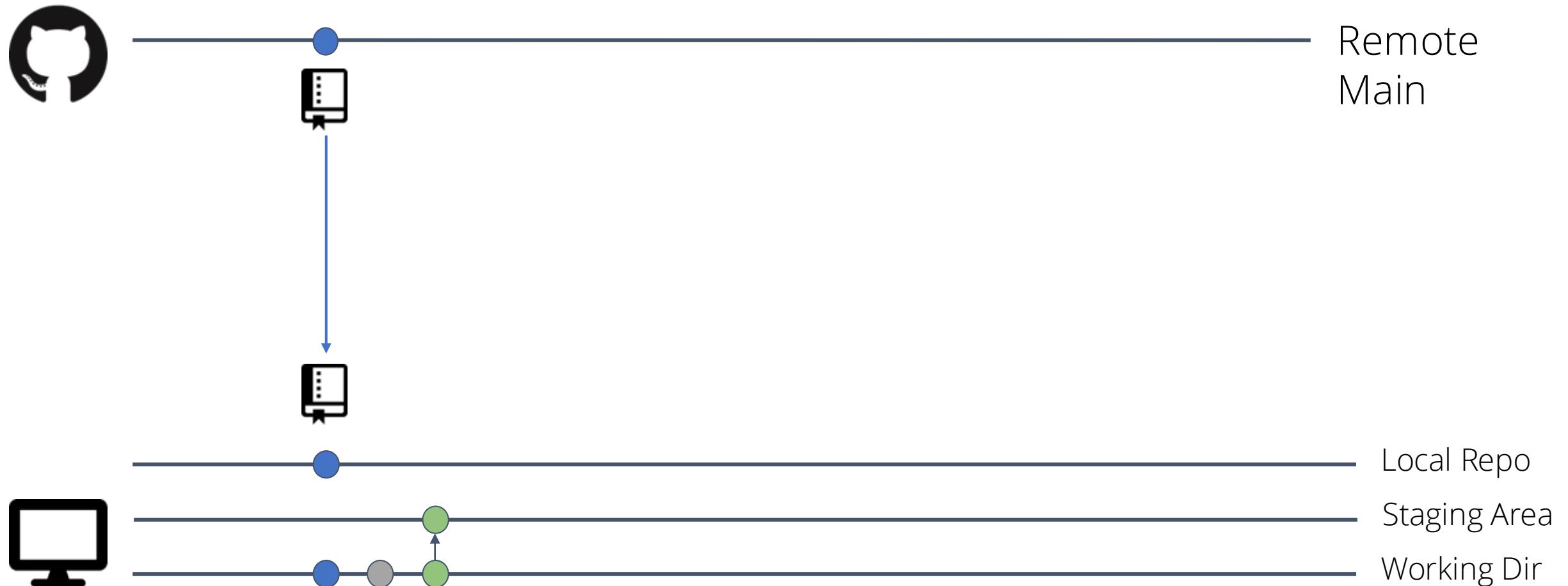
> git clone REPO



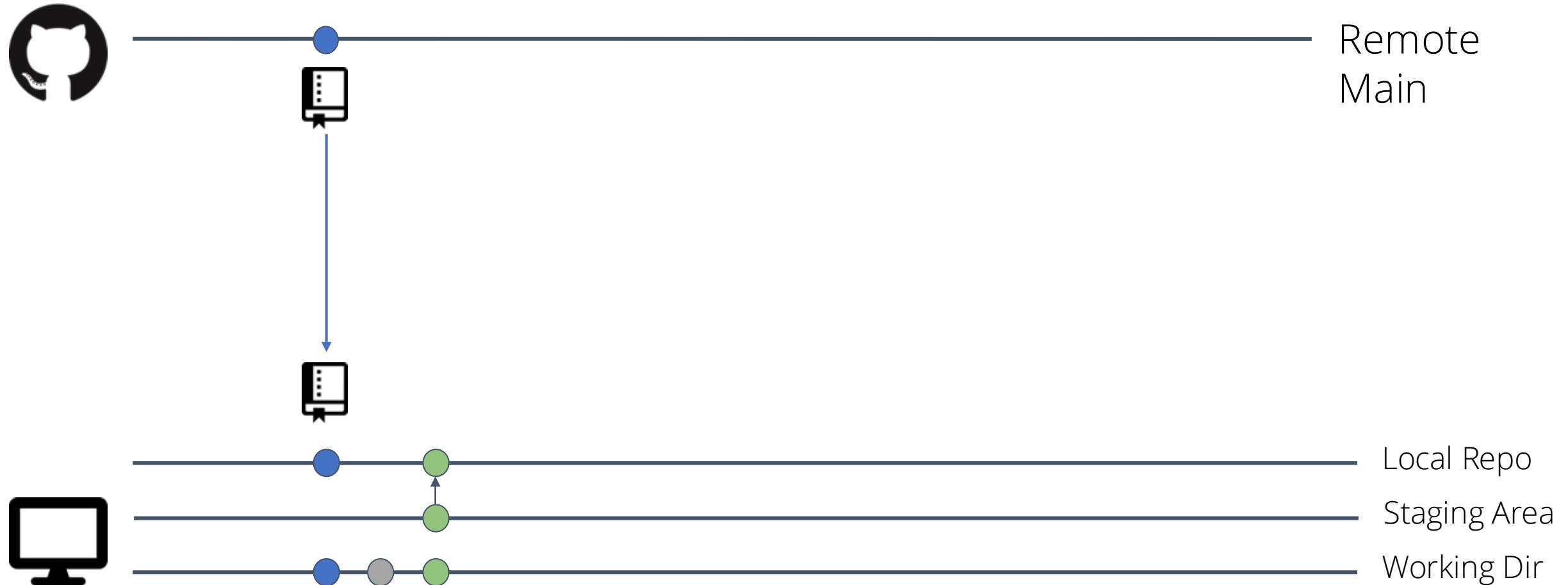
Edit file in working directory



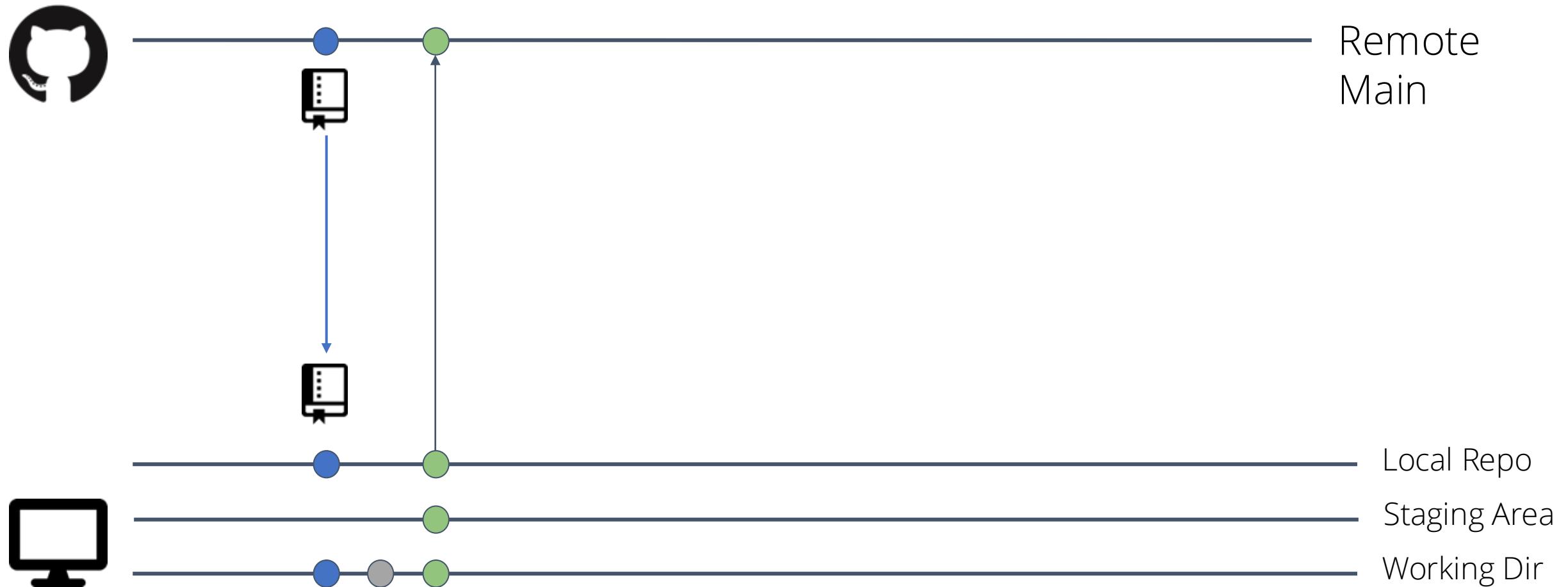
> git add File



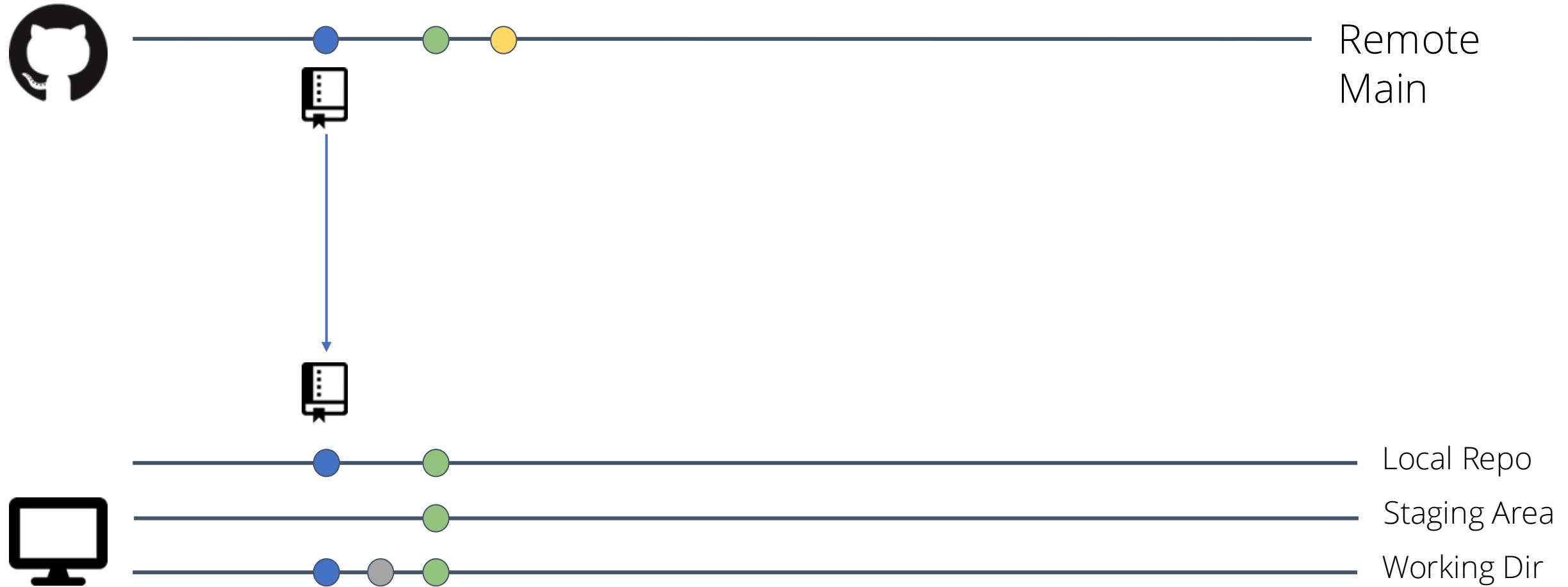
```
> git commit -m "added green edit"
```



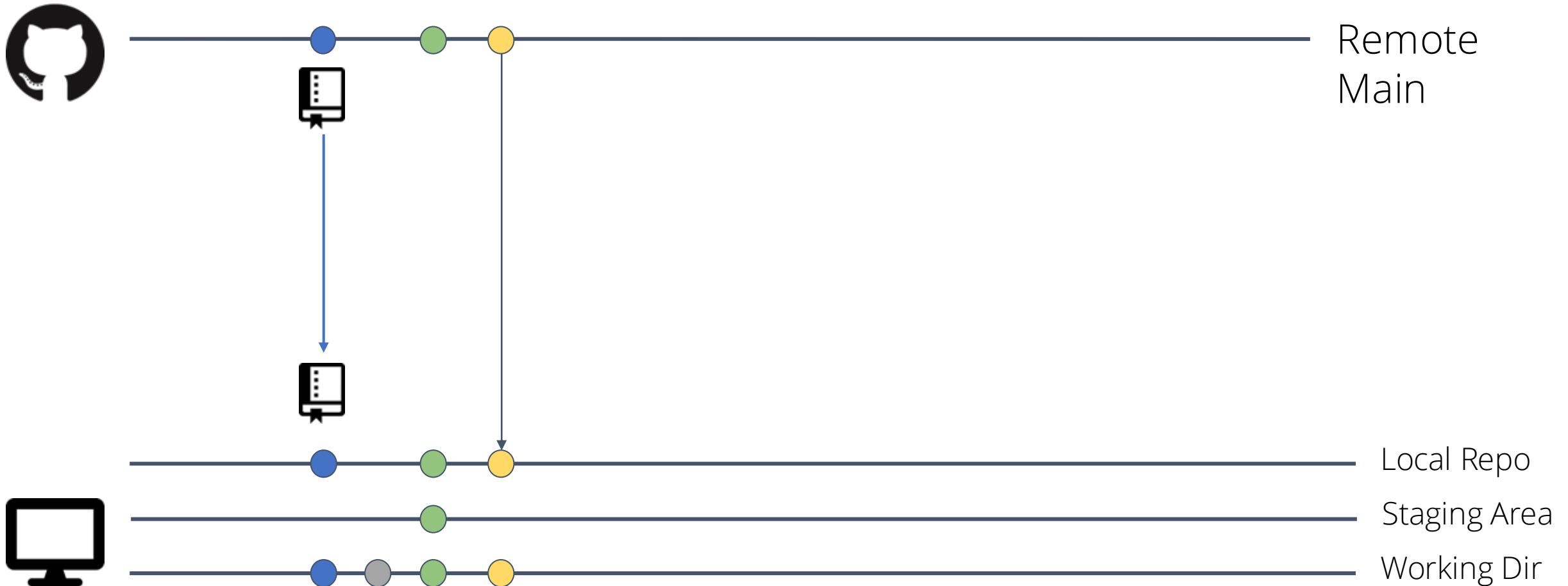
> git push origin main



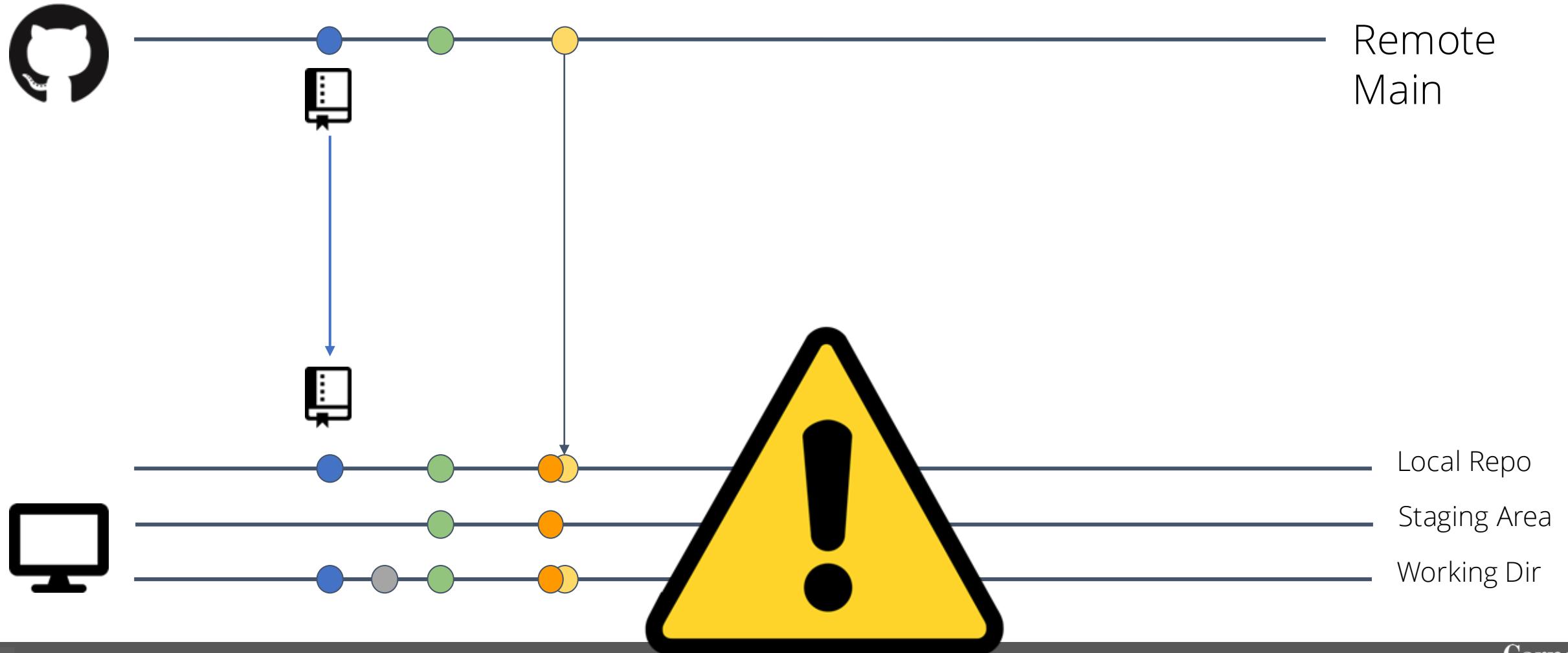
> git fetch (check for changes on remote)



> git pull



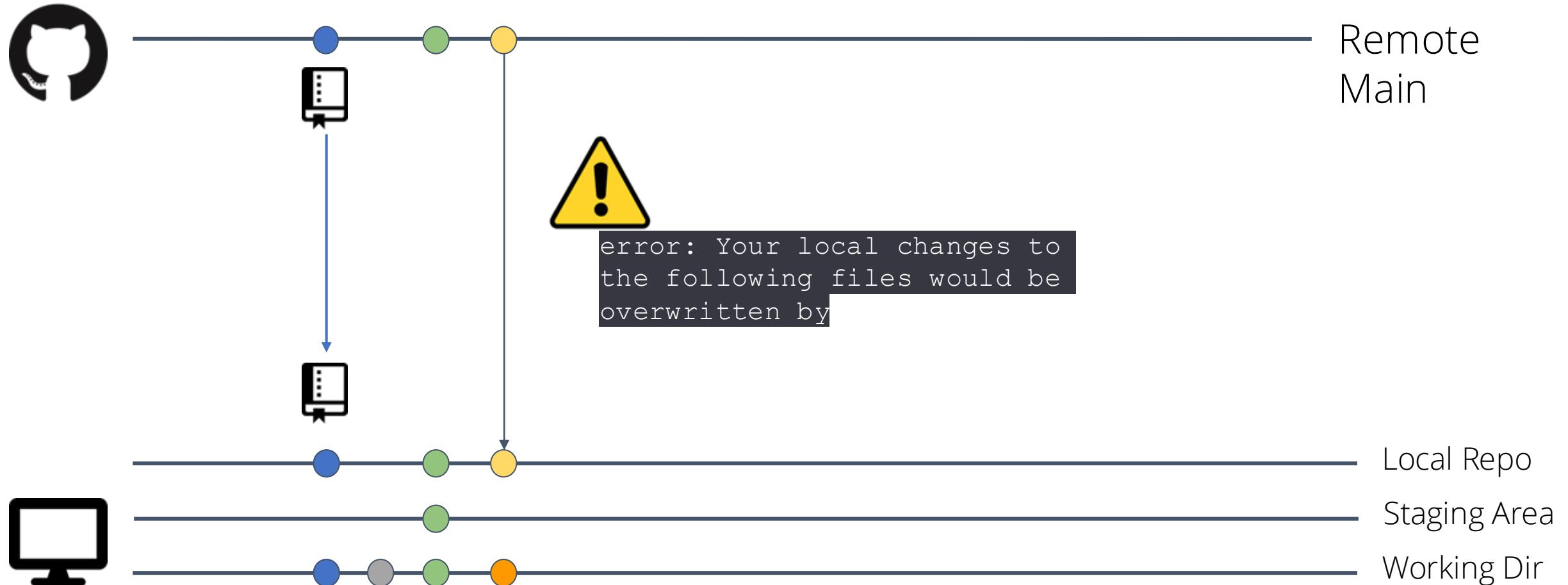
> git pull can lead to Merge Conflicts



Merge Conflict (in VS Code)

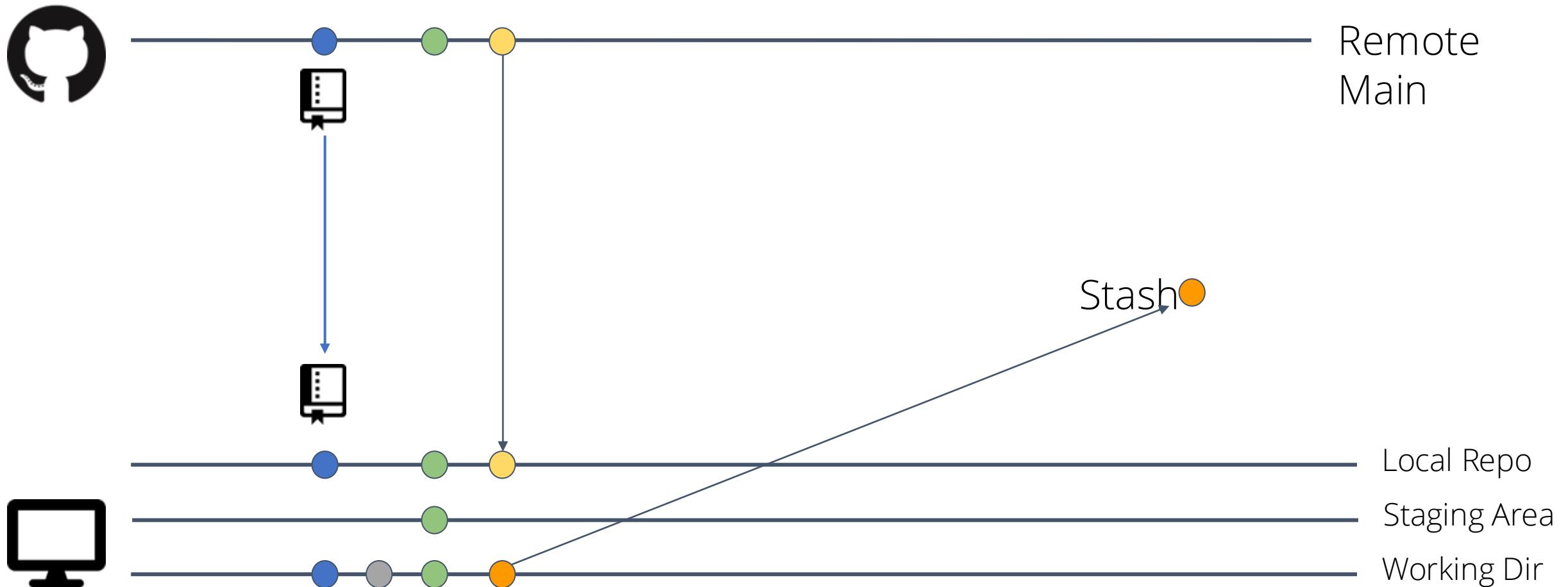
```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
410 <<<<< HEAD (Current Change)
411 →   →   →   →   this.updateSizeClasses();
412 →   →   →   →   this.multiCursorModifier();
413 →   →   →   →   this.contentDisposables.push(this.configurationService.onDidU
414 =====
415 →   →   →   →   this.toggleSizeClasses();
416 >>>> Test (Incoming Change)
417 →   →   →   →   if (input.onReady) {
418 →   →   →   →   →   input.onReady(innerContent);
419 →   →   →   → }
```

> git pull

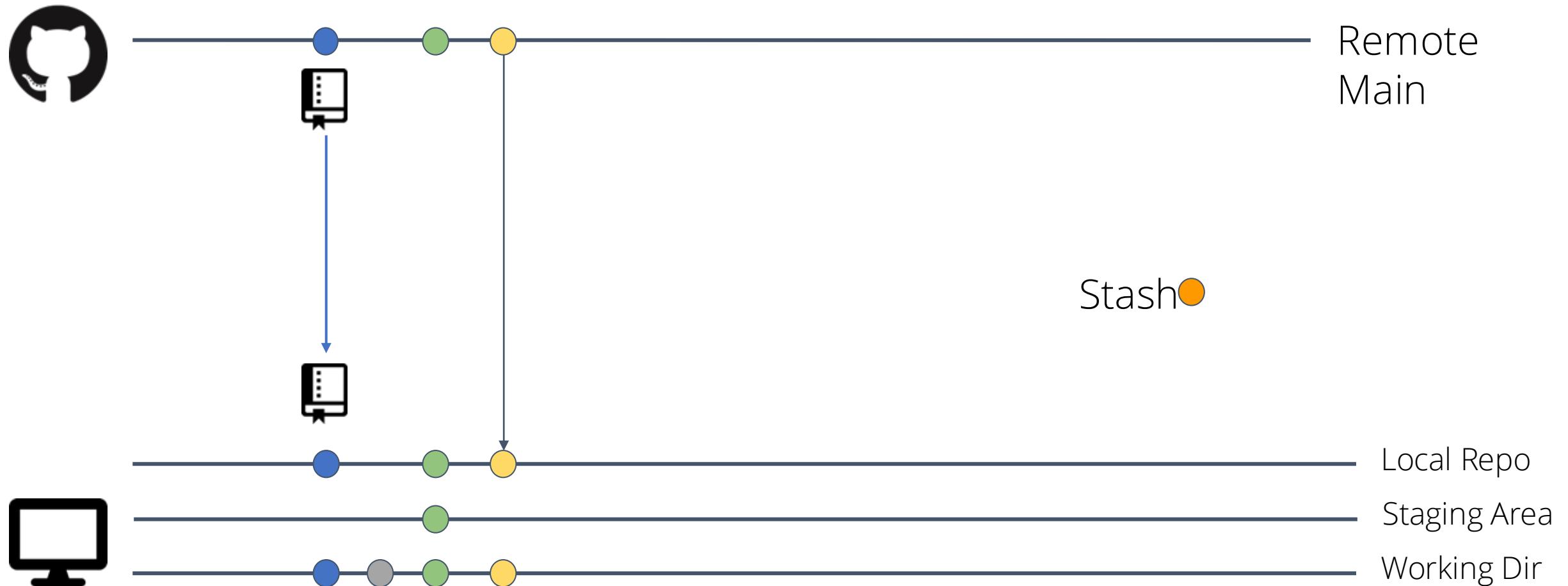


Keep Changes

> git stash

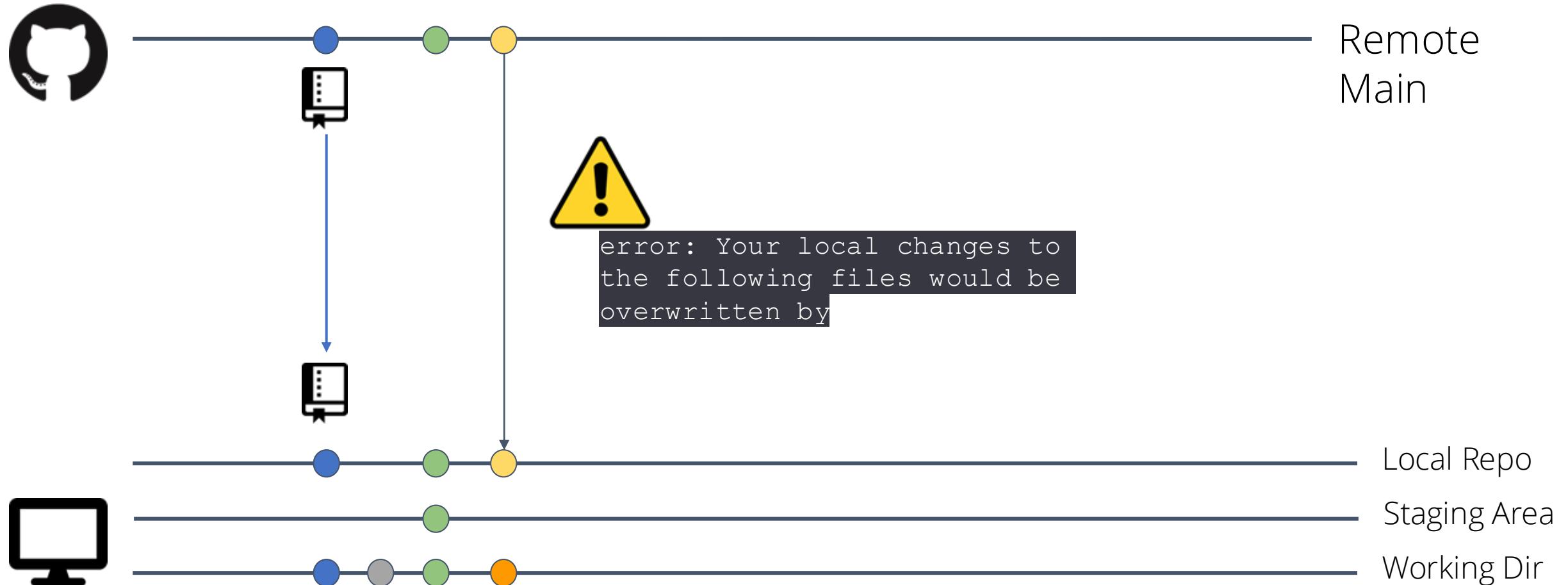


> git pull

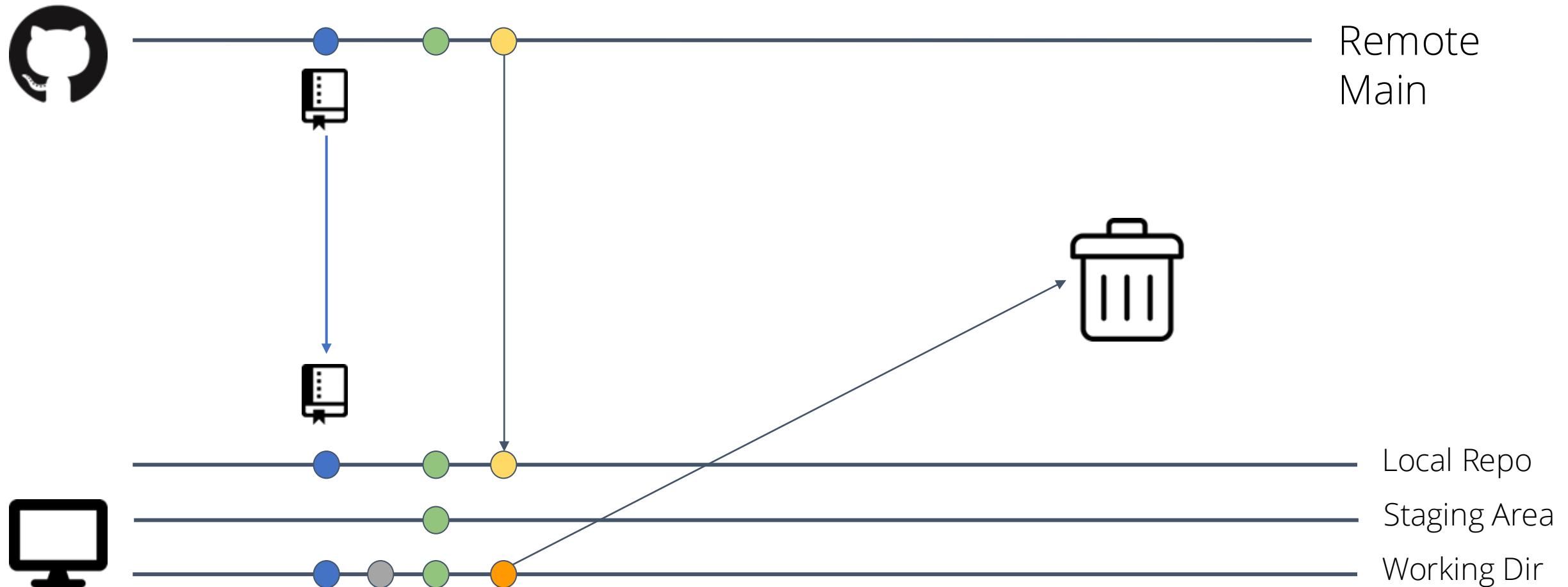


Discard Changes

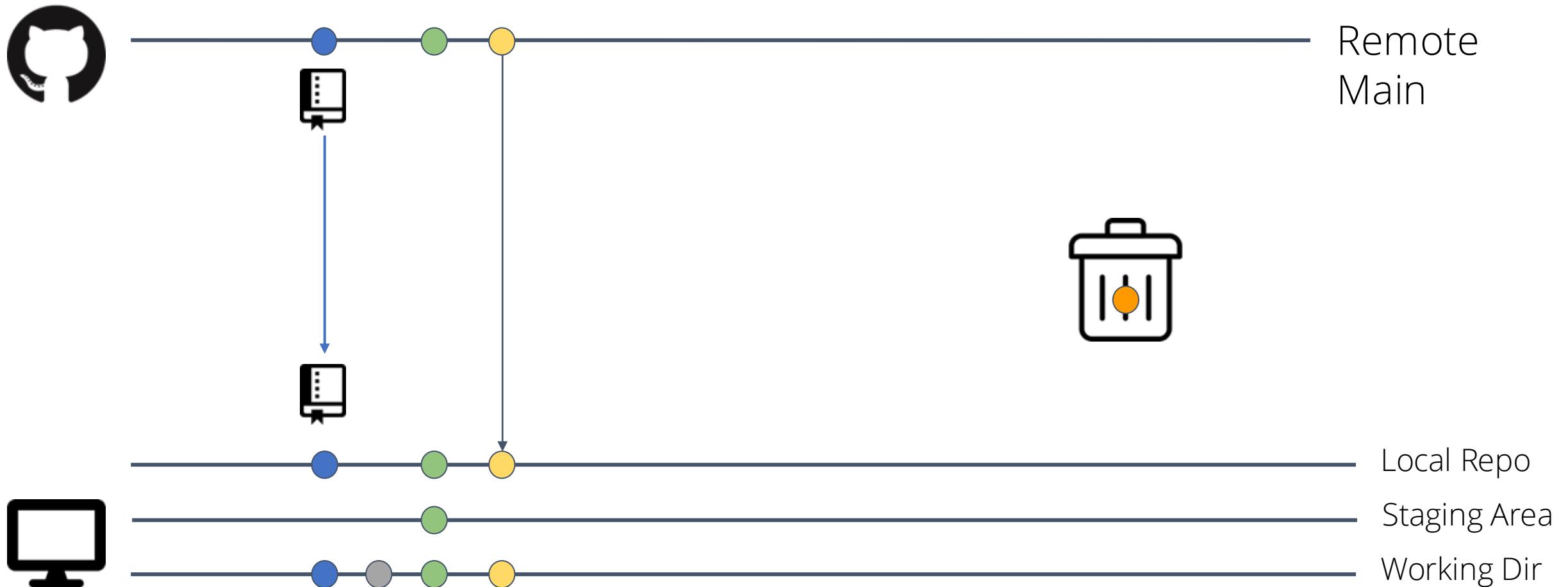
> git pull



> git reset -hard



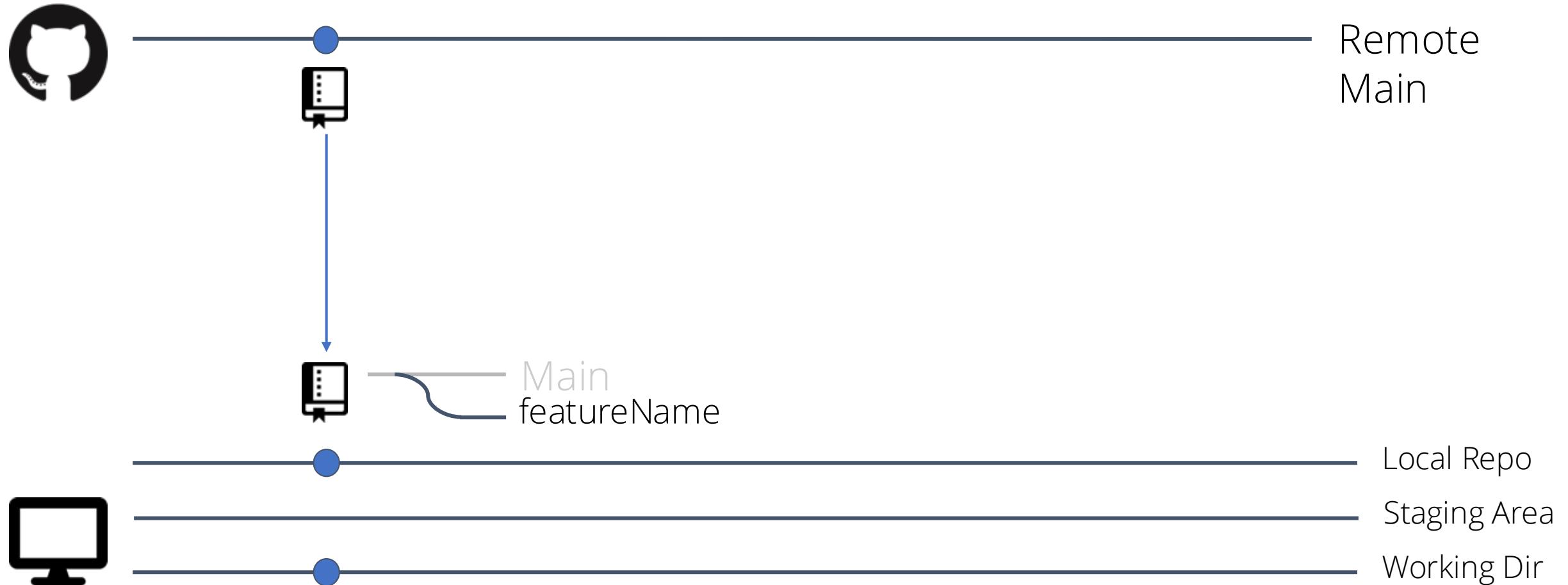
> git pull



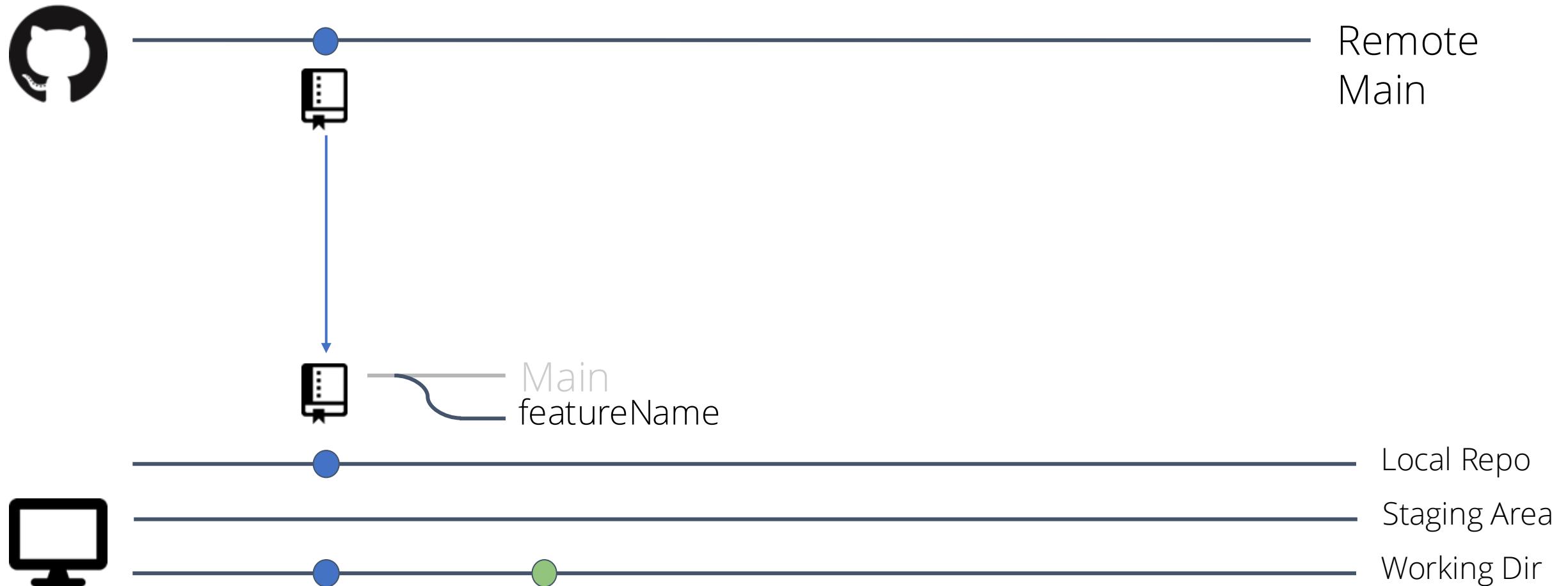
What if you want to work on multiple things at once?

Branches

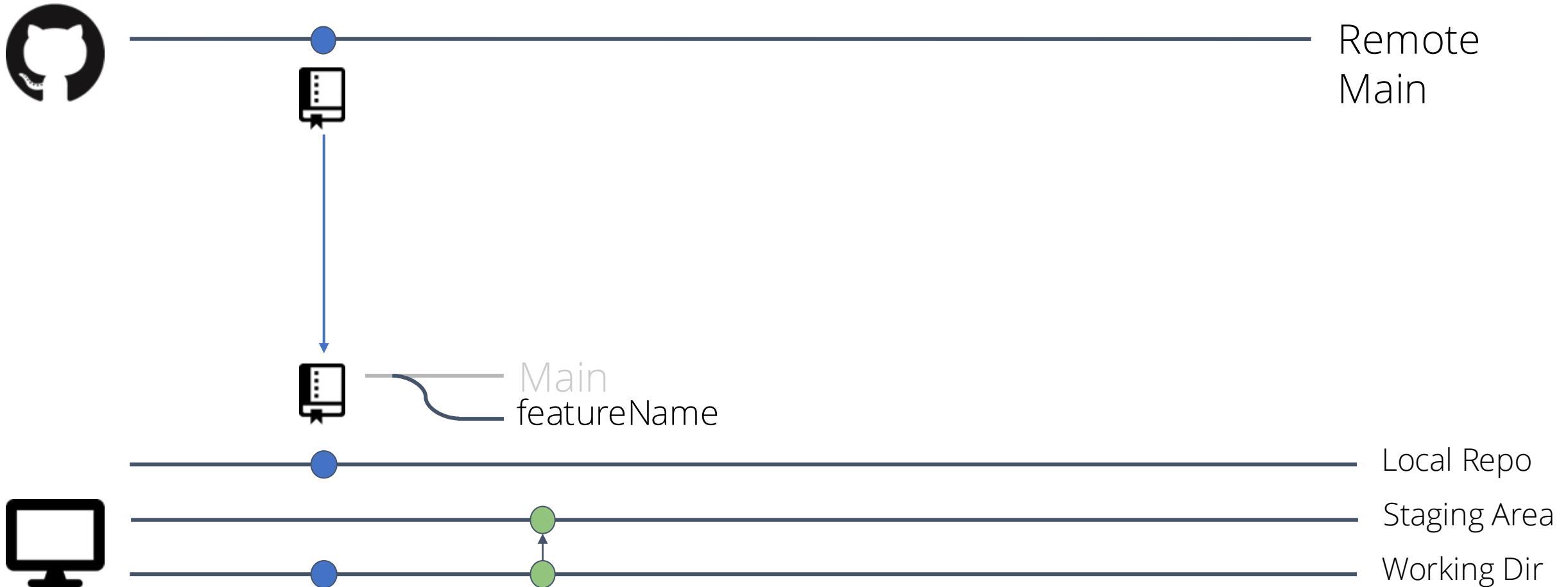
> git checkout -b featureName



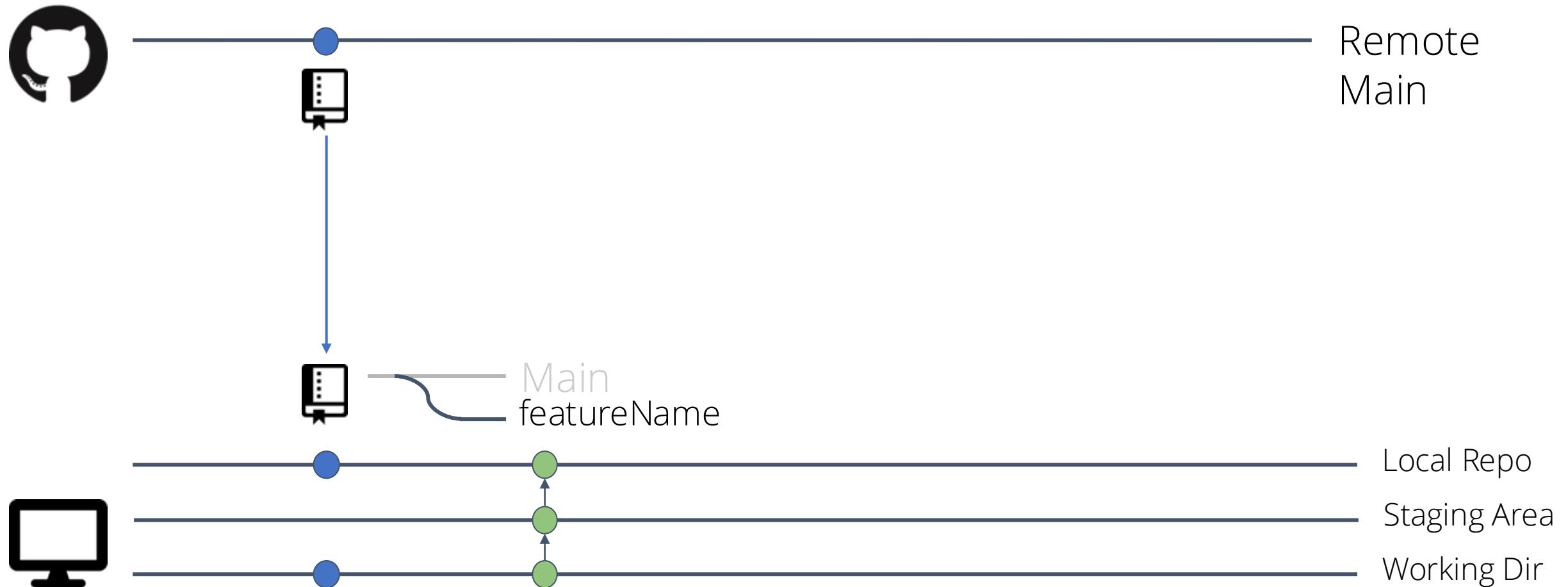
edit file(s) in working directory



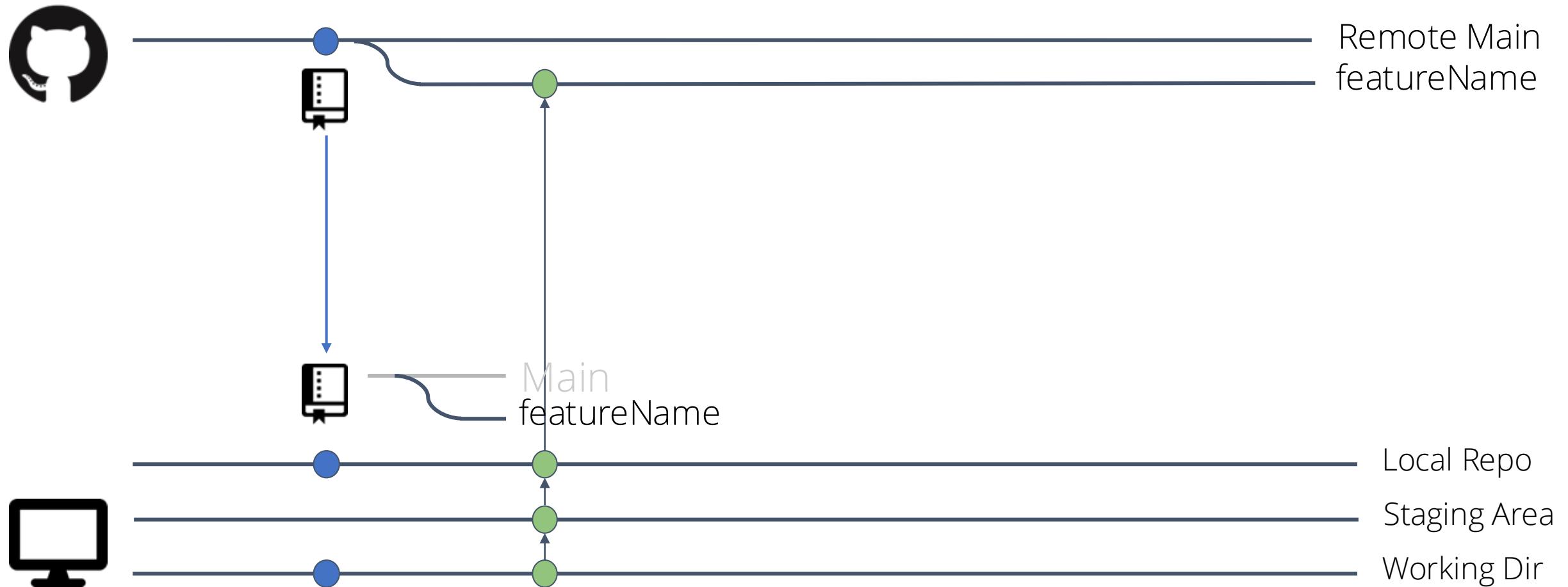
> git add File



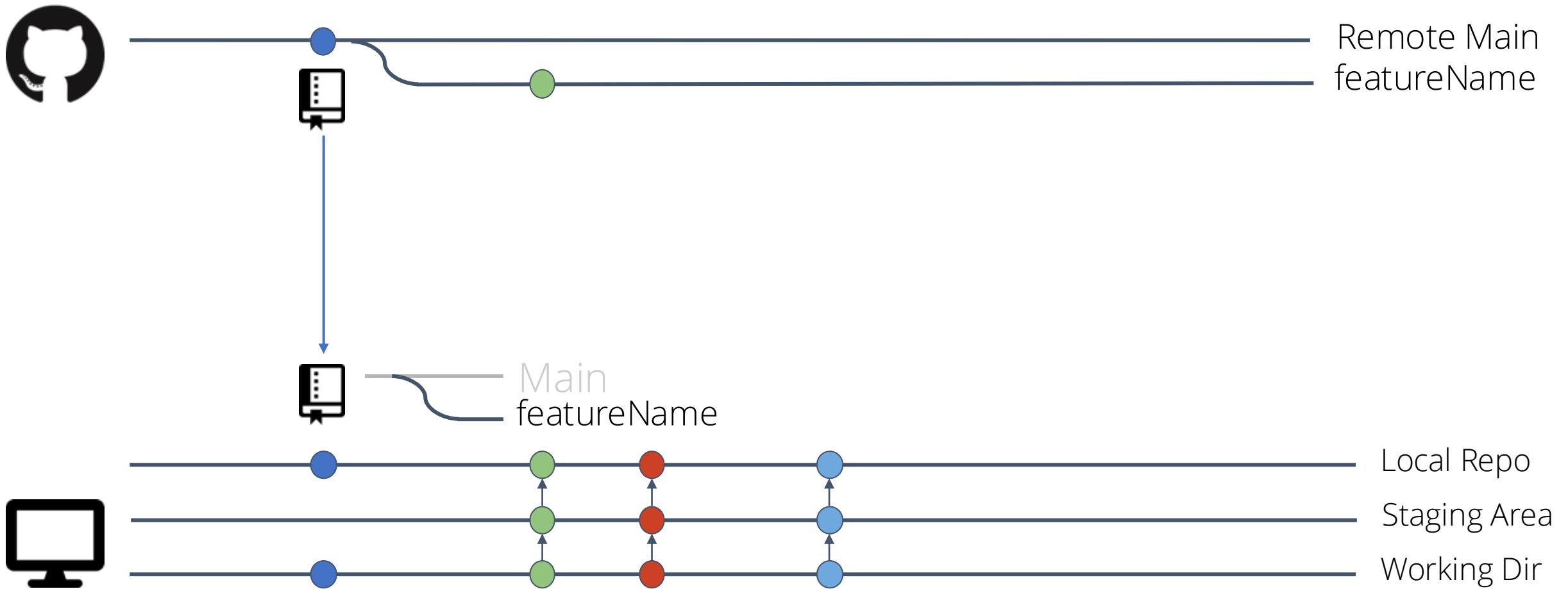
```
> git commit -m "added green edit"
```



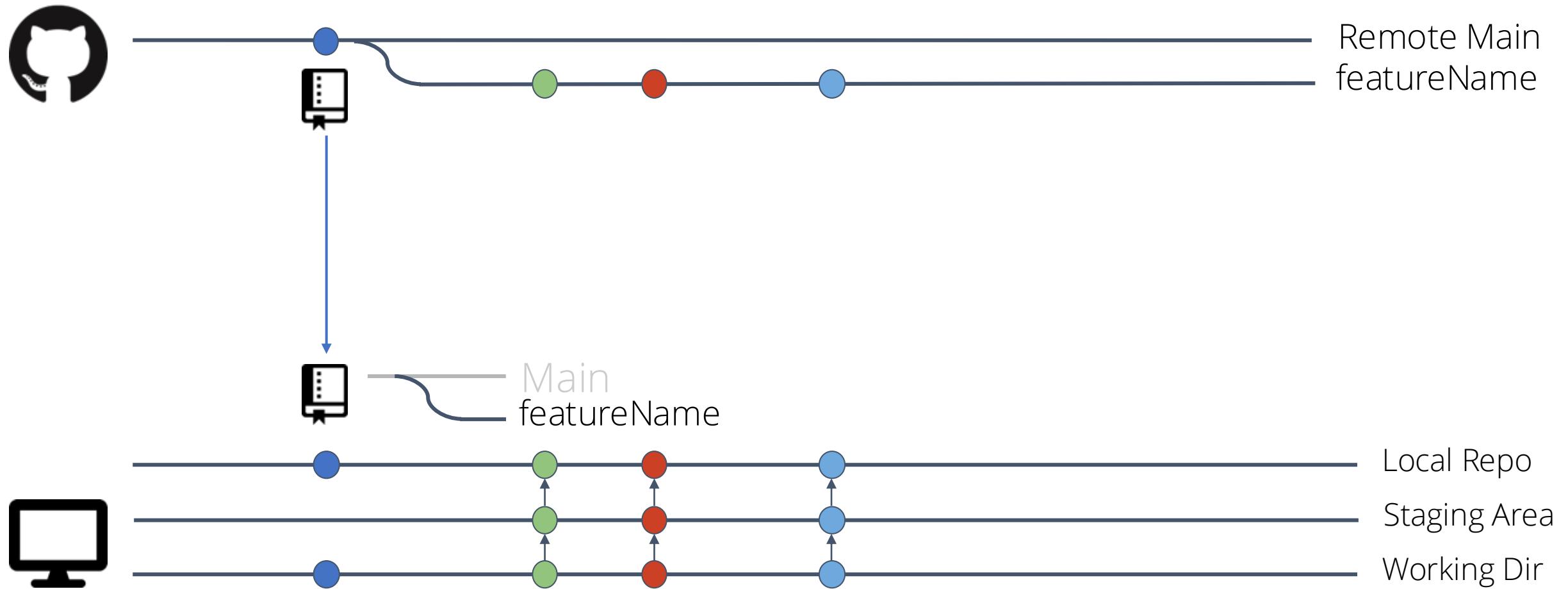
> git push origin featureName



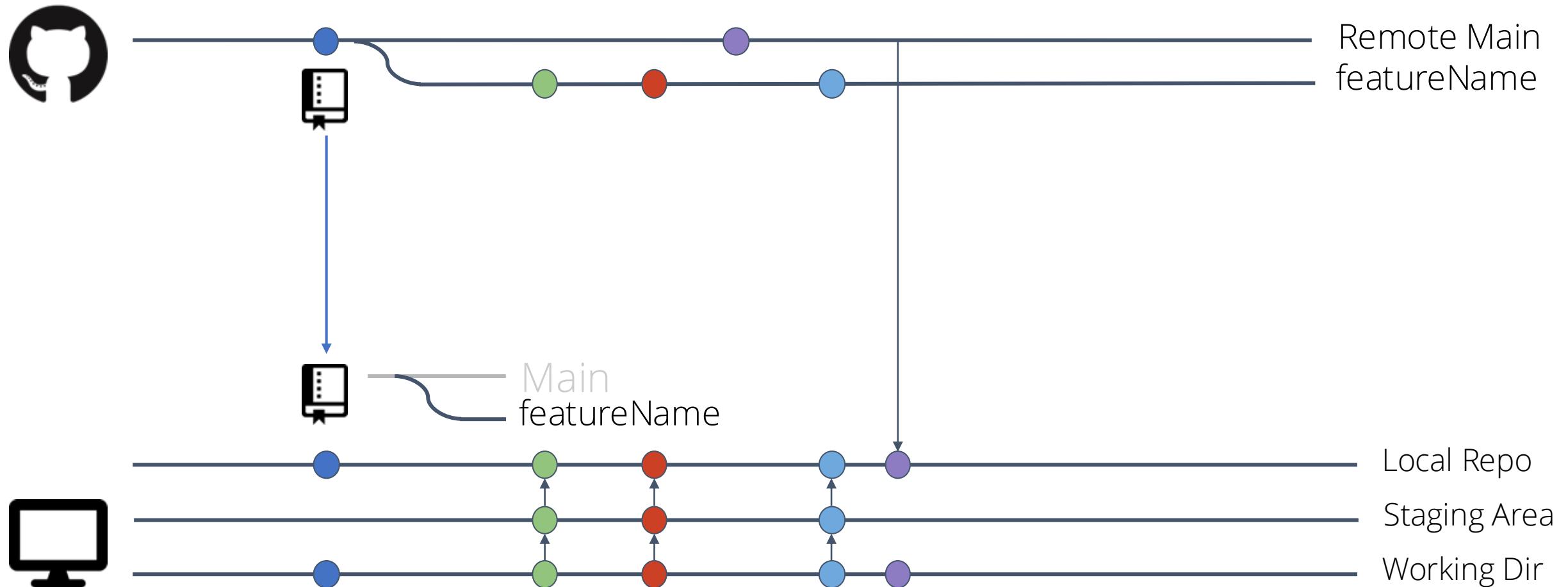
Continue work



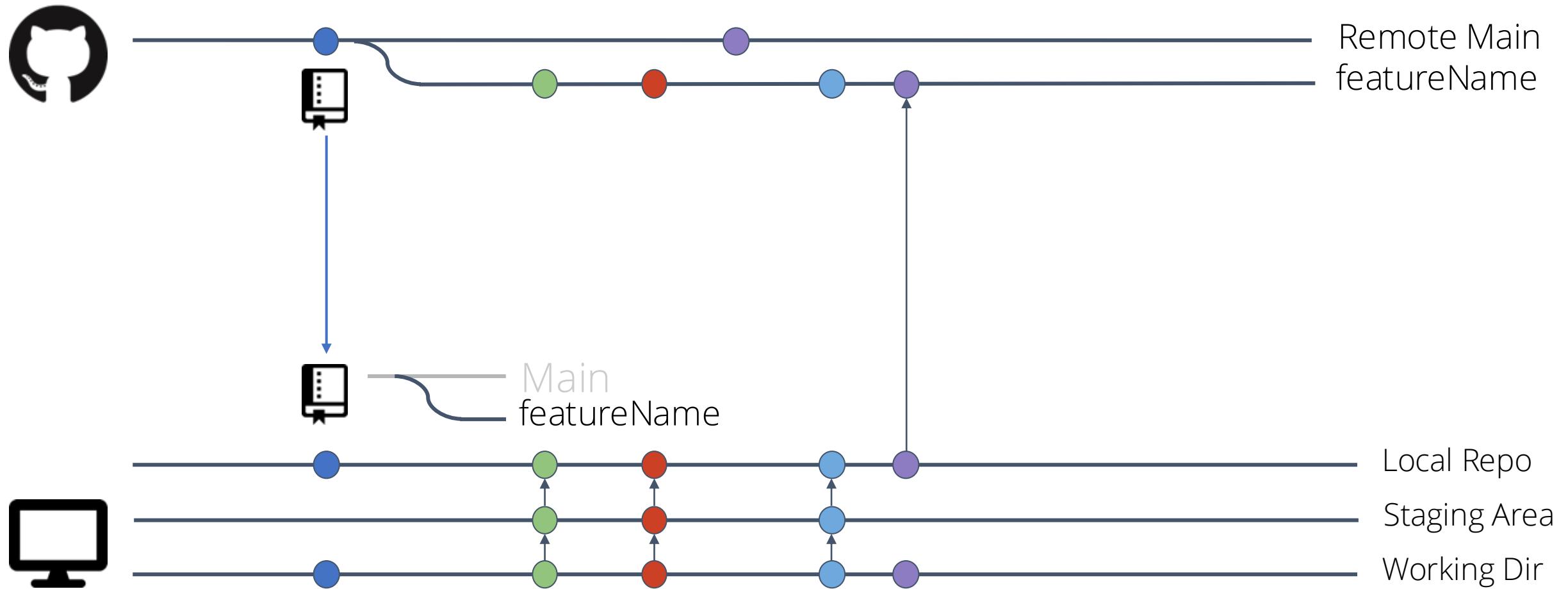
> git push remote featureName



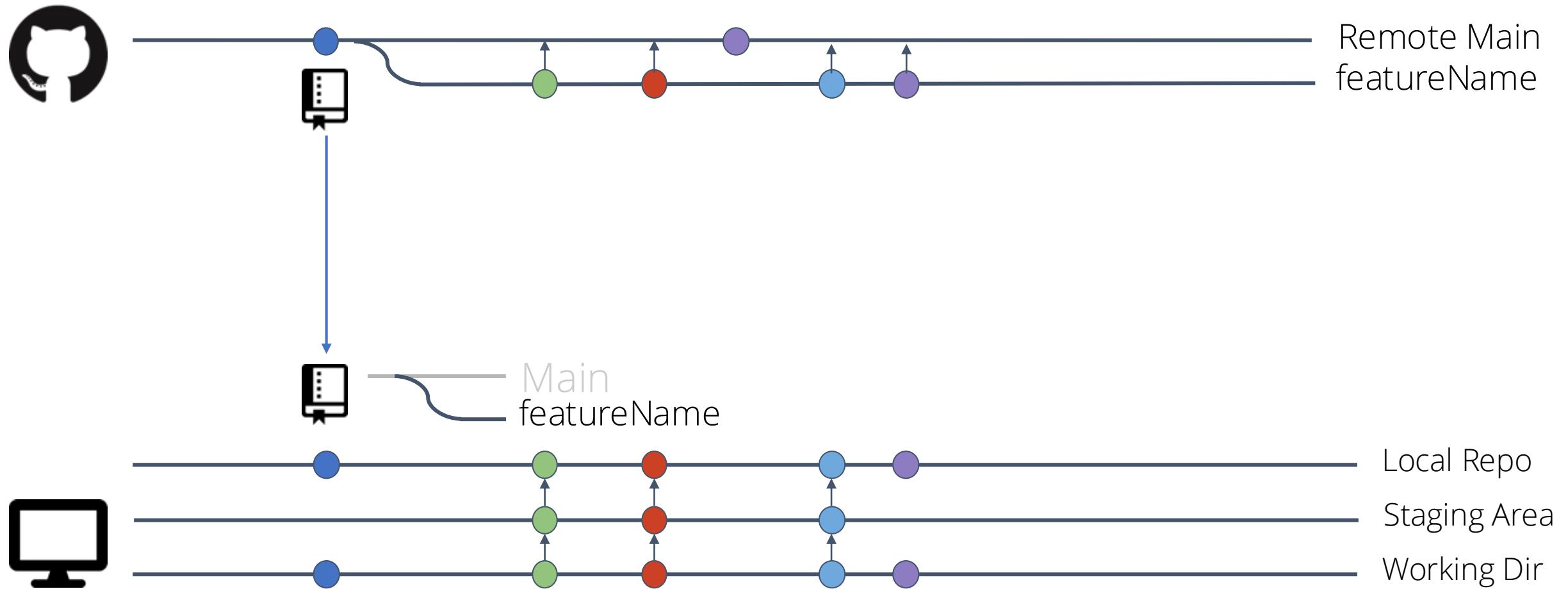
> git pull origin main //Resolve any conflicts



> git push featureName



Open Pull Request on Github



Setup the environment

Problems in running code?

- What version of python are you running? Node? npm?
- What if you have different software that relies on different versions?
- How can you run same code on different platforms?
- How can you automate setup?

Docker



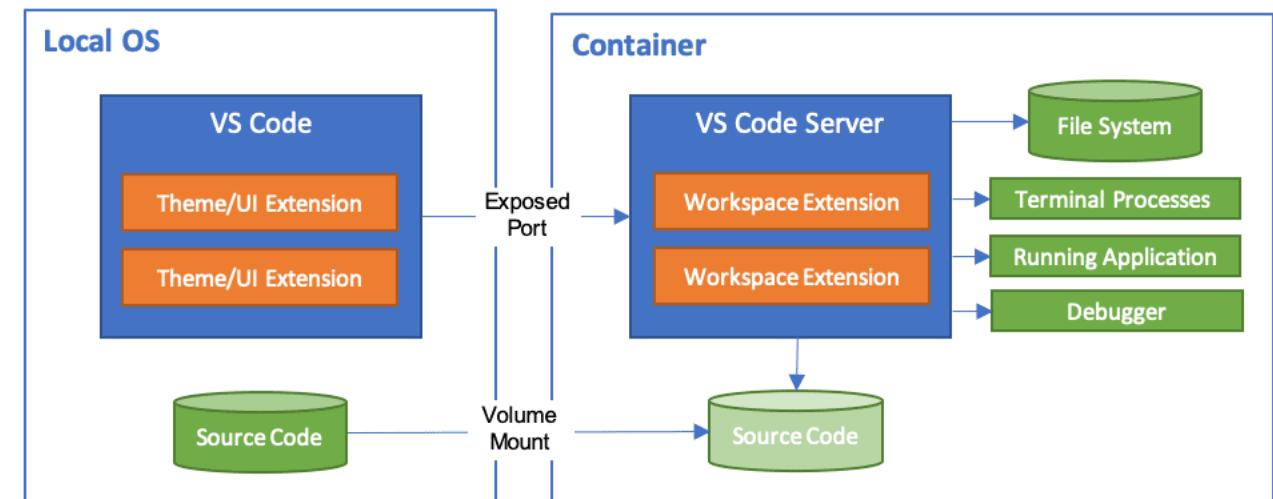
- Docker allows packaging applications and their dependencies into lightweight, portable containers.
- Each container includes everything needed to run the application (code, runtime, libraries, and system tools) in a self-contained environment.
- Containers share the host operating system's kernel but run in isolation from each other, making them much lighter than traditional virtual machines.
- Docker uses images as blueprints for containers, which can be version-controlled and shared through registries like Docker Hub.

Why we use Docker

- Docker solves the "it works on my machine" problem by ensuring applications run consistently across development, testing, and production environments.
- Docker simplifies dependency management by bundling everything an application needs, avoiding conflicts between different projects or system configurations.
- Containers start in seconds and use minimal resources compared to VMs, making development faster and enabling efficient scaling in production.
- HOWEVER: They are complicated to create and manage, and add complexity to running the application.

VSCode Development Containers

- Allow VS Code to handle the development container for us.
- Hides the complexity of containers, while providing all the benefits.
- What we use in this course.



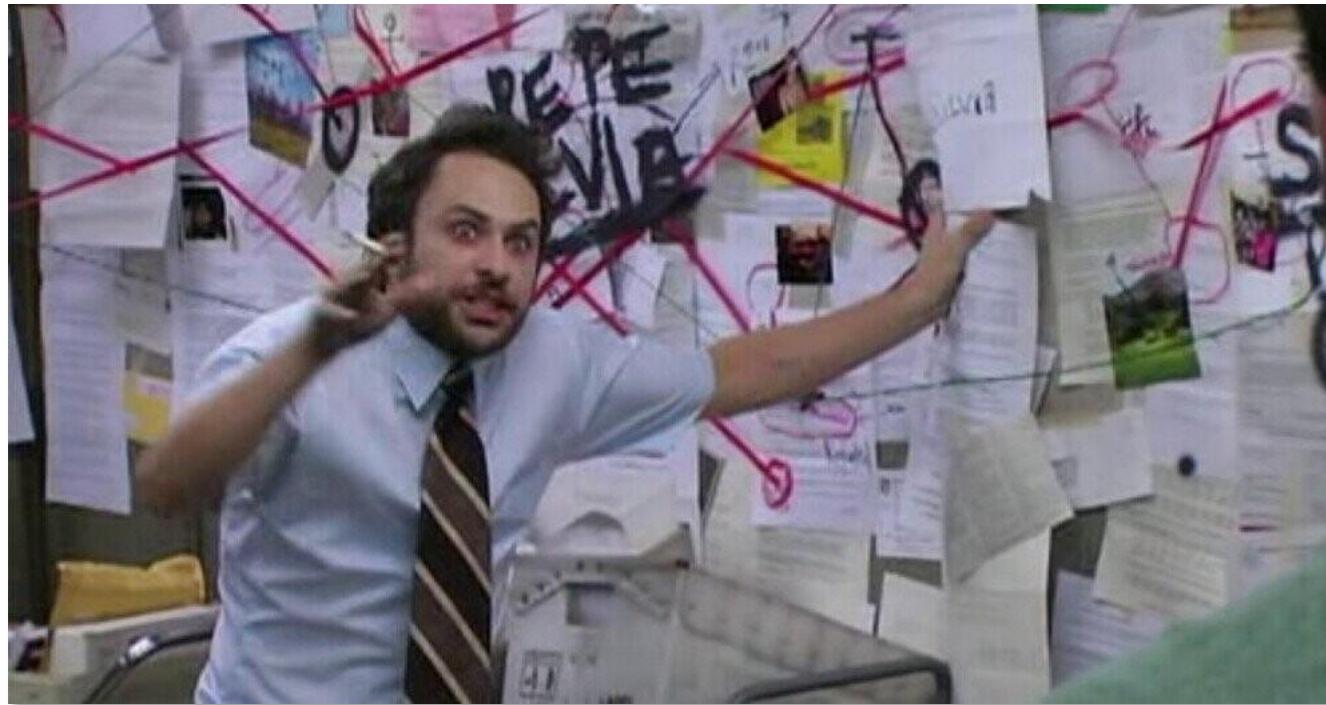
Dive into the code

Participation Activity: Part 1

- Take out a piece of paper (or ask for one)
- Write down the **challenges you've faced trying to understand someone else's code**
- Pair with your neighbor and discuss your answers. Do you agree?
- Share with the class!
- Write your own andrewID on the paper; leave it at the end of class.

**You will never understand
the entire system!**

Challenge: How do I tackle this codebase?



Participation Activity: Part 2

- Write down **strategies to understand a large codebase that is unfamiliar to you**

Challenge: How do I tackle this codebase?

- Leverage your previous experiences (languages, technologies, patterns)
- Consult documentation, whitepapers
- Talk to experts, code owners
- **Follow best practices to build a working model of the system**

Bad news: There are few helpful resources!

- **Working Effectively with Legacy Code**

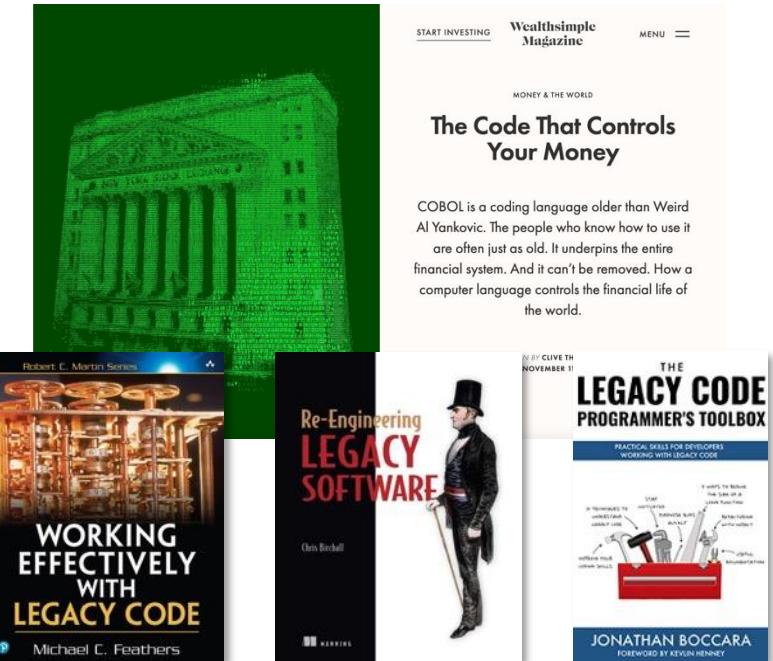
Michael C. Feathers. 2004

- **Re-Engineering Legacy Software**

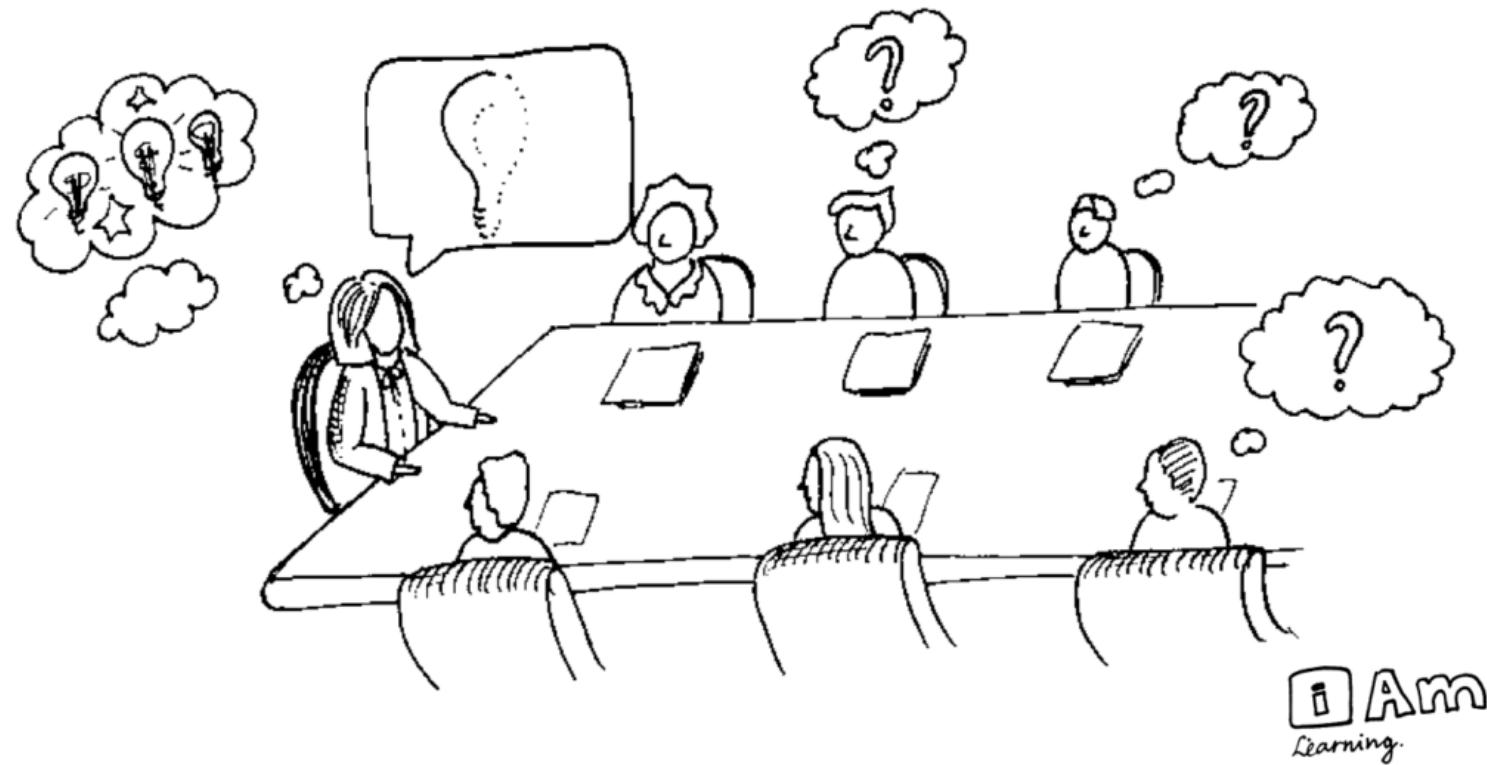
Chris Birchall. 2016

- **The Legacy Code Programmer's Toolbox**

Jonathan Bocca. 2019



Why? Because of Tacit Knowledge



How to Tackle New Codebases

- **Goal:** develop and test a working model about how (part of) a system works
- **Working Model:** an understanding of the pieces of the system (components), and their interactions (connections)
- How to quickly **build, test and refine** models
 - explore various tools, tips, and techniques



essentially,
all models are wrong,
but some are useful

George E. P. Box

Program comprehension strategies

Novice

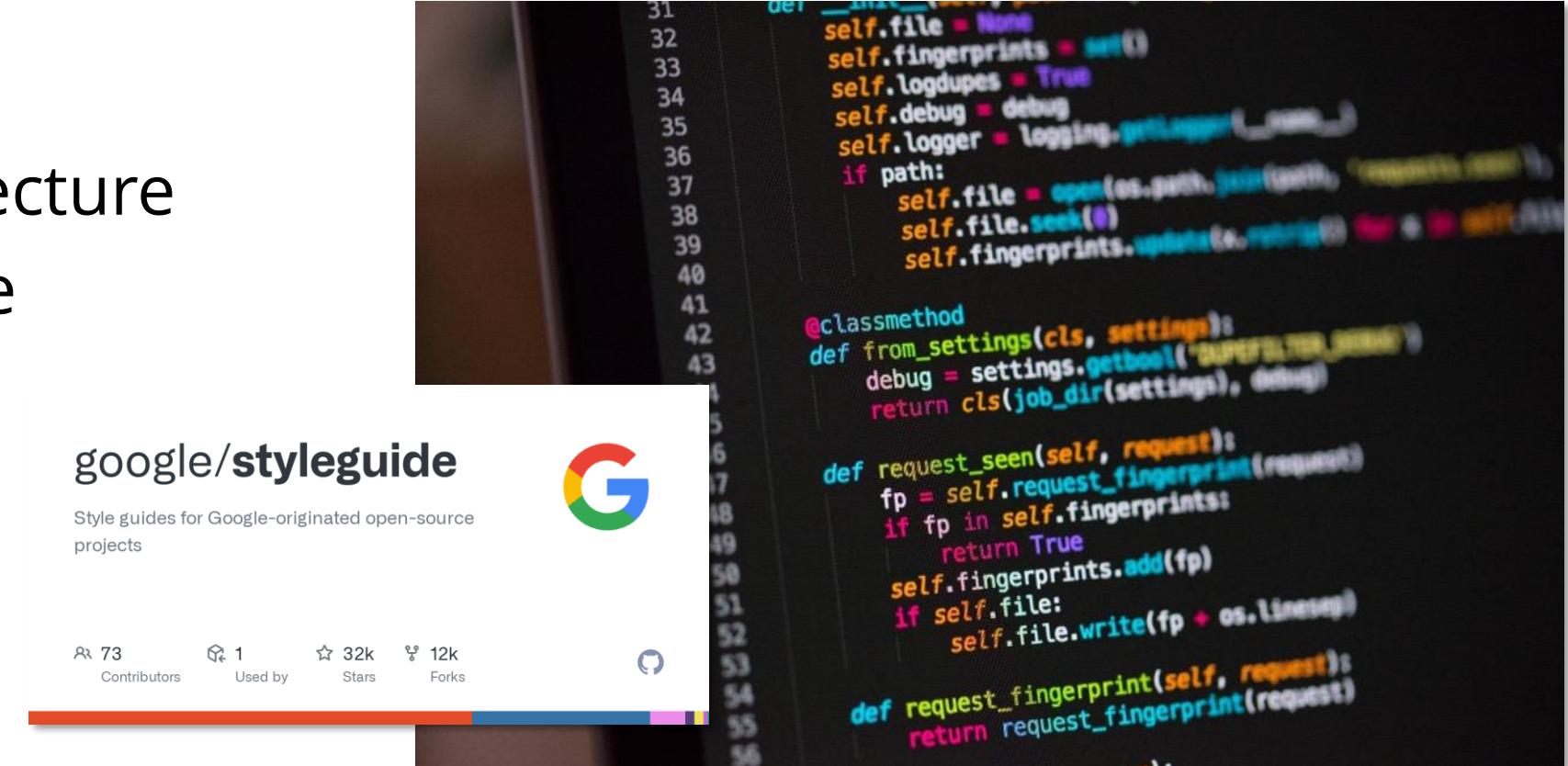
- Reads code line by line
- Revisits same code repeatedly
- Trial and error
- Only tests “happy path”

Expert

- “Top down”
- Recognizes patterns
- Forms hypotheses
- Checks up/downstream consequences

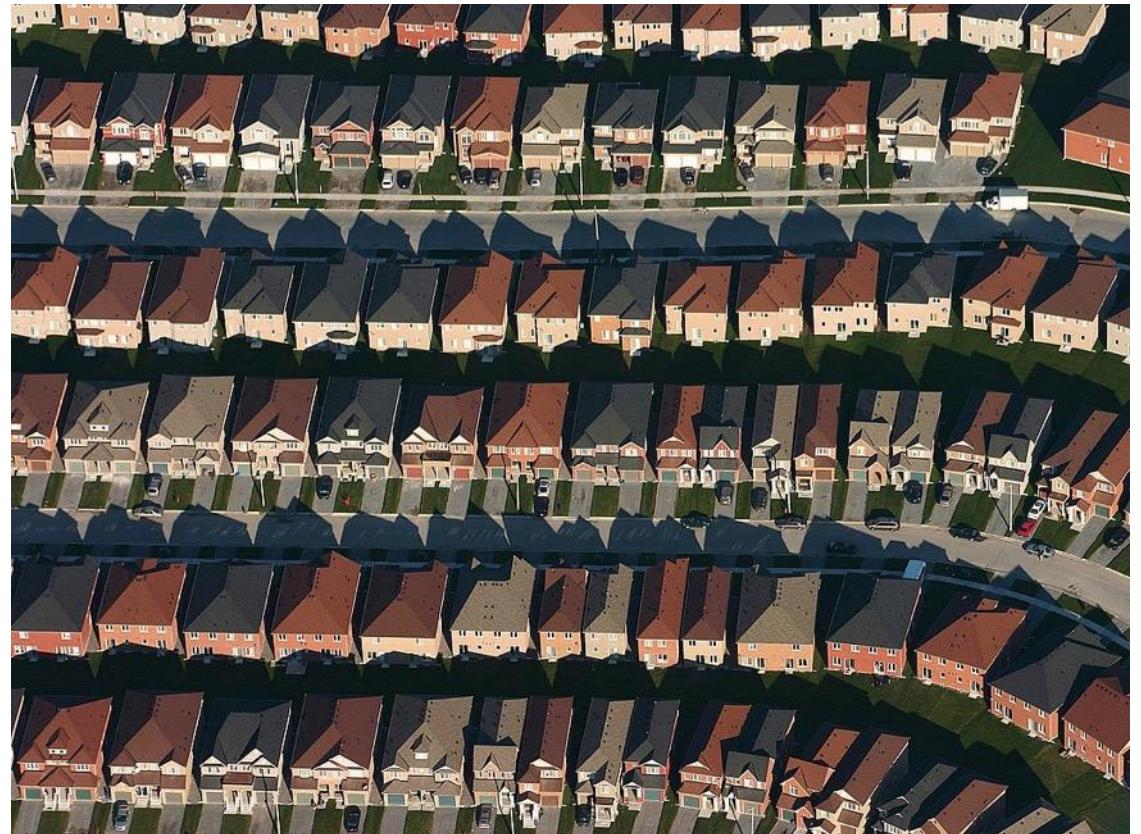
Observation: Software is full of patterns

- File structure
- System architecture
- Code structure
- Names
- ...

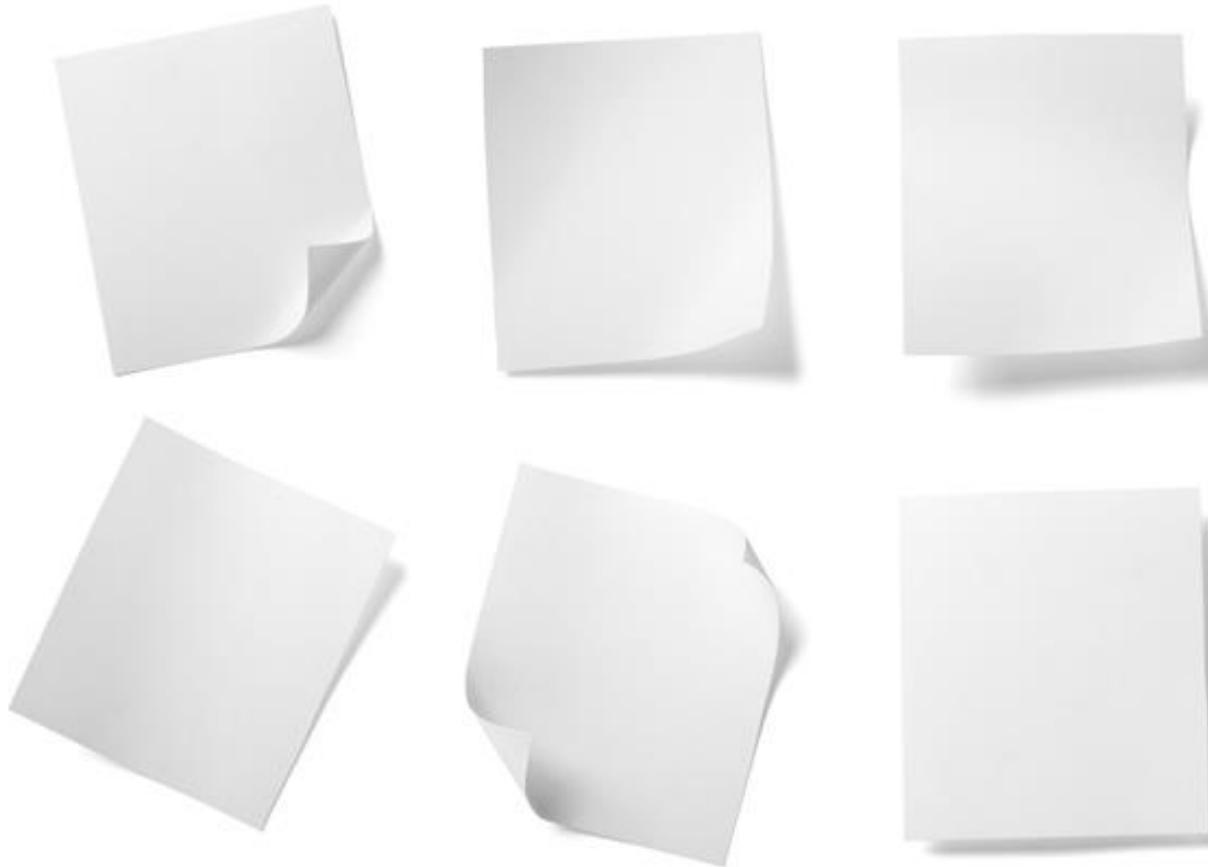


Observation: Software is massively redundant

- There's always something to copy/use as a starting point!



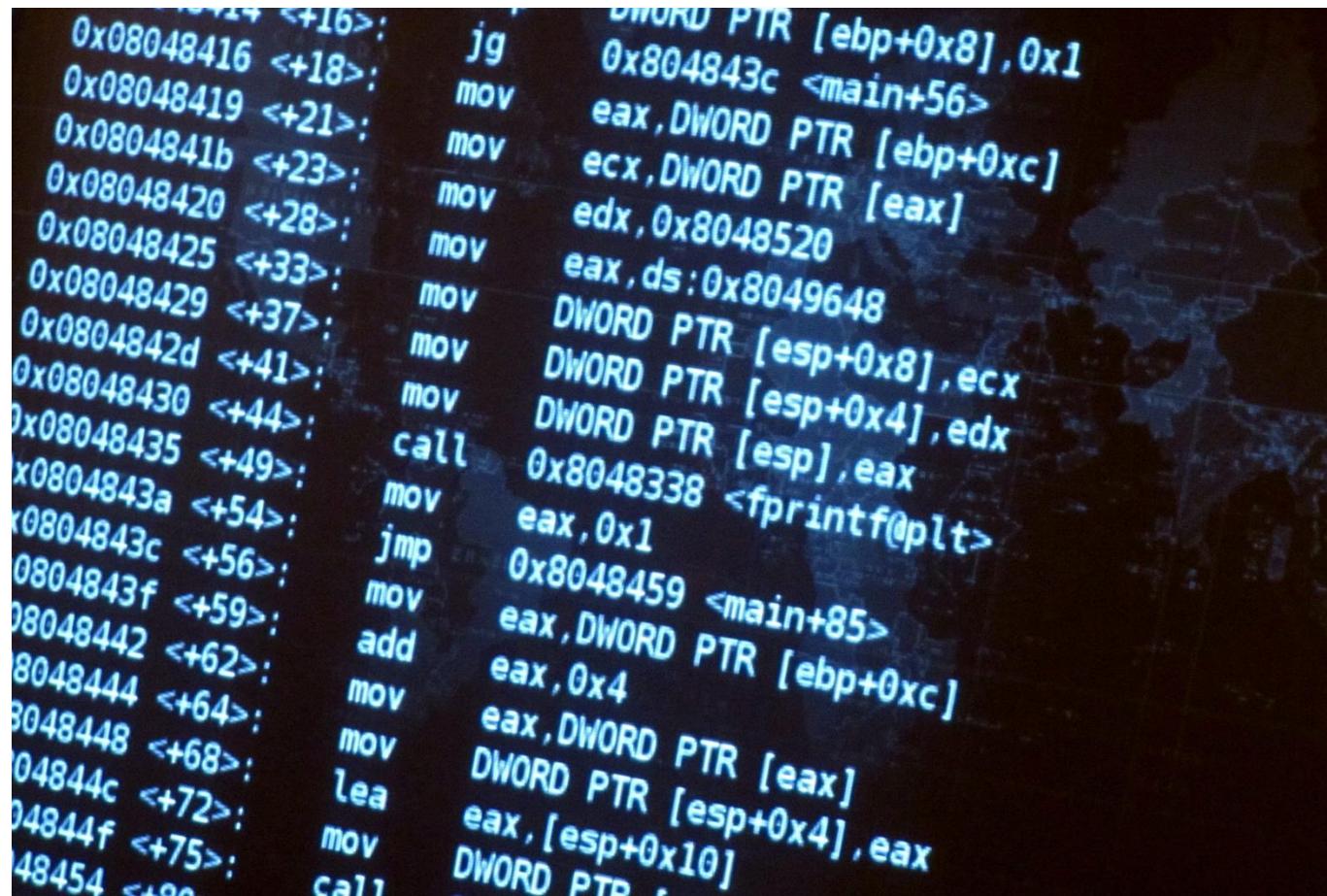
Observation: Code must run to do stuff...



Observation: If code runs, it must have a beginning...



Observation: If code runs, it must exist...



The image shows a screenshot of a debugger interface displaying assembly code. The code is written in Intel x86 assembly language, showing various instructions like jumps, moves, and calls. The assembly code is overlaid on a dark background with a faint map of the world.

```
0x08048414 <+16>:    jg    DWORD PTR [ebp+0x8],0x1
0x08048416 <+18>:    mov   0x804843c <main+56>
0x08048419 <+21>:    mov   eax,DWORD PTR [ebp+0xc]
0x0804841b <+23>:    mov   ecx,DWORD PTR [eax]
0x08048420 <+28>:    mov   edx,0x8048520
0x08048425 <+33>:    mov   eax,ds:0x8049648
0x08048429 <+37>:    mov   DWORD PTR [esp+0x8],ecx
0x0804842d <+41>:    mov   DWORD PTR [esp+0x4],edx
0x08048430 <+44>:    mov   DWORD PTR [esp],eax
0x08048435 <+49>:    call  0x8048338 <fprintf@plt>
0x0804843a <+54>:    mov   eax,0x1
0x0804843c <+56>:    jmp   0x8048459 <main+85>
0x0804843f <+59>:    mov   eax,DWORD PTR [ebp+0xc]
0x08048442 <+62>:    add   eax,0x4
0x08048444 <+64>:    mov   eax,DWORD PTR [eax]
0x08048448 <+68>:    mov   DWORD PTR [esp+0x4],eax
0x0804844c <+72>:    lea   eax,[esp+0x10]
0x0804844f <+75>:    mov   DWORD PTR [esp],eax
0x08048454 <+80>:    call  _
```

⚠️ WARNING ⚠️ AVOID SLOT MACHINE PROGRAMMING



How to build, test, and refine mental models



Examine
artifacts without
running code

Probe
running system to
observe behavior

Modify
code, rebuild, and
assess impact

How to build, test, and refine mental models



Examine
artifacts without
running code

Probe
running system to
observe behavior

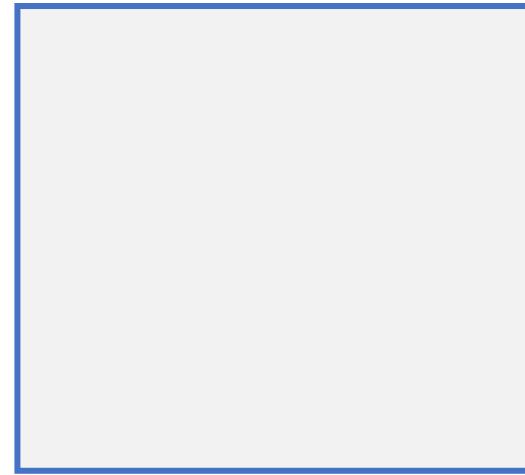
Modify
code, rebuild, and
assess impact

Can code be examined, probed, and modified?

White-box



Grey-box



Black-box



Source code built locally

- ✓ examine
- ✓ probe
- ✓ modify

Binaries running locally

- | Open Source | Closed Source |
|-------------|---------------|
| ✓ examine | ✗ examine |
| ✓ probe | ✓ probe |
| ✗ modify * | ✗ modify * |

Server-side apps running remotely

- | Open Source | Closed Source |
|-------------|---------------|
| ✓ examine | ? |
| ✗ probe | Talk to NSA |
| ✗ modify | |

Creating a model of unfamiliar code

Source code built
locally

Live Demonstration: NodeBB

The screenshot shows the homepage of the NodeBB website. At the top, there is a navigation bar with links for HOME, PRODUCT, PRICING, ABOUT, SHOWCASE, COMMUNITY, Sign in, Start free trial, and Get in touch. The main headline reads "A better community platform for the modern web". Below the headline, a subtext states "NodeBB is next-generation forum software – powerful, mobile-ready and easy to use." A prominent button labeled "Choose a Plan" is visible. On the right side of the page, there is a large graphic featuring a laptop screen displaying a mobile phone interface, set against a background of blue concentric circles.

How to build, test, and refine mental models



Examine
artifacts without
running code

Probe
running system to
observe behavior

Modify
code, rebuild, and
assess impact

How to build, test, and refine mental models



Examine
artifacts without
running code



Probe
running system to
observe behavior



Modify
code, rebuild, and
assess impact



Examine artifacts to build a mental model

Ask

- How do we build / test / run it?
- How is this system structured?
 - Where are the entrypoints?
 - Where are the seams?
Can we probe them?
 - Where is data persisted?
- What technologies does it use?
- What are its stated features?
Limitations?
- Is the project active?

Scan

- Source: code
- Build/CI: package.json, Docker, workflows
- Config: env vars, config.json, ...
- Docs: README, Documentation
- History: commits, issues, PRs, projects

Goal

- a **build/run** command
- an **entry point** that you can target
- a **seam** that you can probe

Tip: Configure and use your IDE to its full potential



- We will provide support for **DevContainers** in VSCode in this course
 - bundles together everything you need into a Docker image that behaves like a native install
 - **Right click** on code to learn more
 - variables, functions, classes, modules, ...
 - Go to Definition, Go to References, Rename Symbol, Refactor, ...
 - Install and explore IDE **Extensions**
 - Redis, ESLint, OpenAPI Editor, LiveShare, ...

Tip: Consider documentation and tutorials judiciously

- Info on how to build the system, its dependencies, and how to use it
- Great for finding **entry points**
- Can tell you about the overall system architecture; more on that topic later in the semester
- **⚠ Often out of date!** Treat as a starting point rather than truth

The screenshot shows the "Getting Started" section of the NodeBB documentation. The left sidebar lists navigation links such as Home, Installation, Configuration, Federation (v4), Development, Getting Started (which is currently selected), Quickstart plugin, Read API Reference, Write API Reference, Plugins, Themes, Writing Widgets for NodeBB, Database structure, Internationalisation (i18n), Style Guide, Admin Panel, and FAQ. The main content area starts with a heading "For optimized development, the NodeBB team (and many of the plugin authors) use Grunt as part of their workflow." It explains the workflow involving Grunt and provides instructions for running tests using either the command `npm test` or `npx mocha test/your_test_file.js`. It also shows examples of configuration blocks for `config.json` and `config.test.json`.

nodeBB Getting Started

For optimized development, the NodeBB team (and many of the plugin authors) use [Grunt](#) as part of their workflow. The pre-requisite dependencies are installed with NodeBB during the `npm install` step, although you may need to install `grunt-cli` (or your distribution's appropriate package for grunt) to allow `grunt` to be called via the command line.

Once installed, you can simply run `grunt` to build all assets and run NodeBB. Grunt will also watch for file changes and selectively build a subset of NodeBB's assets so there is no need to hop back to the console to stop NodeBB, build, and restart NodeBB.

Alternatively, you can run `grunt --skip`, which skips the build step and simply runs NodeBB with file watching enabled.

Testing

In order to run tests on NodeBB, add the following block to your `config.json` file:

```
"test_database": {  
  "host": "127.0.0.1",  
  "port": "27017",  
  "username": "",  
  "password": "",  
  "database": "test"  
}
```

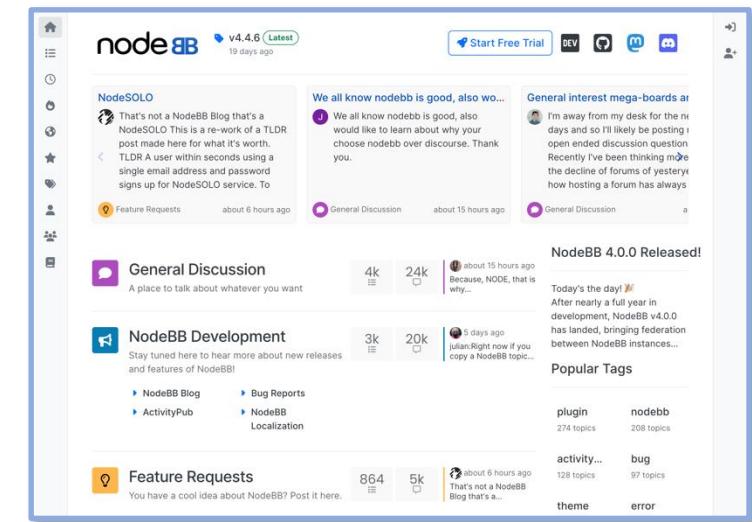
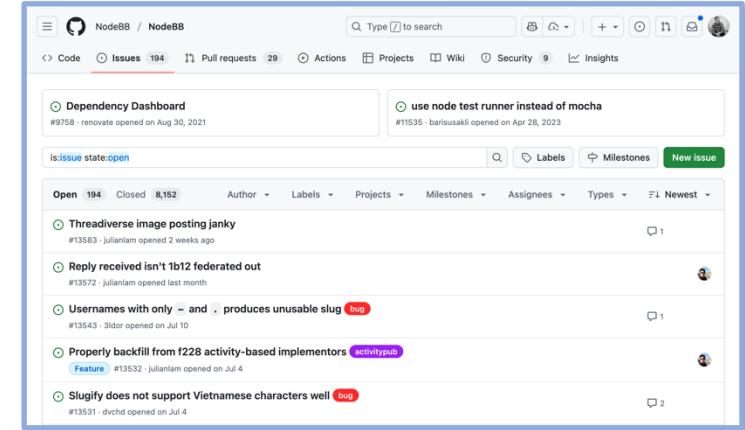
Run the whole suite of tests via `npm test` or an individual file via `npx mocha test/your_test_file.js`.

If you need to activate a certain plugin for testing as well, add the following block to your `config.json` file:

```
"test_plugins": [  
  "nodebb-plugin-xyz"  
]
```

Tip: Use discussion boards and issue trackers

- Are features unimplemented?
- Is the project still being maintained?
- Is someone else having the same issue?
- Found an issue with the code? **File a GitHub issue**
- Having a hard time getting some to work? Trying to change something? **Post to the NodeBB forums**
- Have a question about {Node, Redis, Express, ...}? **Post to StackOverflow or Slack.**



Tip: Use AI to explain parts of the code — but be careful

- Used carefully, AI tools can help you quickly tackle new codebases
- These tools fail confidently; expect errors and omissions, and cross-check against code, docs, and tests before trusting results.



- We will have a **whole lecture** on this new, emerging skill later in the course — for now, **experiment with AI, but don't rely on it**

Tip: Look at file structure, ownership, and history

- Files are not randomly named and organized. Directory structures and naming conventions reveal patterns.
- Inspect history to learn ownership and stability: identify contributors, recency of changes, and churn. Treat stale or recently rewritten files with caution.

A screenshot of a code blame interface. At the top, it shows "Code" and "Blame" tabs, with "Blame" selected. It indicates 142 lines (120 loc) and 3.69 KB. Below this is a color-coded timeline from "Older" (dark red) to "Newer" (dark green). The main area displays a snippet of code with commit history. The commits are:

- 9 years ago: ACP search updated to support...
- 6 years ago: refactor: async/await
- 5 years ago: feat: async/await admin/search
- 9 years ago: ACP search updated to support...
- 6 years ago: refactor: async/await
- 4 years ago: chore: eslint prefer-destructu...
- 9 years ago: ACP search updated to support...
- 9 years ago: Tests for admin search, simpli...
- 4 years ago: chore: eslint prefer-arrow-call...
- 9 years ago: Tests for admin search, simpli...
- 6 years ago: refactor: async/await file
- 4 years ago: chore: eslint prefer-arrow-call...
- 8 years ago: Use js instead of jst
- 9 years ago: Tests for admin search, simpli...
- 9 years ago: Use async instead of Promises
- 4 years ago: chore: eslint prefer-arrow-call...
- 7 years ago: style(eslint): match operator-ii...
- 4 years ago: chore: eslint prefer-arrow-call...

The code snippet includes imports for fs, path, sanitizeHTML, nconf, and winston, and defines a filterDirectories function that maps over directories, filters them, and excludes .js files.

A screenshot of a GitHub repository interface. The top navigation bar shows "CMU-313 / NodeBB". The "Code" tab is selected. Below the navigation is a breadcrumb trail: "main / NodeBB / src /". A search bar and a "Go to file" button are also present. The main content area lists files with their last commit details:

Name	Last commit message	Last commit date
..		
activitypub	fix: clearTimeout if item is evicted from cache	last month
admin	Request, Fetch, closes #10341 (#12236)	2 years ago
api	test: fix openapi	last month
cache	Merge remote-tracking branch 'origin/develop' into ...	8 months ago
categories	fix: use sharp to convert svg to png, closes #13534	last month
cli	feat: upgrade commander, get rid of custom color ...	5 months ago
controllers	fix: apply sanitizeSvg to regular uploads and upload...	last month
database	fix: sql injection in sortedSetScan	3 months ago

How to build, test, and refine mental models



Examine
artifacts without
running code



Probe
running system to
observe behavior



Modify
code, rebuild, and
assess impact



Probe to test your mental model

🧪 Hypothesis → Experiment

- Introduce a probe to observe the system at a given seam or entry point
- Use the observed behavior to confirm or refute your hypothesis
- Gradually build confidence in your understanding of system behavior
- **Example:**
When I click X, handler Y runs
→ Set a breakpoint in Y then trigger X

🔬 Probes & Triggers

- Add [breakpoints](#), [logpoints](#), and step
- Logging: `./nodebb dev`
- Print statements
- Bruno / Postman / curl / httpie
- Database viewers

🏆 Goal

- One confirmed or refuted hypothesis
- One short note (trigger → code path → signal)
- One next probe or modification

Tip: Instrument the source code

- **Print debugging**

- Quick and easy
- Cons: need to rebuild + restart; easy to commit by accident

```
console.log('Administrator found, skipping Admin setup');
```

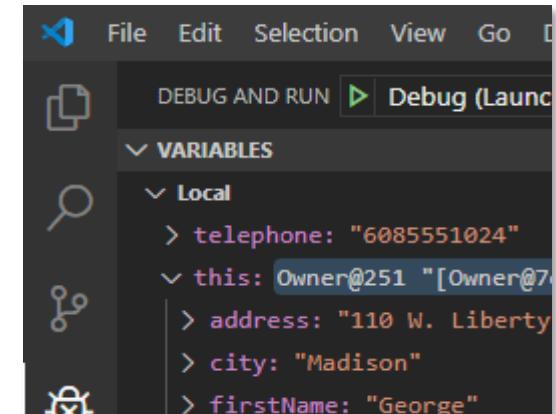
- **Structured logging**

- Add levels, timestamps, and context; better for collecting data in deployment
- Cons: need to rebuild + restart

```
winston.warn(`Flooding detected! Calls : ${socket.callsPerSecond}, Duration : ${socket.elapsedTime}`);
```

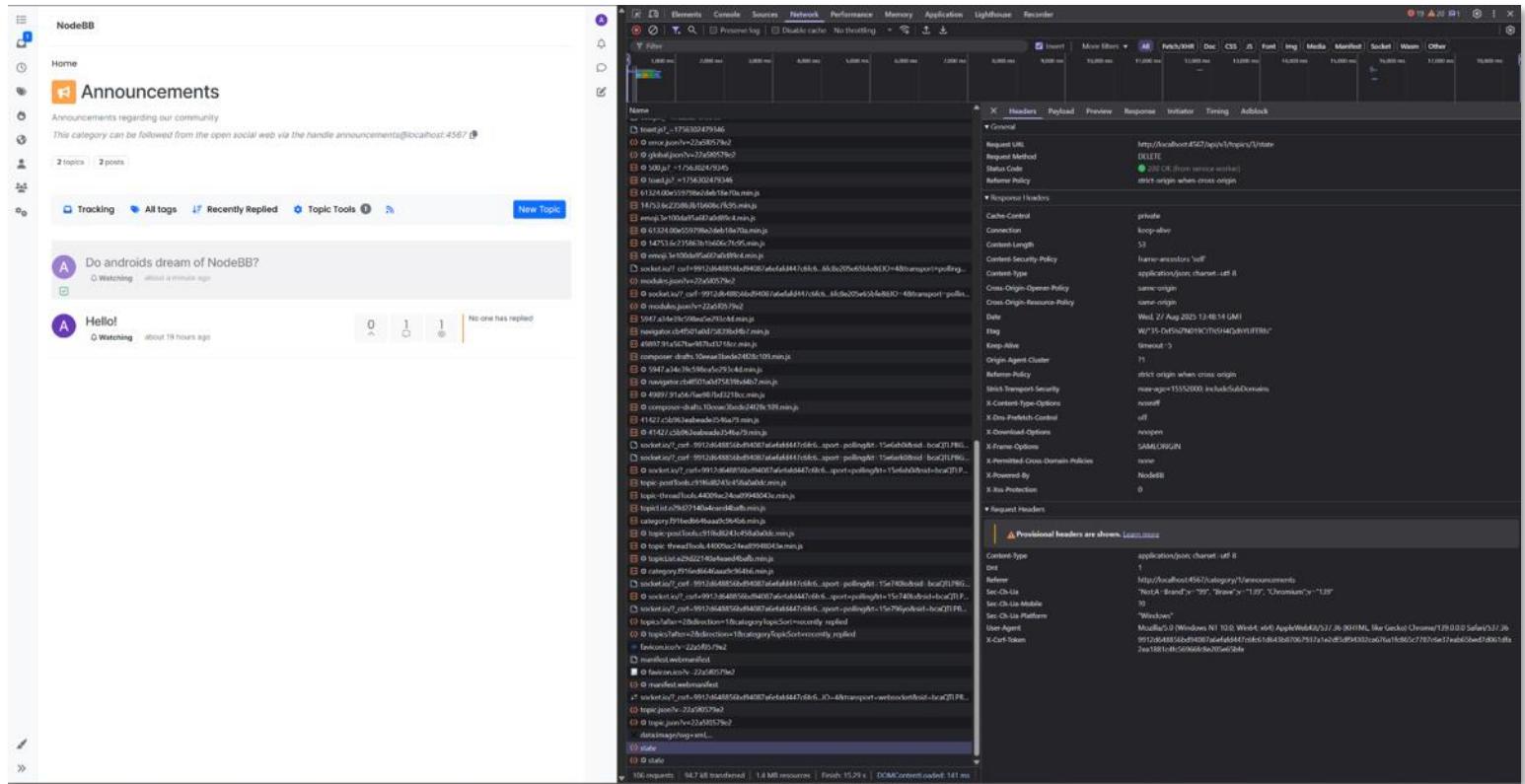
- **Debuggers**

- Inspect locals, call stack, evaluate expressions
- Add breakpoints as you go; no need to rebuild + restart
- No changes to the code means no risk of accidental changes
- We will explore the debugger in more depth later in the course



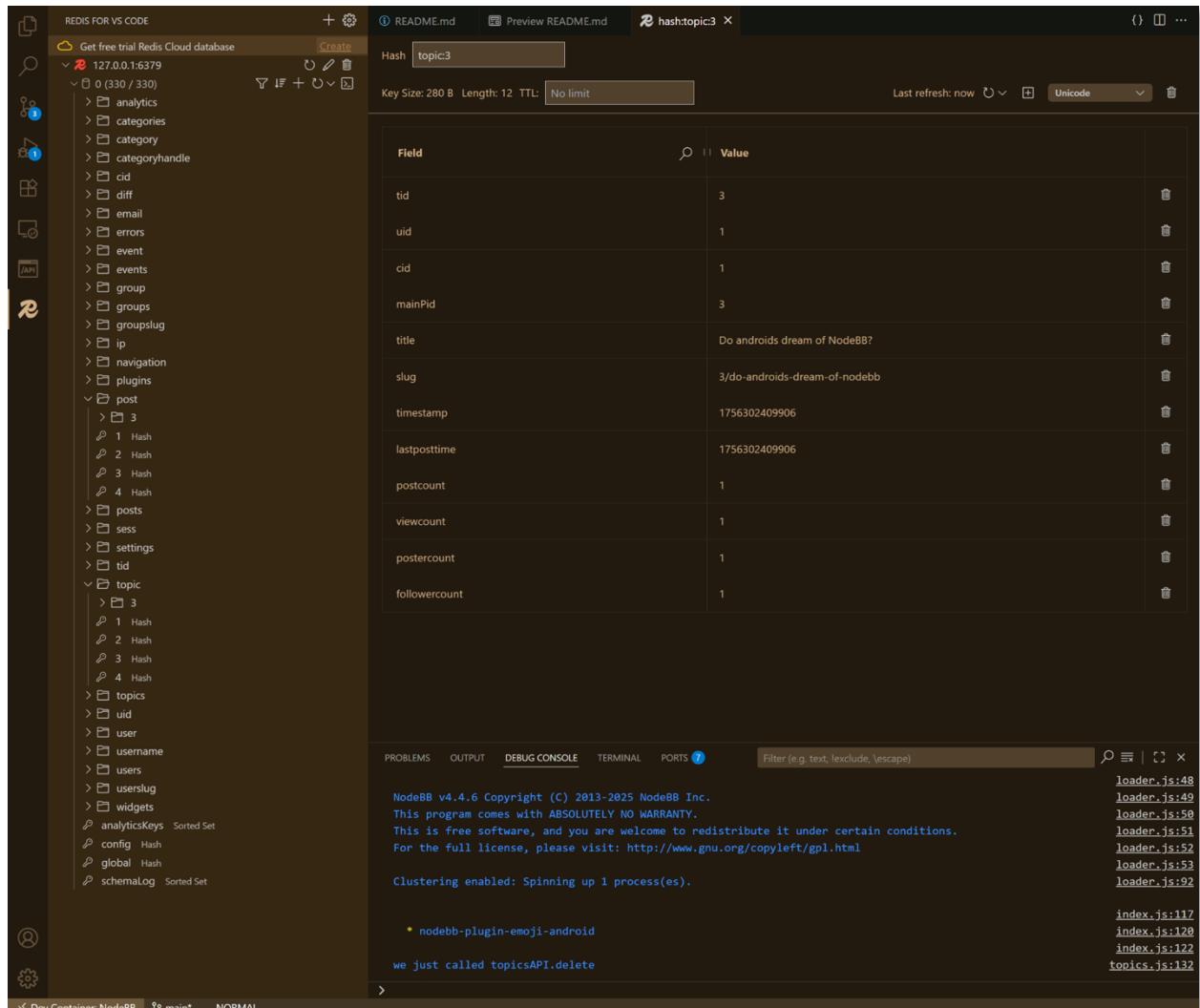
Tip: Use developer tools in the browser to spy on traffic

- Spy on web traffic while you use the app
- [Chrome DevTools](#) (also used by Brave)
- [Firefox Dev Tools](#)
- [Safari Web Inspector](#)
- **Bonus:** Use [Bruno](#), [Postman](#), [httpie](#), or [curl](#) to trigger API requests



Tip: Peek at the database

- Use the **Redis extension** that's provided with the DevContainer
- Perform an action (e.g., create or delete a topic) and watch which keys / fields change
 - filter by **prefix** to keep things manageable (topic:*, post:*, user:*)
- Use to confirm or refute your hypotheses about data flow



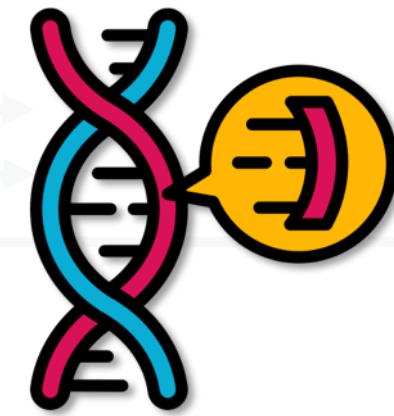
How to build, test, and refine mental models



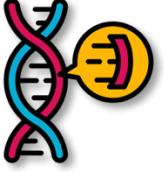
Examine
artifacts without
running code



Probe
running system to
observe behavior



Modify
code, rebuild, and
assess impact



Modify code to validate your model

Plan and execute your change

- What **behavior** should change if your model is correct?
- What's the **simplest change** that you can make?
- What **signal** can you observe? (user interface, API, logs, database, test case)
- Rebuild the code and see what happens!
- Tip: **delete debugging** is a powerful tool

Assess impact

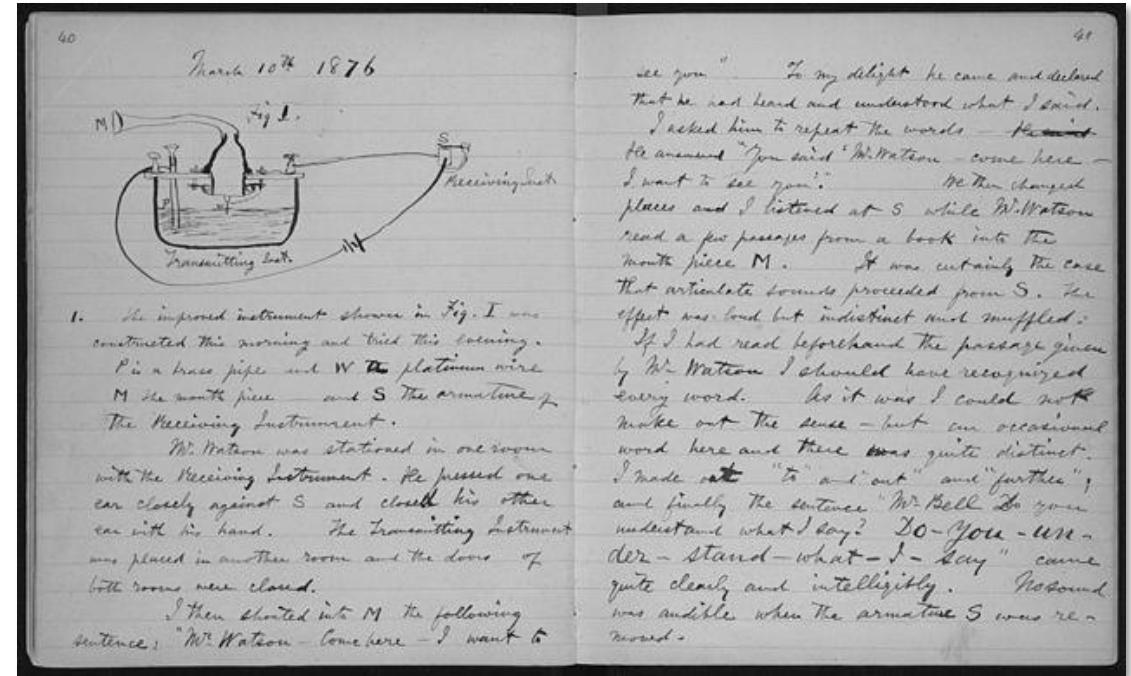
- Did the **predicted signal** change?
- If yes, your model **holds** for now.
- If not, you need to **revise** it.

Goal

- One change with a clear effect
- A note of what it confirms or refutes
- A next step (examine, probe, modify)

Document and share your findings!

- Update README and docs
 - Or better: use a **Developer Wiki**
 - Use [Mermaid](#) for diagrams
- Collaborate with others
 - use [LiveShare](#) to debug, explore, and program collaboratively
- Include negative results, too!



Next Time: 737-MAX Case Study

