GitHub and Git in Practice

By
Vijaya Nandini M

GitHub and Git in Practice

- Let’s shift our focus more towards GitHub, understanding workflow patterns using Git and features specific to GitHub.

GitHub and Git in Practice

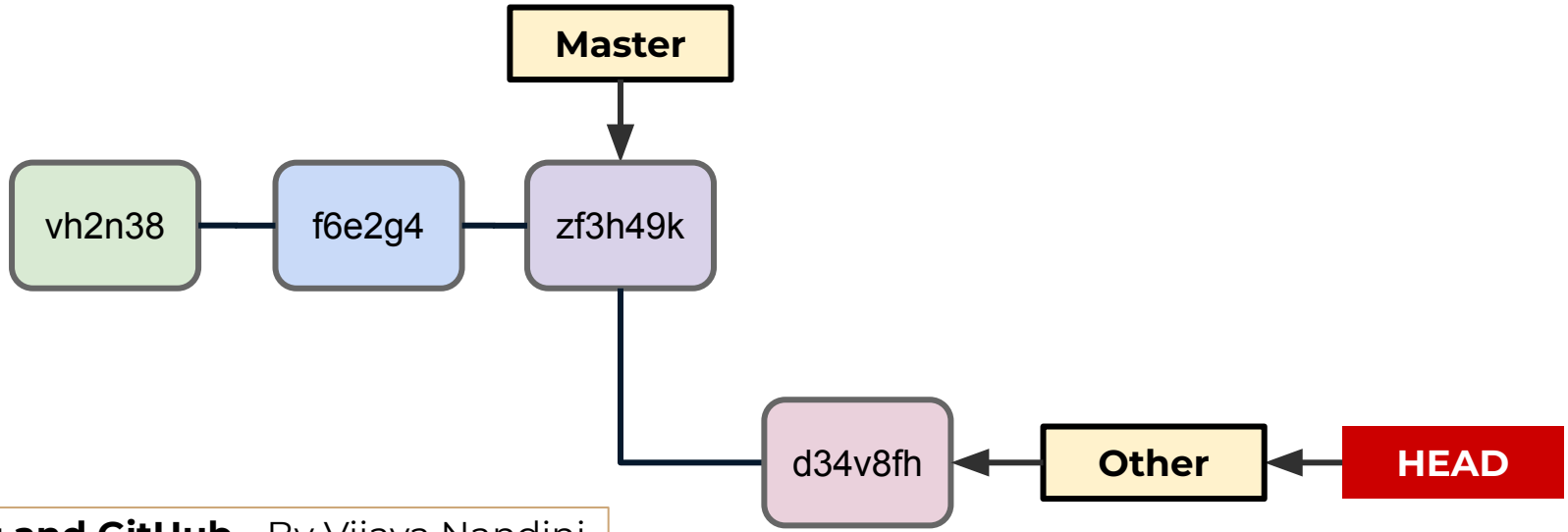
- **Topics:**
  - Using git stash
  - Push and Pull and Git Remote Review
  - Common Workflow Patterns
  - GitHub Repository Tour
  - GitHub Pull Requests
  - Forking on GitHub
  - GitHub Actions

Git Stash

GitHub and Git in Practice

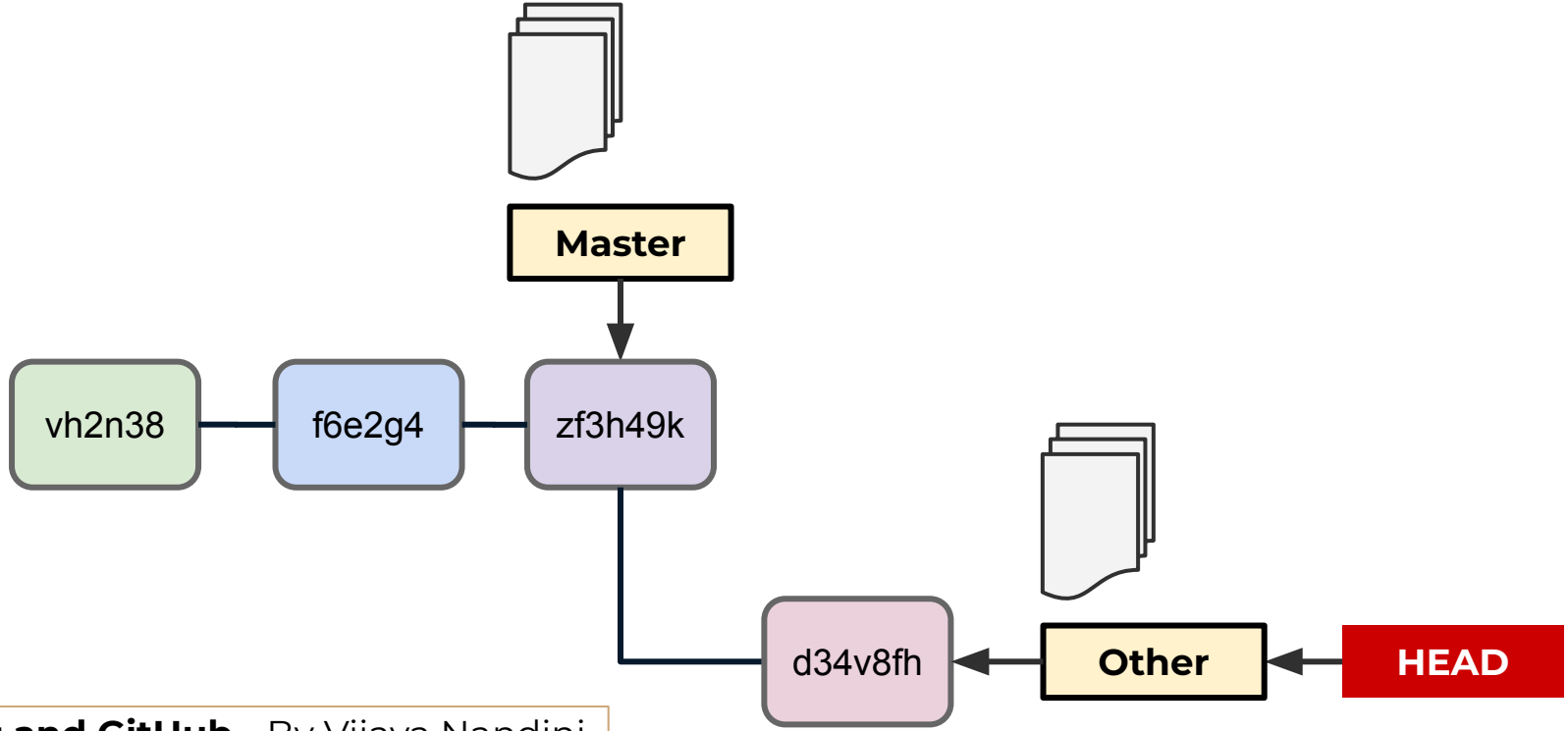
- **Git stash**
  - Temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.
  - Stashing is useful when you find yourself needing to quickly switch branches to work on something else, but are in the middle of changing a file.

GitHub and Git in Practice



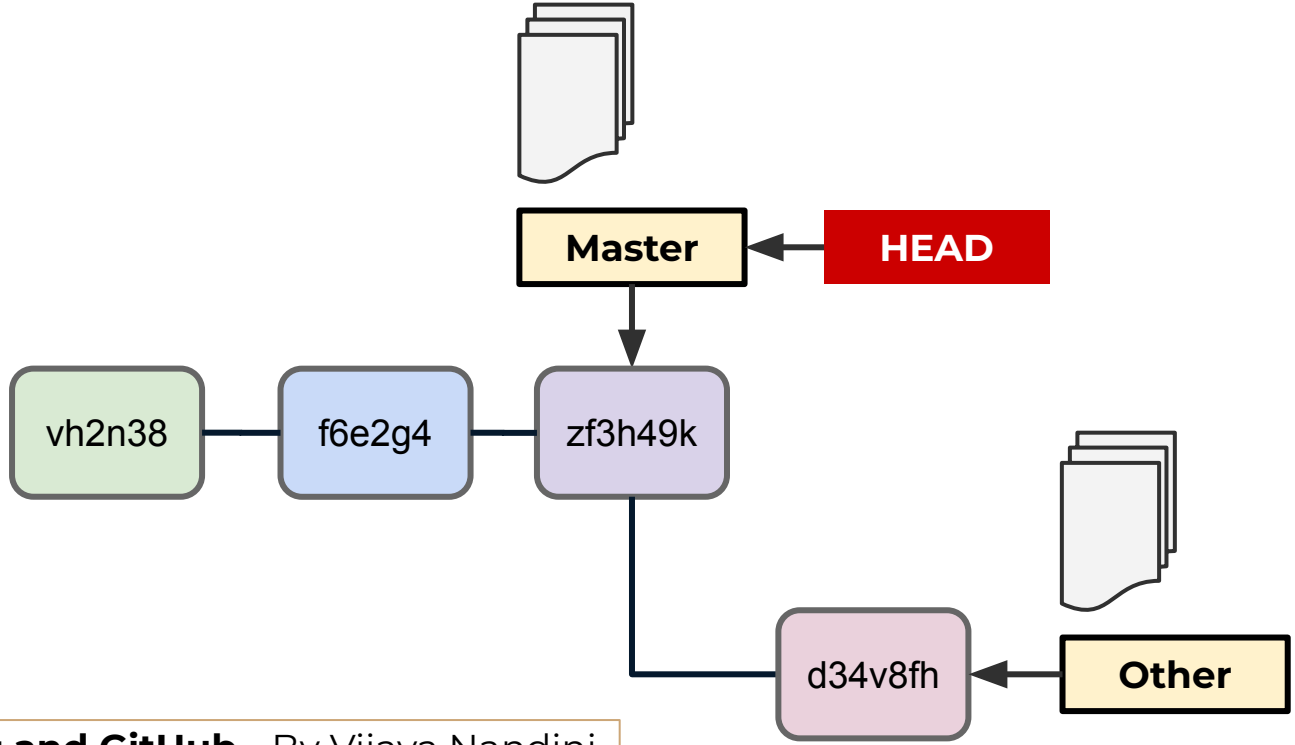
GitHub and Git in Practice

- Working on Files



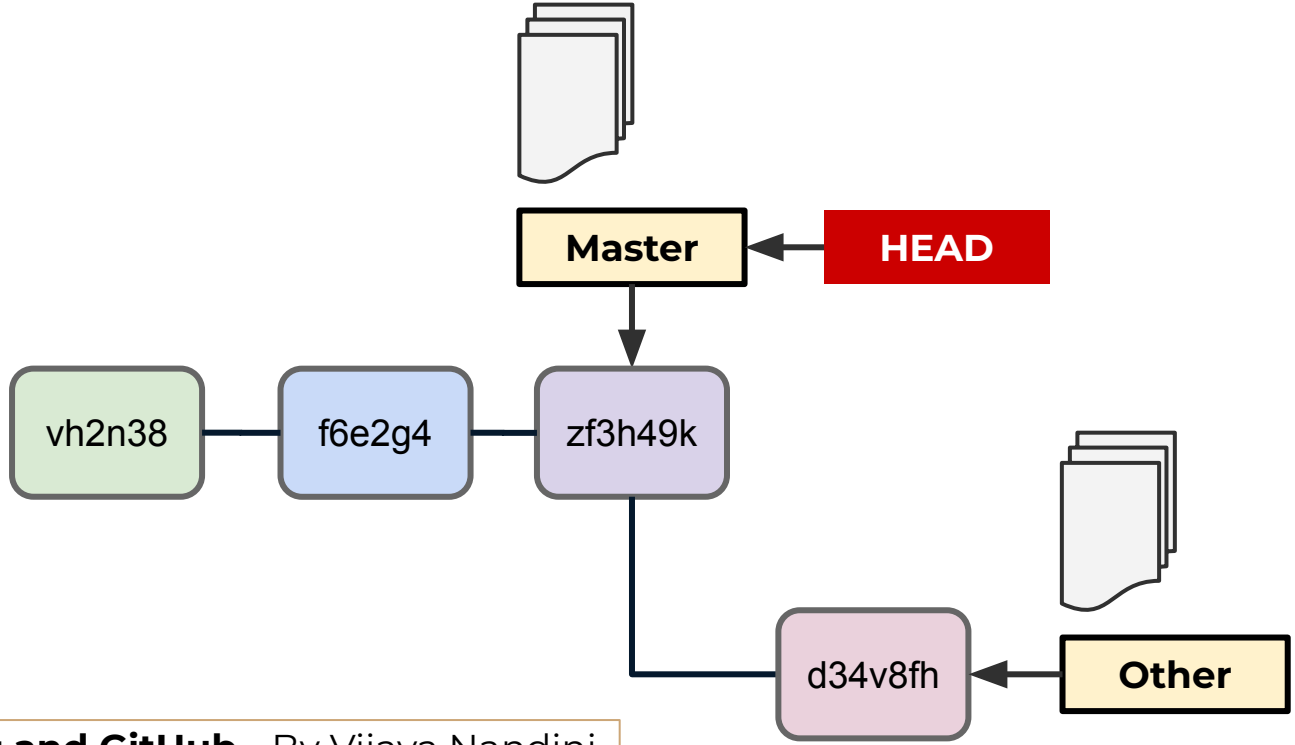
GitHub and Git in Practice

- git switch master



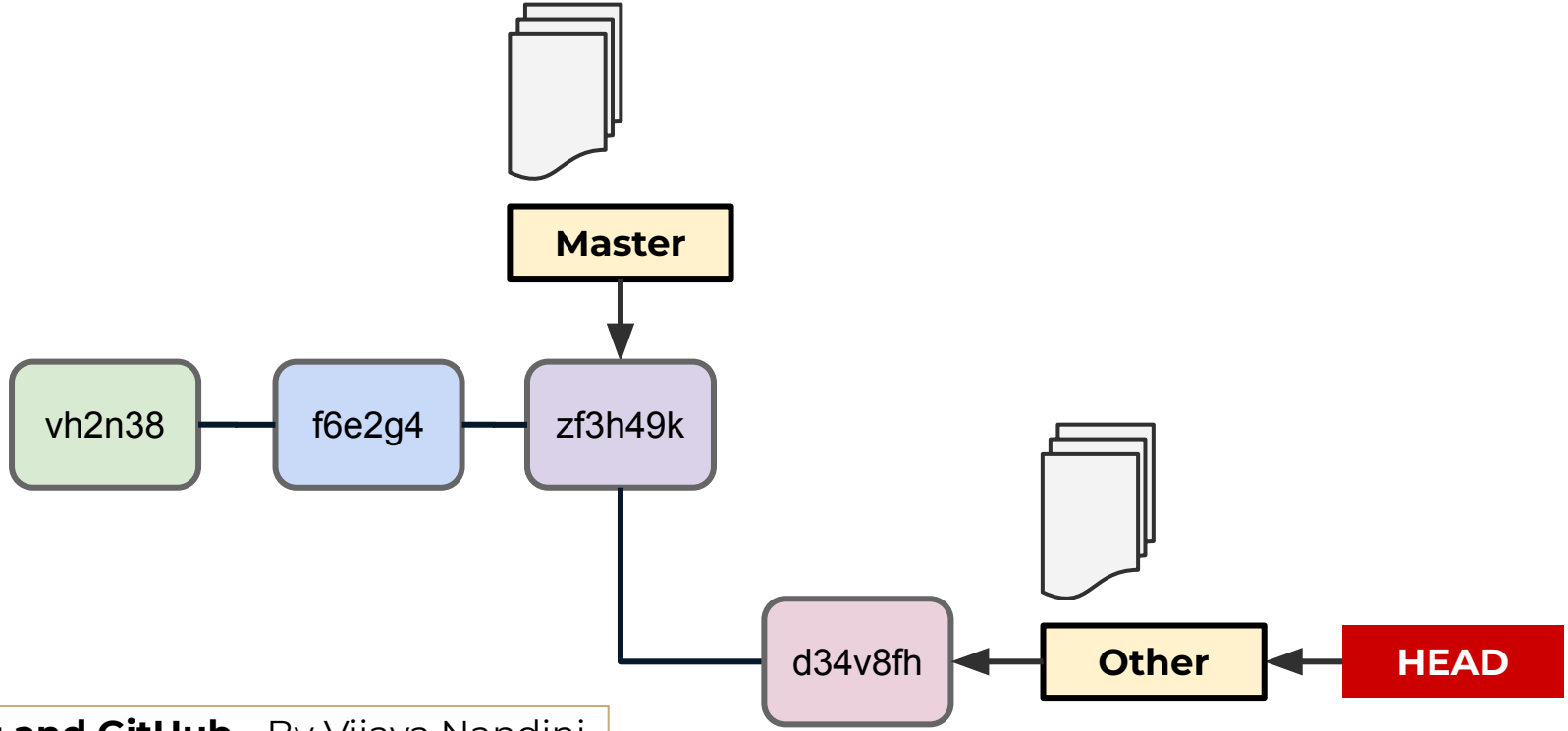
GitHub and Git in Practice

- But what happens when you had changes?



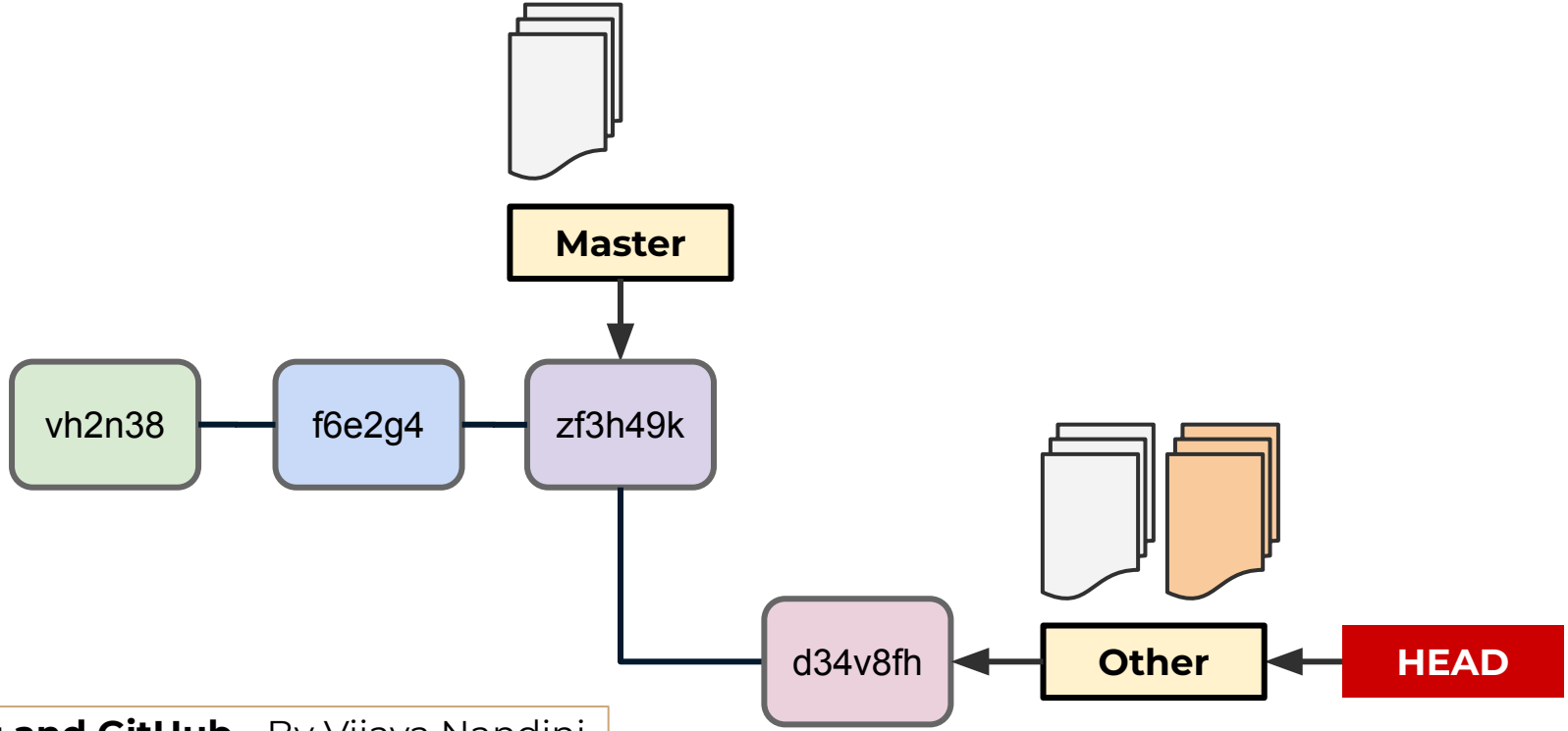
GitHub and Git in Practice

- Working on Files



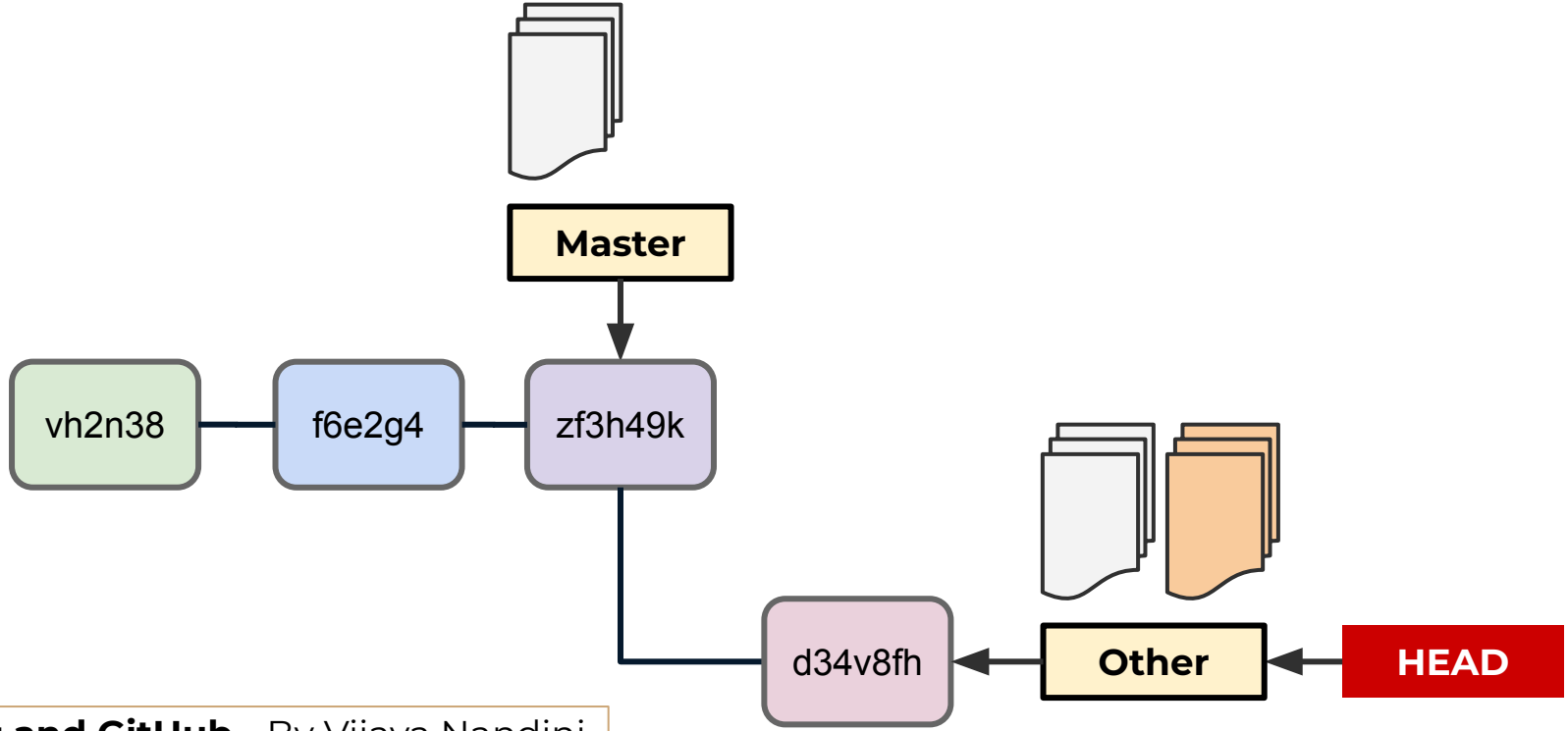
GitHub and Git in Practice

- Created new non-conflict files on Other



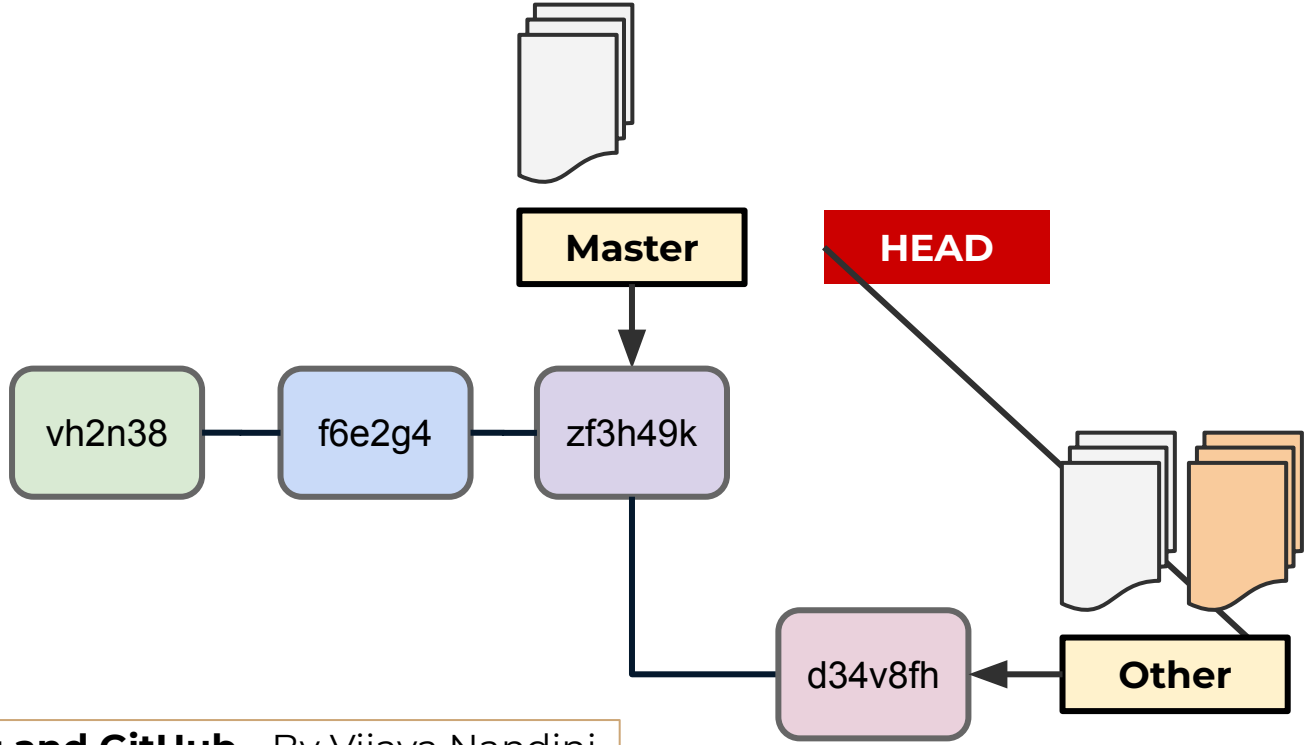
GitHub and Git in Practice

- You have also not even committed them...



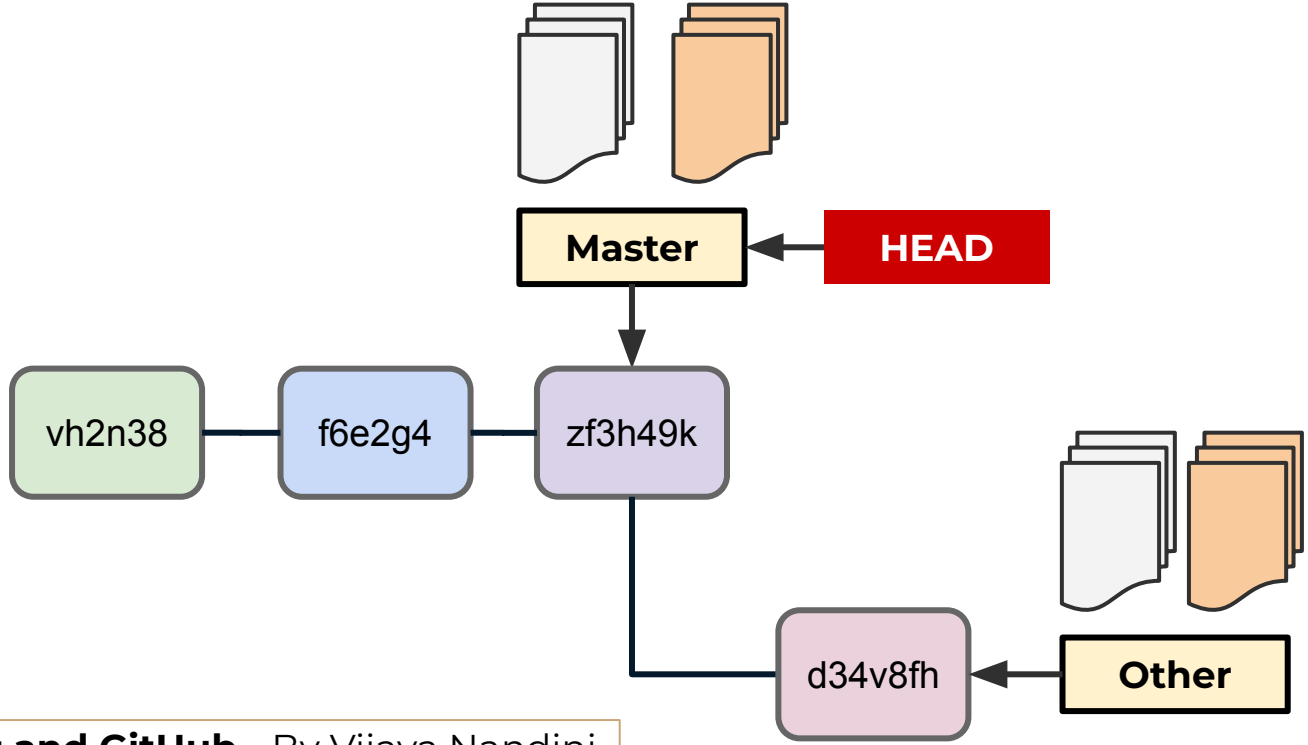
GitHub and Git in Practice

- If you switch over with **git switch master...**



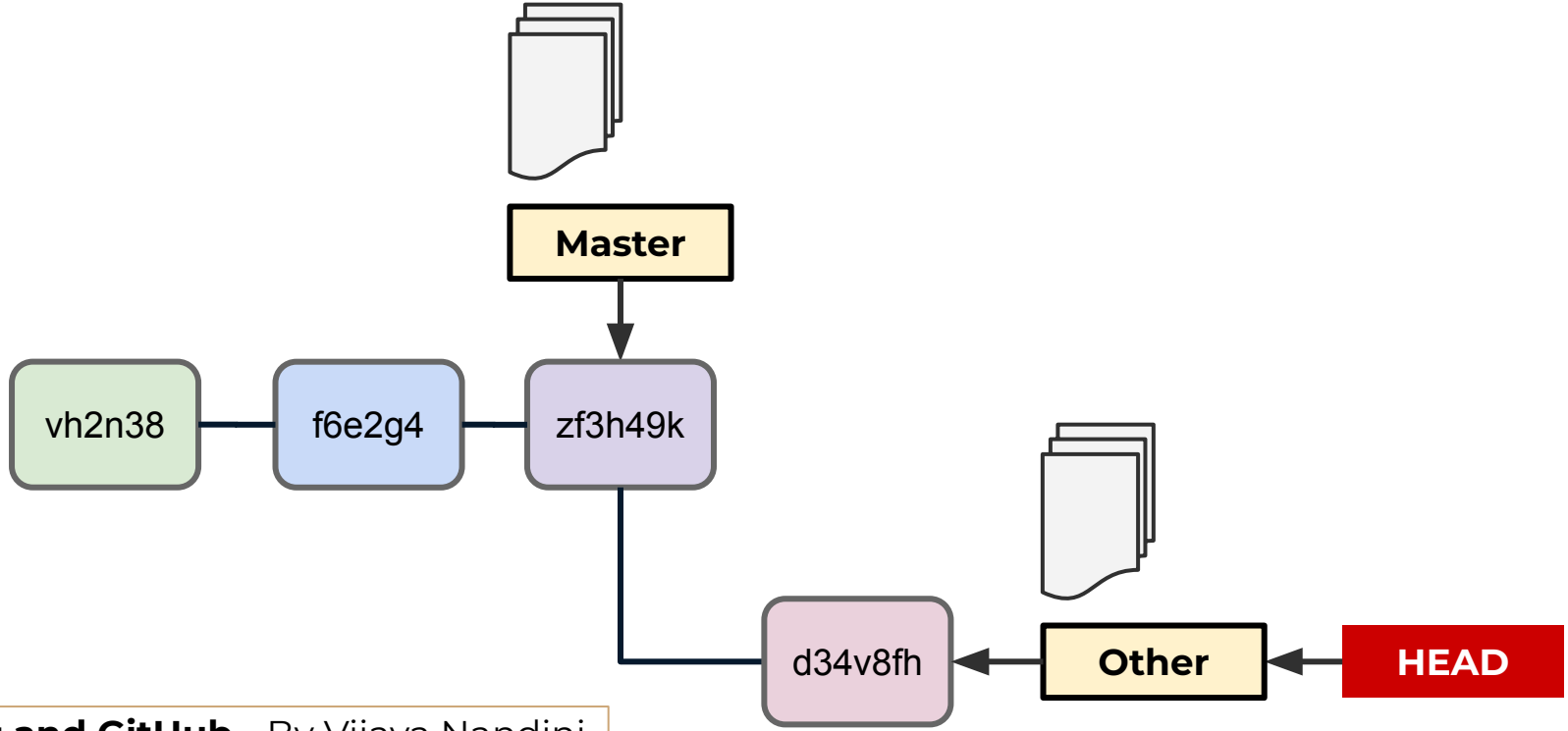
GitHub and Git in Practice

- Non-conflict files/changes come too!



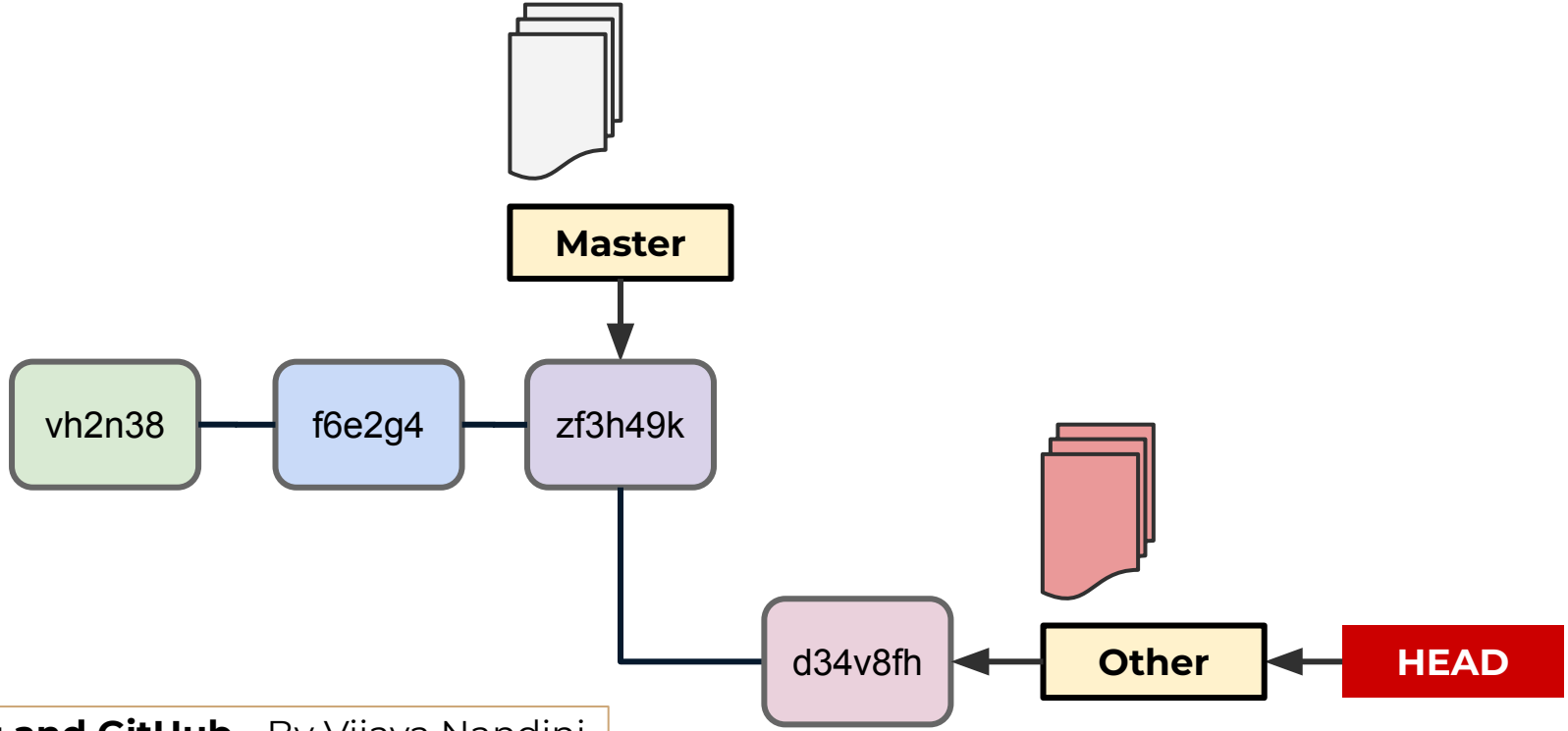
GitHub and Git in Practice

- Or if you have a conflicting change



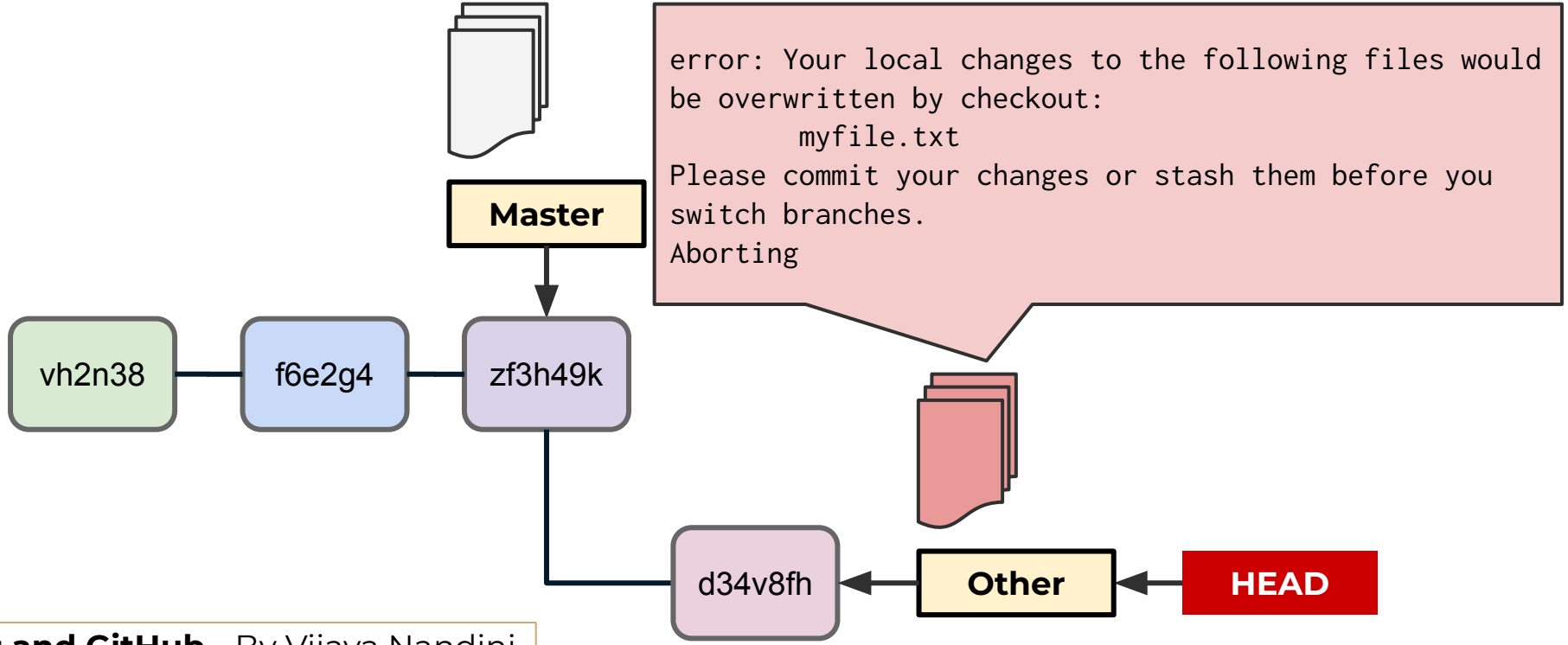
GitHub and Git in Practice

- Or if you have a conflicting change



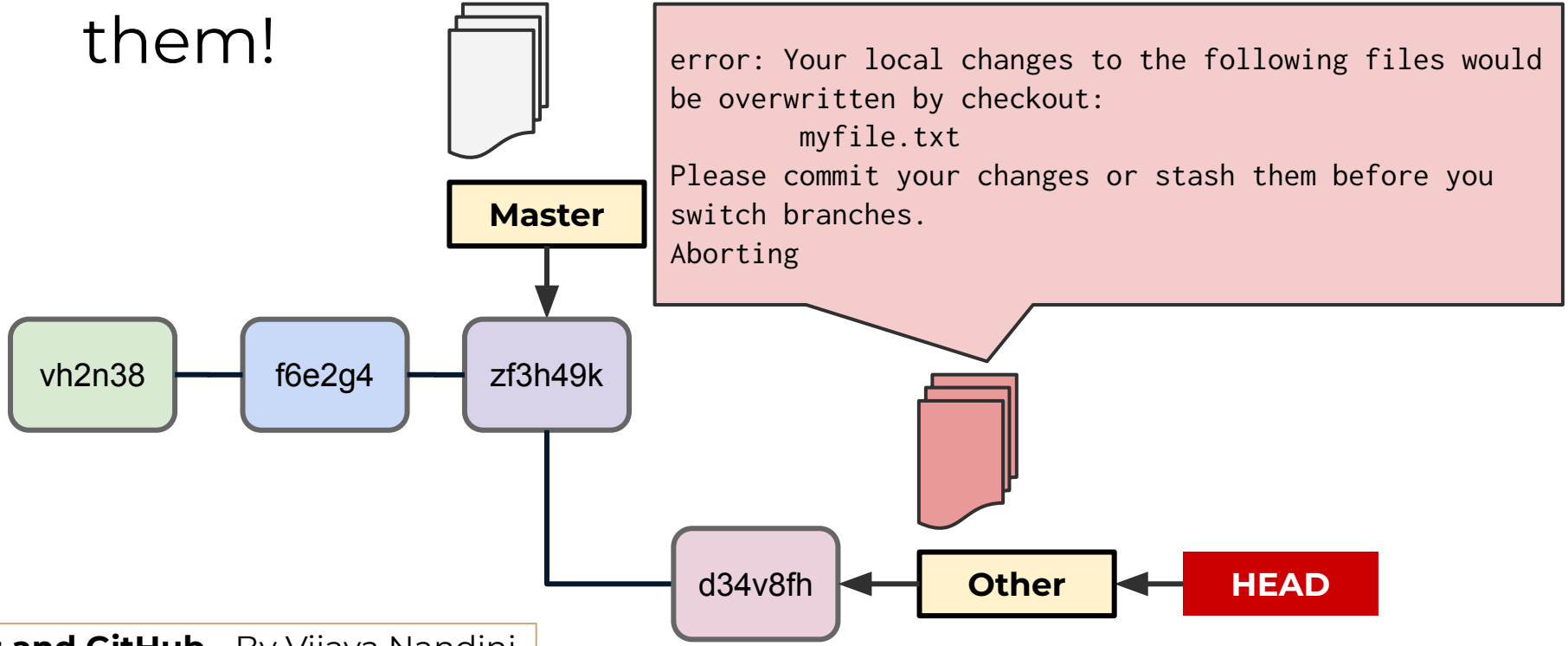
GitHub and Git in Practice

- Or if you have a conflicting change



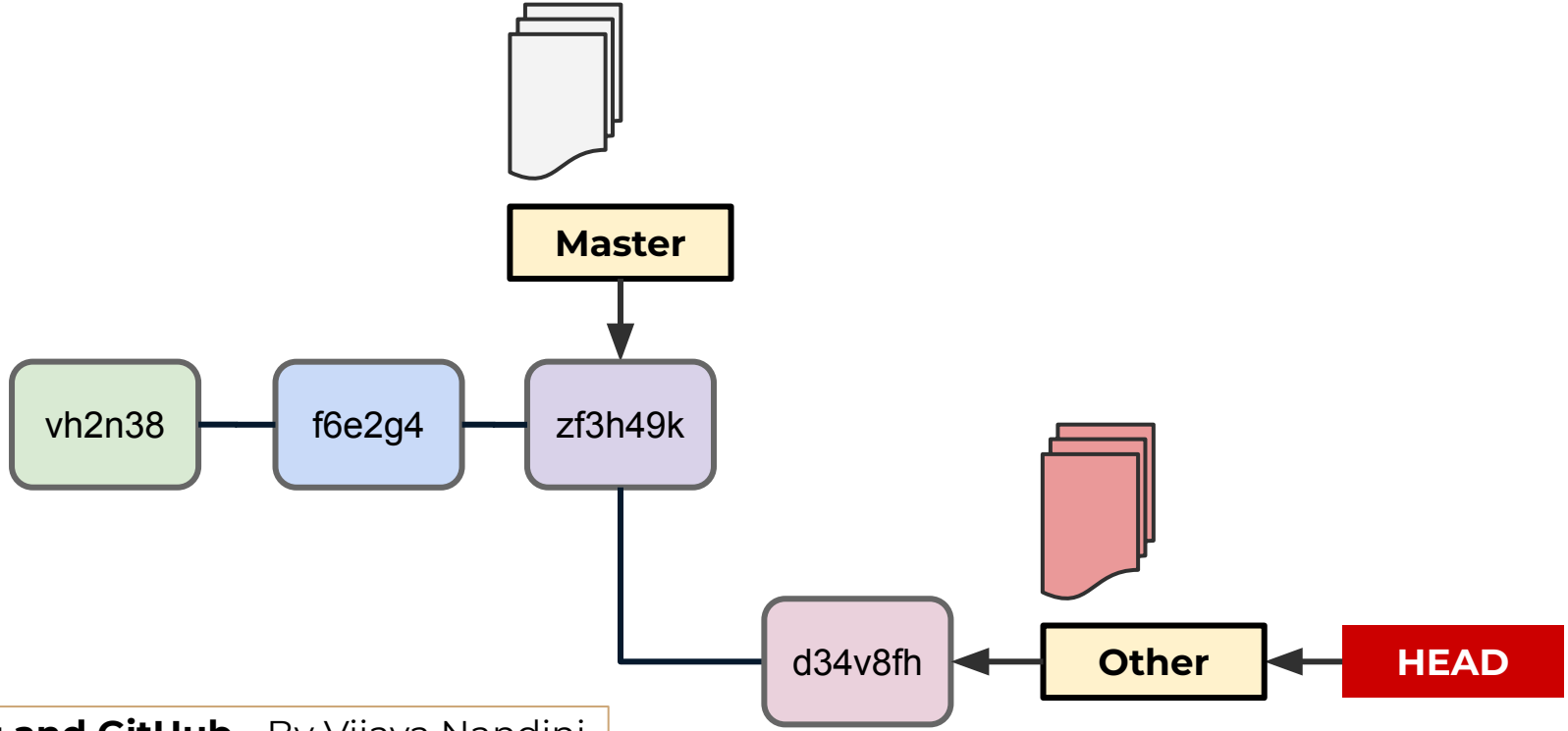
GitHub and Git in Practice

- We can stash our changes to preserve them!



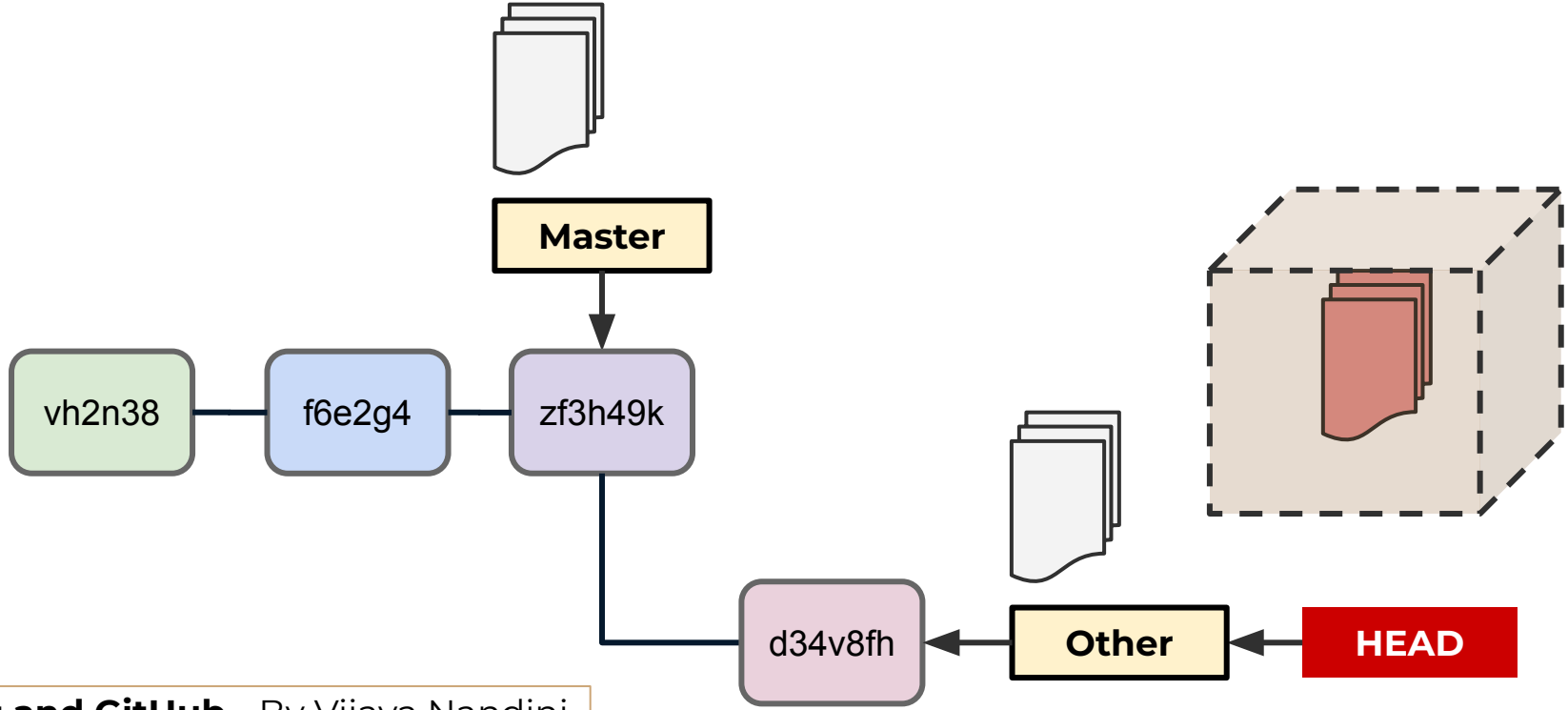
GitHub and Git in Practice

- You can stash changes without a commit



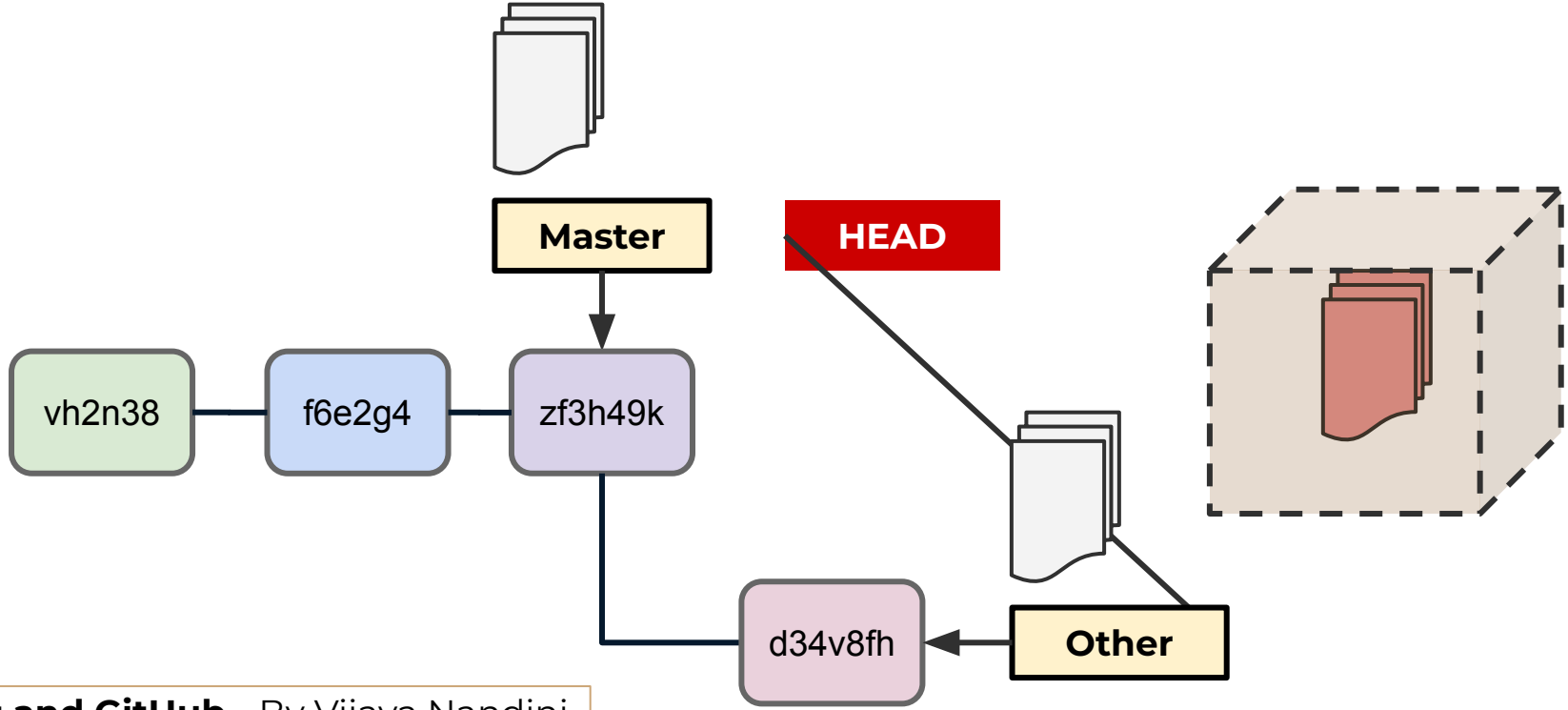
GitHub and Git in Practice

- git stash



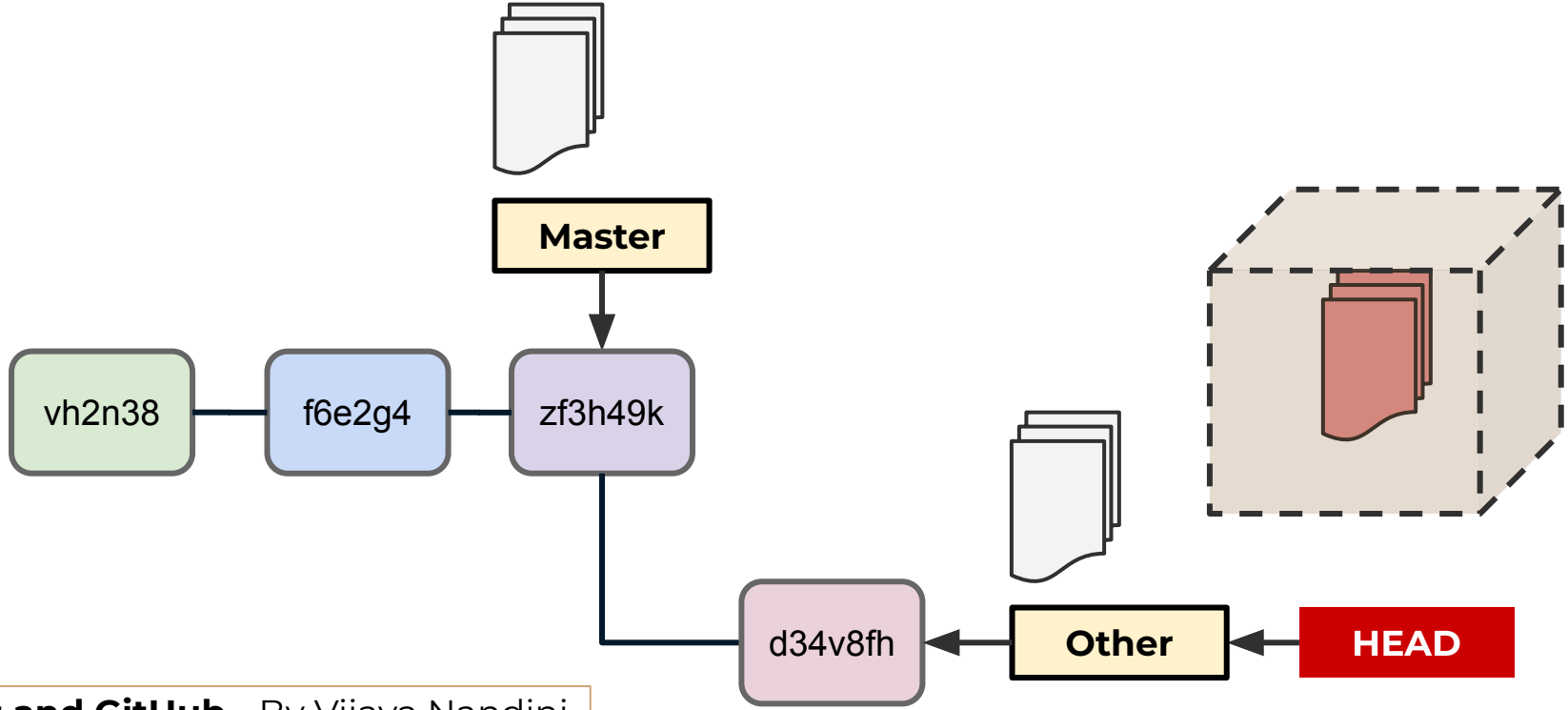
GitHub and Git in Practice

- git switch master



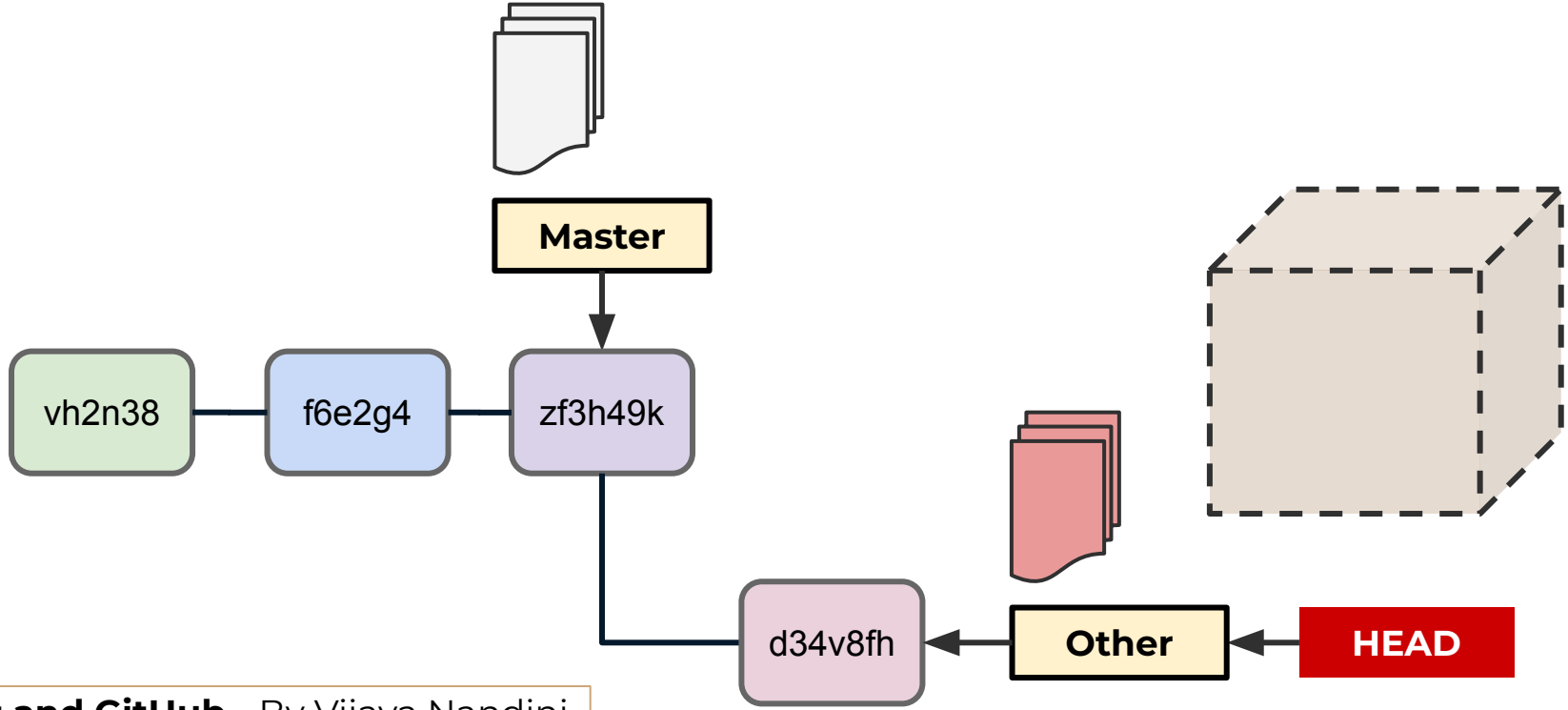
GitHub and Git in Practice

- git switch other



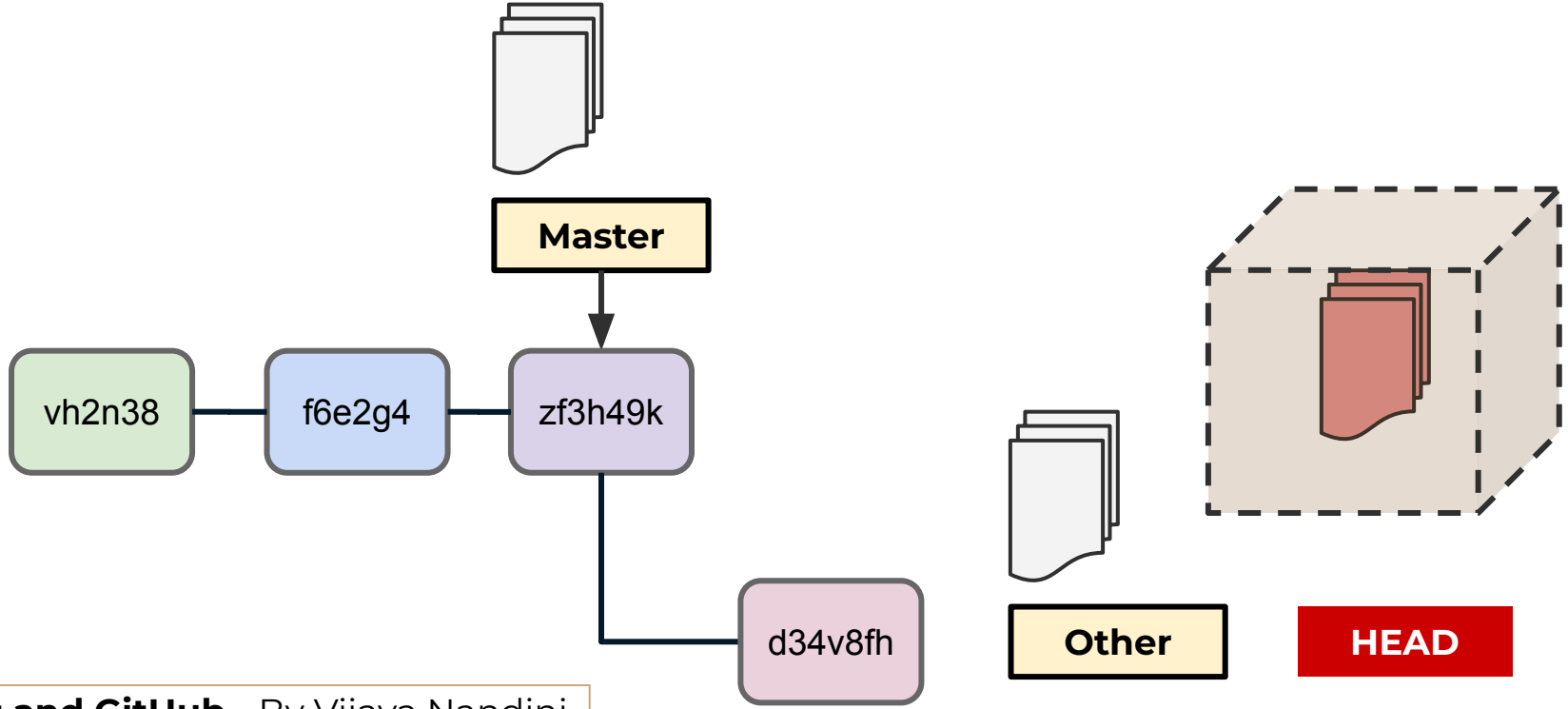
GitHub and Git in Practice

- git stash pop



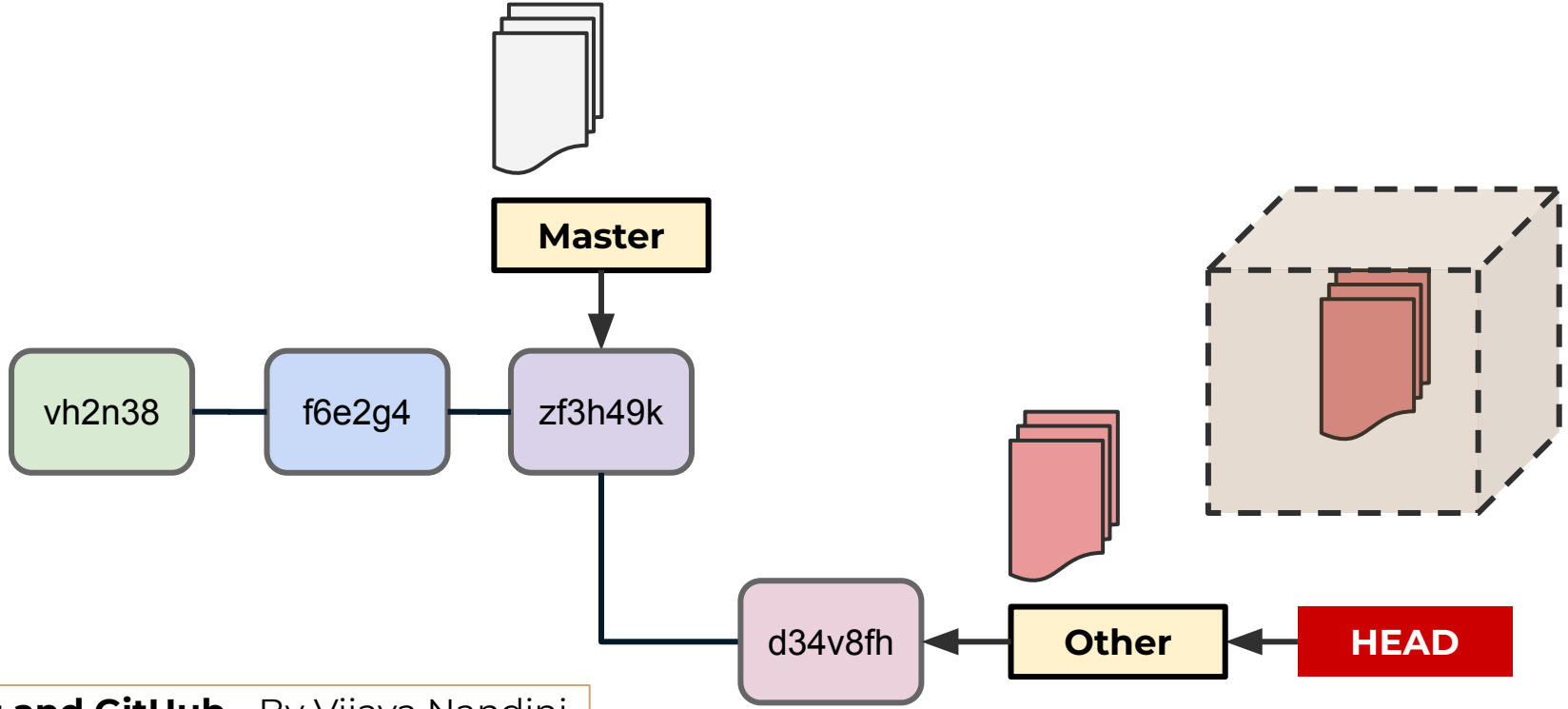
GitHub and Git in Practice

- git switch other



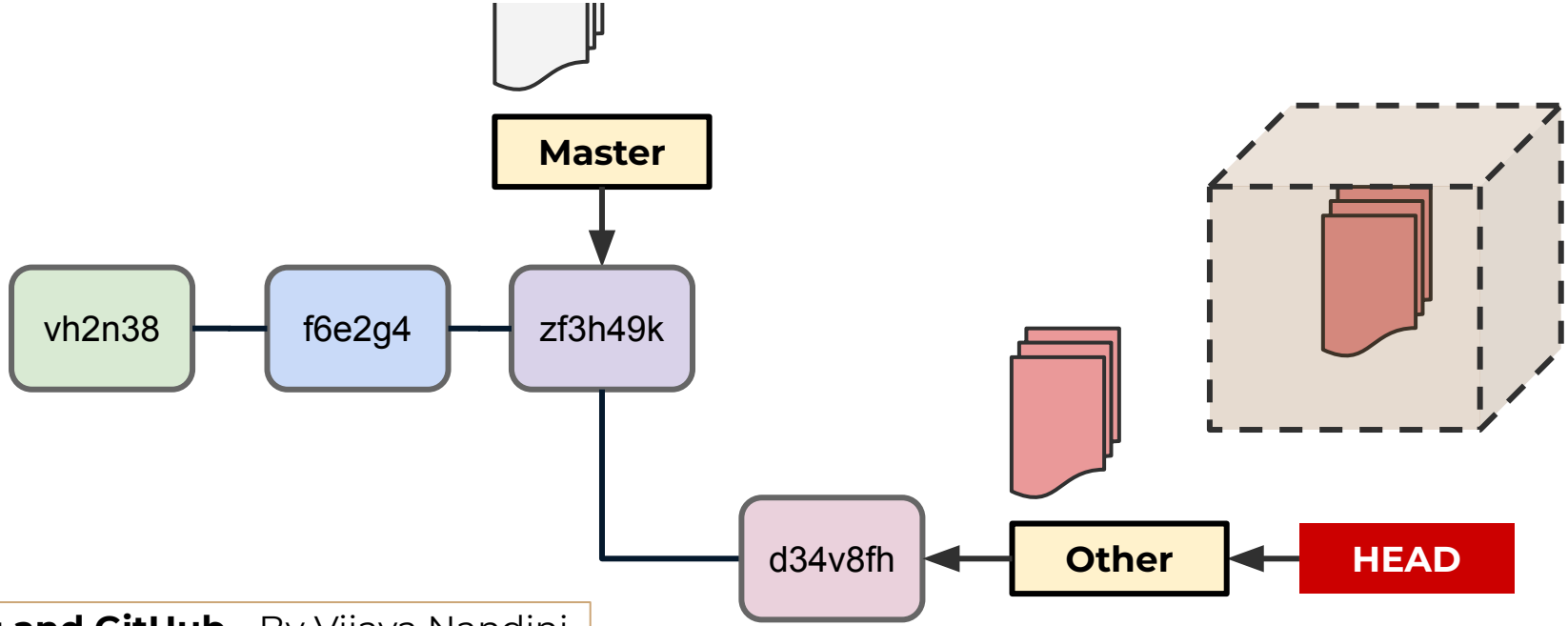
GitHub and Git in Practice

- git stash apply



GitHub and Git in Practice

- Useful if you want to apply the same stashed changes to multiple branches.



GitHub and Git in Practice

- Note that by default Git won’t stash changes made to untracked or ignored files.

GitHub Push and Pull Review

GitHub and Git in Practice

- Recall our discussions of GitHub repositories as a “remote”.
- Let’s clarify what **git remote** is more formally:
- The **git remote** command lets you create, view, and delete connections to other repositories. Remote connections are more like bookmarks rather than direct links into other repositories.

GitHub and Git in Practice

- The remote is a URL where the host repository lives, such as a GitHub URL.
- As we've seen before, in some cases that URL is hidden (private repository) or requires permissions to directly access and interact with.
- You can also have *multiple* git remote configurations.

GitHub and Git in Practice

- Running just **git remote** will list the remote connections you have.
- Running **git remote -v** will list the remote connections along with their URL.
- We can add or remove remote repositories with the commands:
  - **git remote add <name> <url>**
  - **git remote rm <name>**

GitHub and Git in Practice

- When working with a remote repository such as GitHub, we often have files that we don’t want to track or push, such as a SQL database which already tracks its own changes or secret files like API tokens.
- The **.gitignore** file is a file we can create to tell Git to ignore certain files or even entire directories.

GitHub and Git in Practice

- Creating a file called **.gitignore** at the root of the Git repository, and then you can define patterns to ignore:
  - **Mypasswords.txt**
    - Ignores this specific file
  - **directory_name/**
    - Ignores everything in this directory
  - ***.sql**
    - Ignores anything ending with .sql

Common Workflow Patterns

GitHub and Git in Practice

- Common Workflow Patterns
  - Single Branch Workflow
  - Branch Based Development

GitHub and Git in Practice

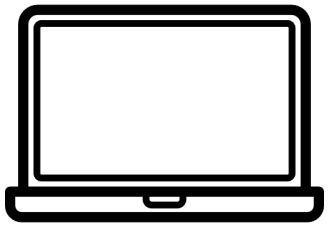
- **Single Branch Workflow**
  - Sometimes a common workflow for beginners, a single branch just does all the work on the master branch, using push and pull requests to update code on the main branch.

GitHub and Git in Practice

- **Single Branch Workflow**
  - This essentially treats Git as more of a save tool or online backup than a fully featured version control system with branching ability.
  - Let’s take a look at the workflow and then discuss its drawbacks...

GitHub and Git in Practice

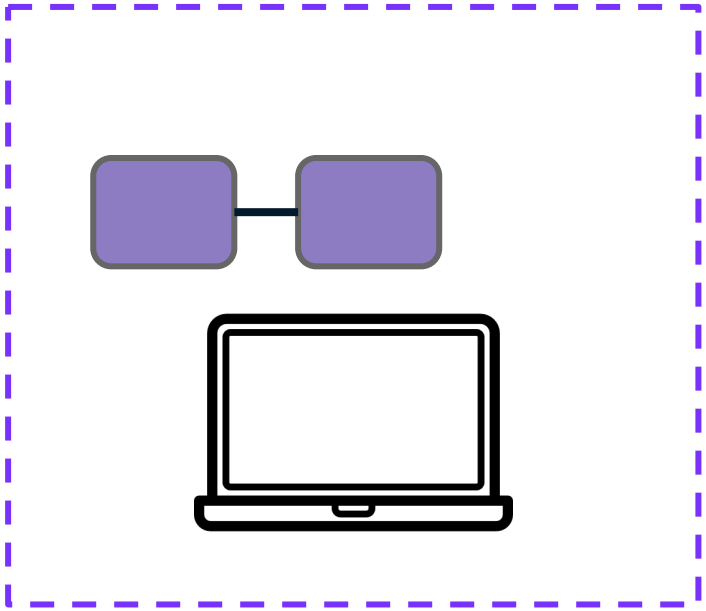
- **Single Branch Workflow**



GitHub

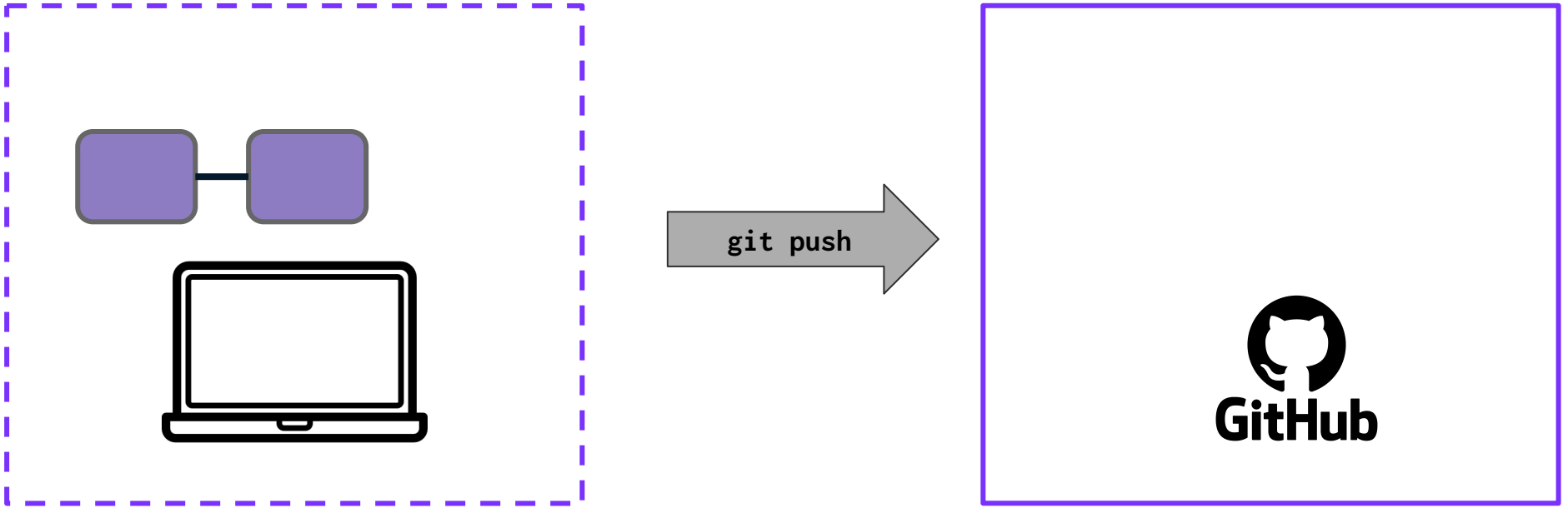
GitHub and Git in Practice

- **Single Branch Workflow**



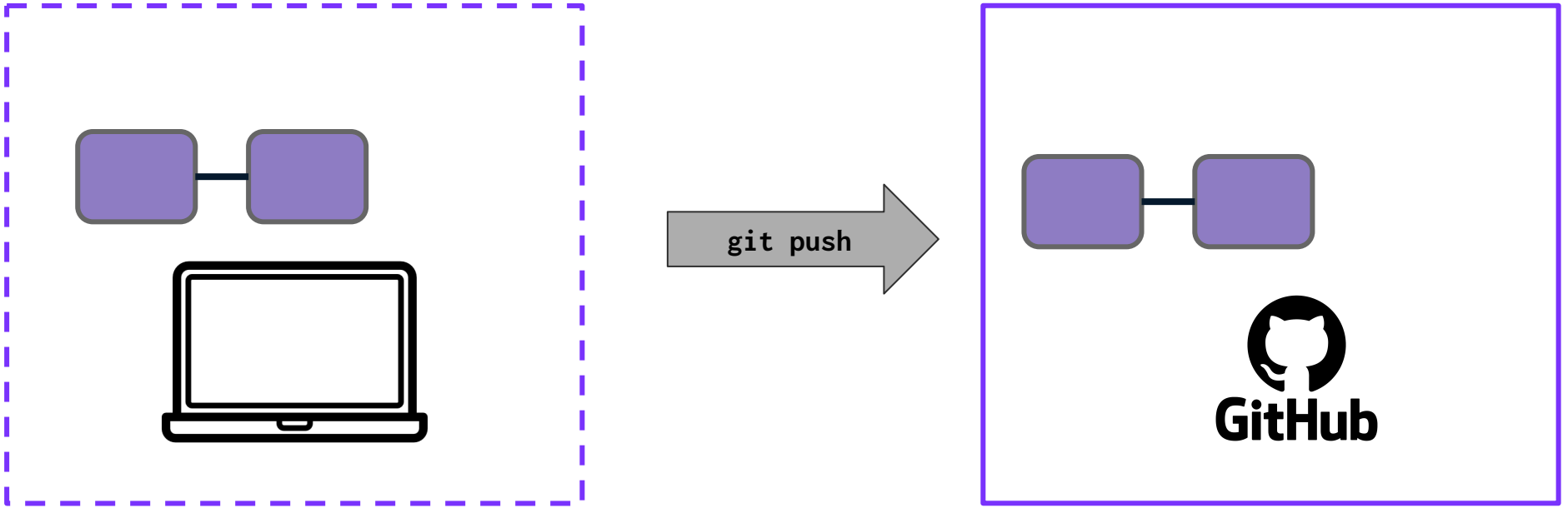
GitHub and Git in Practice

- **Single Branch Workflow**



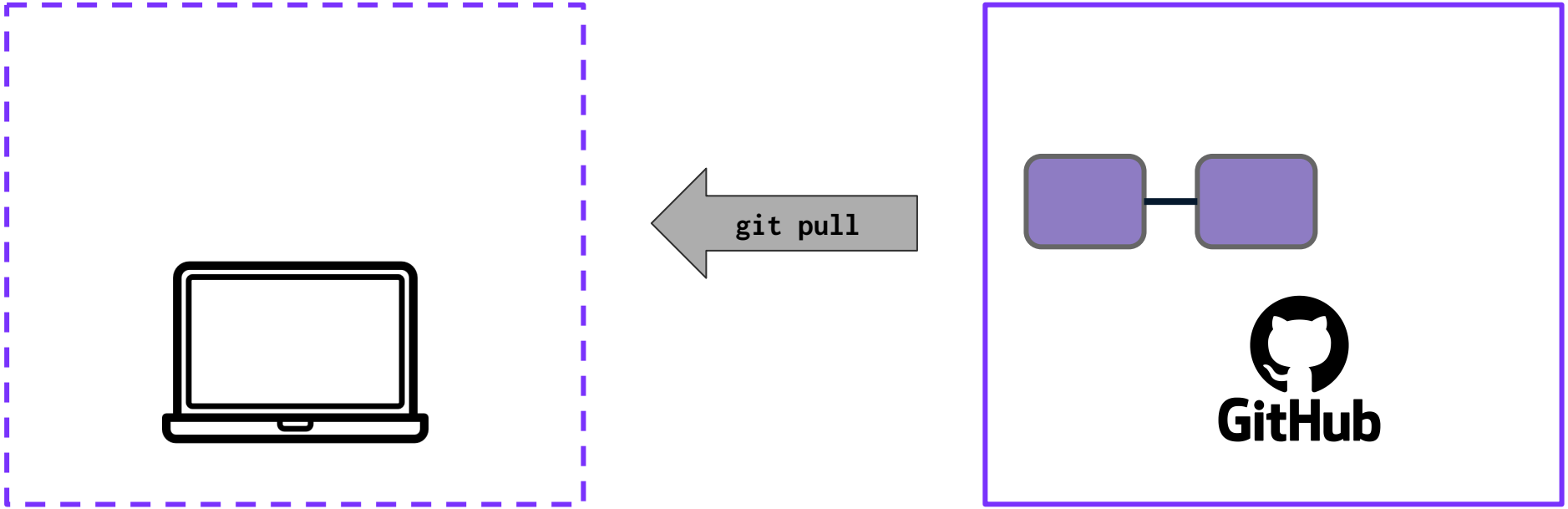
GitHub and Git in Practice

- **Single Branch Workflow**



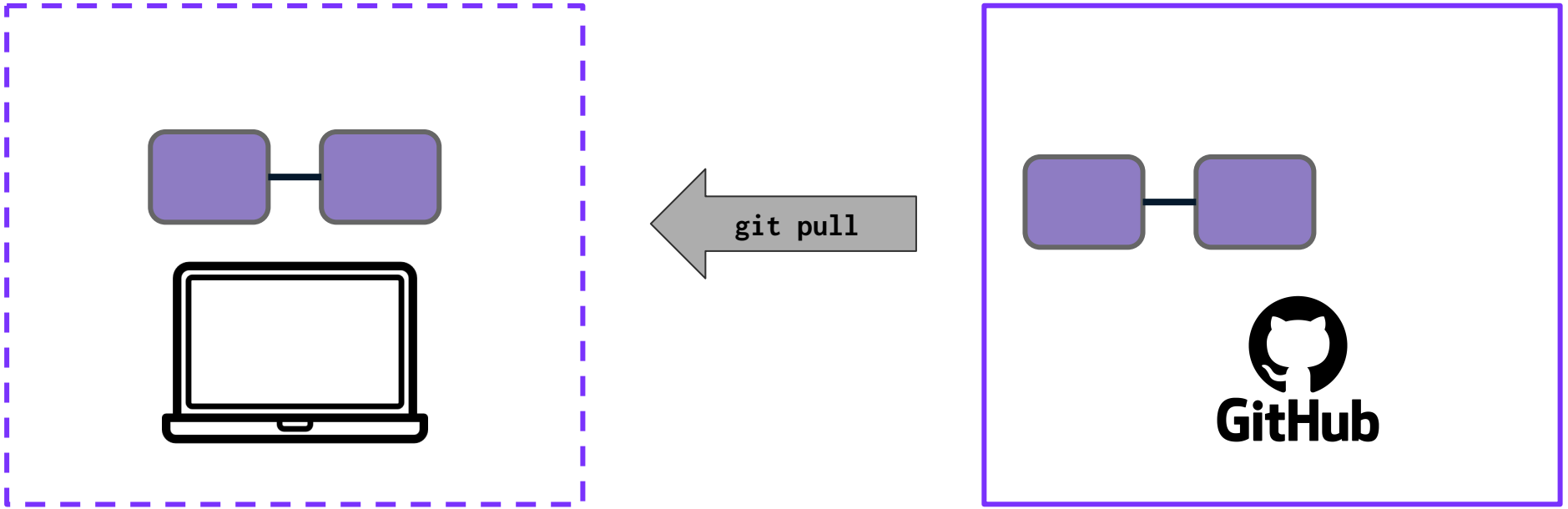
GitHub and Git in Practice

- Single Branch Workflow



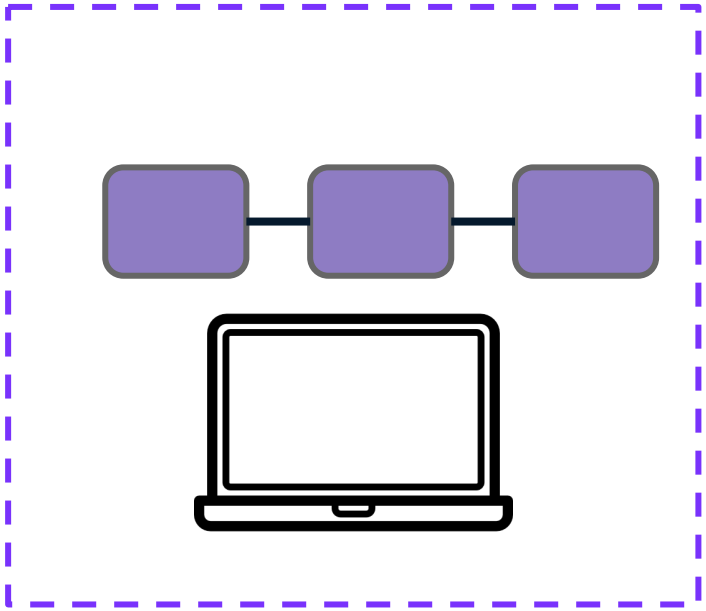
GitHub and Git in Practice

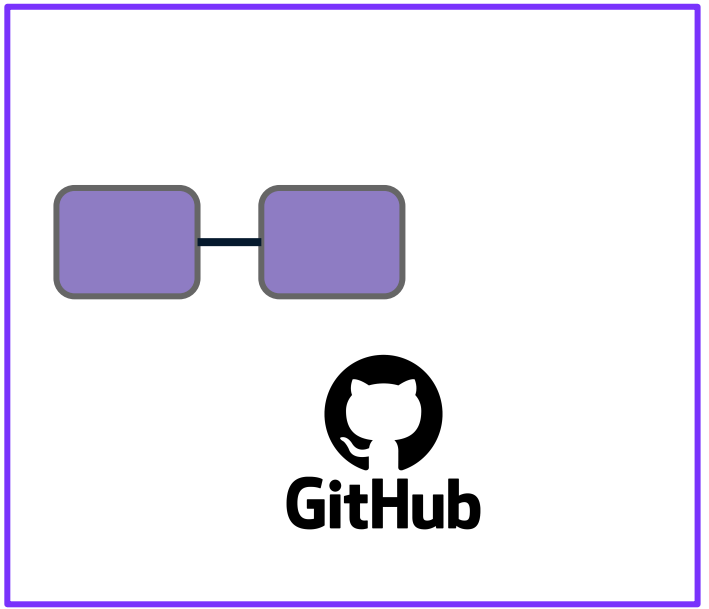
- Single Branch Workflow



GitHub and Git in Practice

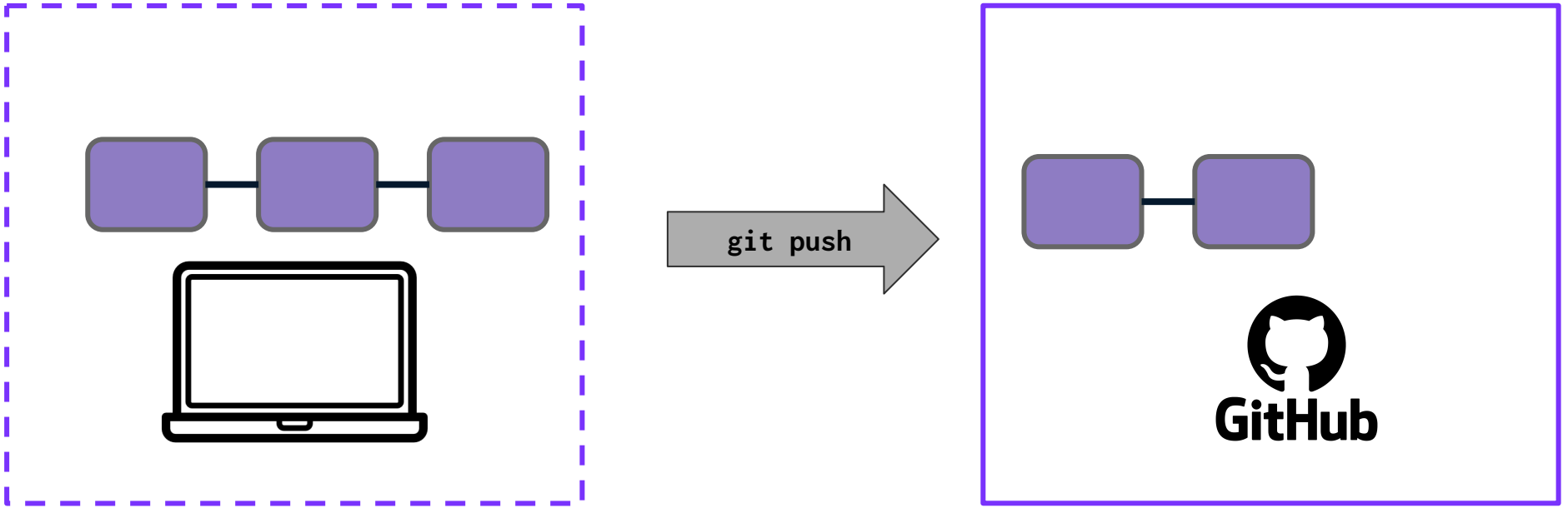
- Single Branch Workflow





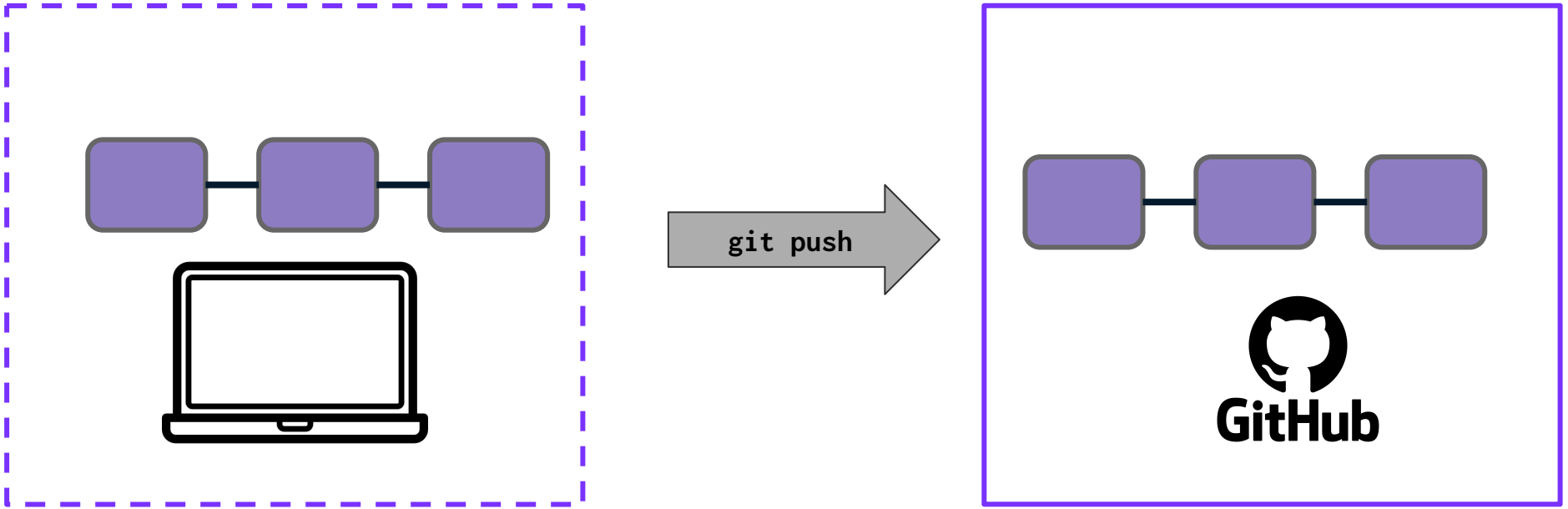
GitHub and Git in Practice

- Single Branch Workflow



GitHub and Git in Practice

- Single Branch Workflow



GitHub and Git in Practice

- **Single Branch Workflow**
  - Pros:
    - Very simple workflow, never need to worry about other branches, conflicts, or merges.
    - Could work well for code systems that don’t require a lot of testing and are non-consumer facing.

GitHub and Git in Practice

- **Single Branch Workflow**
  - Cons:
    - Becomes almost impossible to maintain a proper workflow with team members, everyone would need to run **git pull** and merge the master branch changes before pushing their own commits.

GitHub and Git in Practice

- **Single Branch Workflow**
  - Cons:
    - Even for solo developers, what happens when you want to try out experimental features?
    - You would still need to commit them to the master branch, potentially introducing breaking updates!

GitHub and Git in Practice

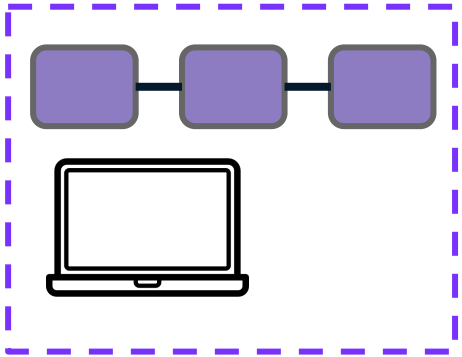
- **Branch Based Development**
  - The master/main branch is never worked on directly.
  - All new development and features are worked on separate branches, and then if collaboration is needed, the branches are also pushed to the remote repository.

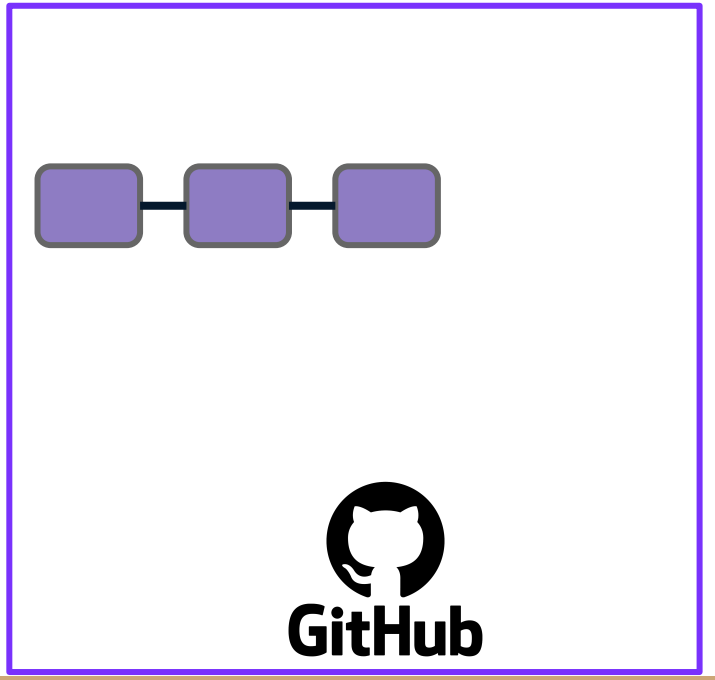
GitHub and Git in Practice

- **Branch Based Development**
  - The master/main branch serves as a “source of truth” and usually has procedures for merging branches back into the main branch, for example creating tests for the new features of the branch and passing historically created tests.

GitHub and Git in Practice

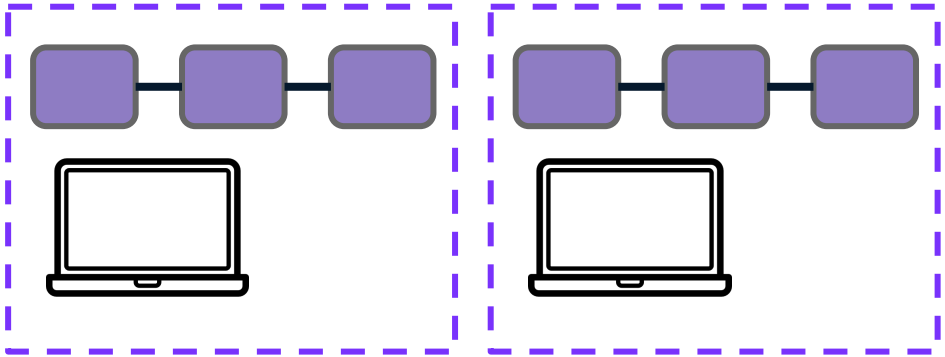
- **Branch Based Development**

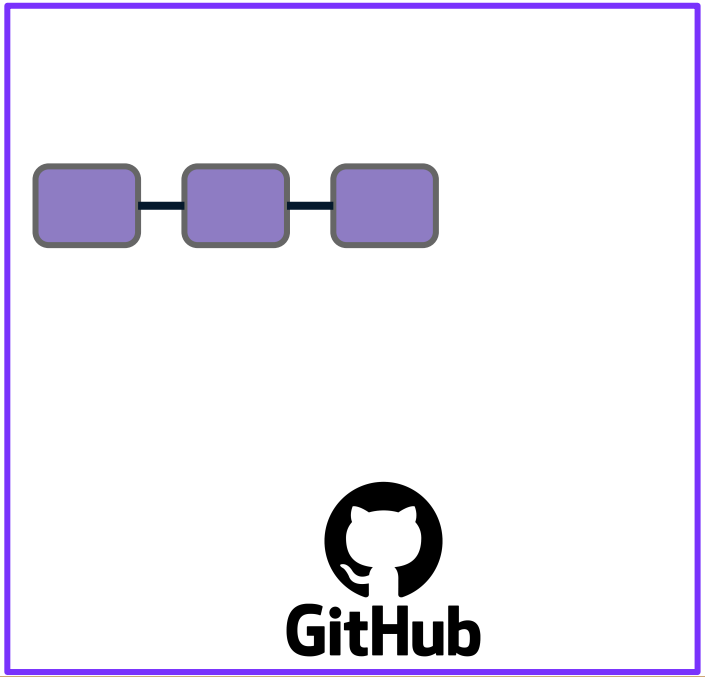




GitHub and Git in Practice

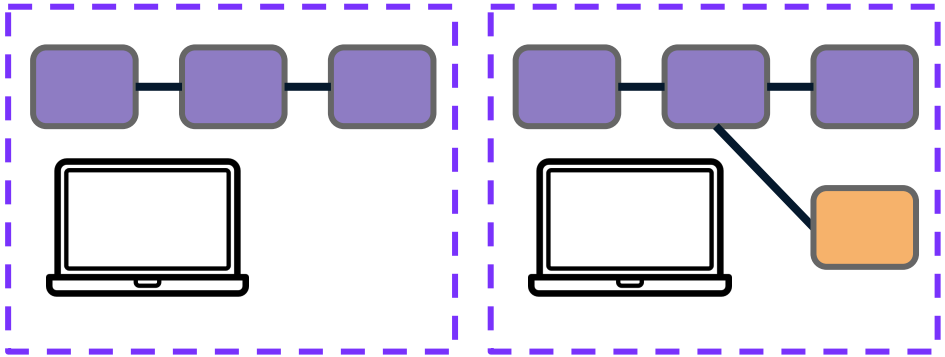
- **Branch Based Development**

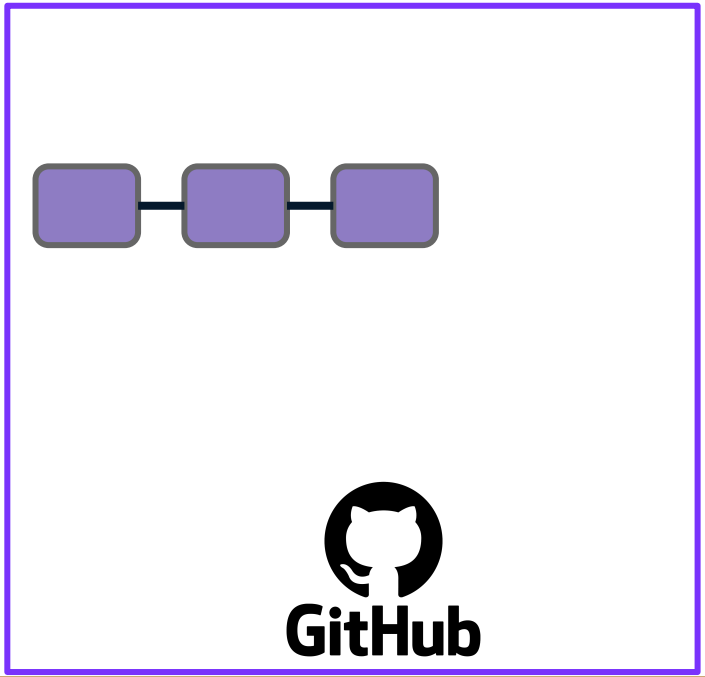




GitHub and Git in Practice

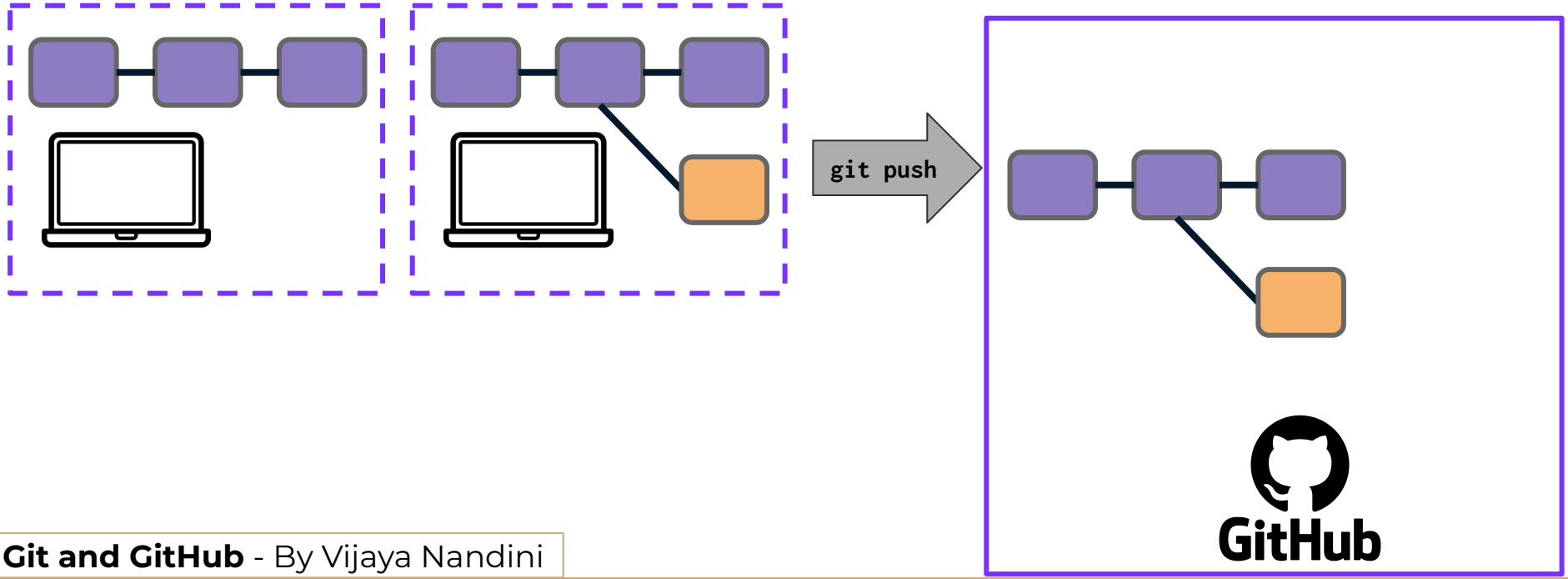
- **Branch Based Development**





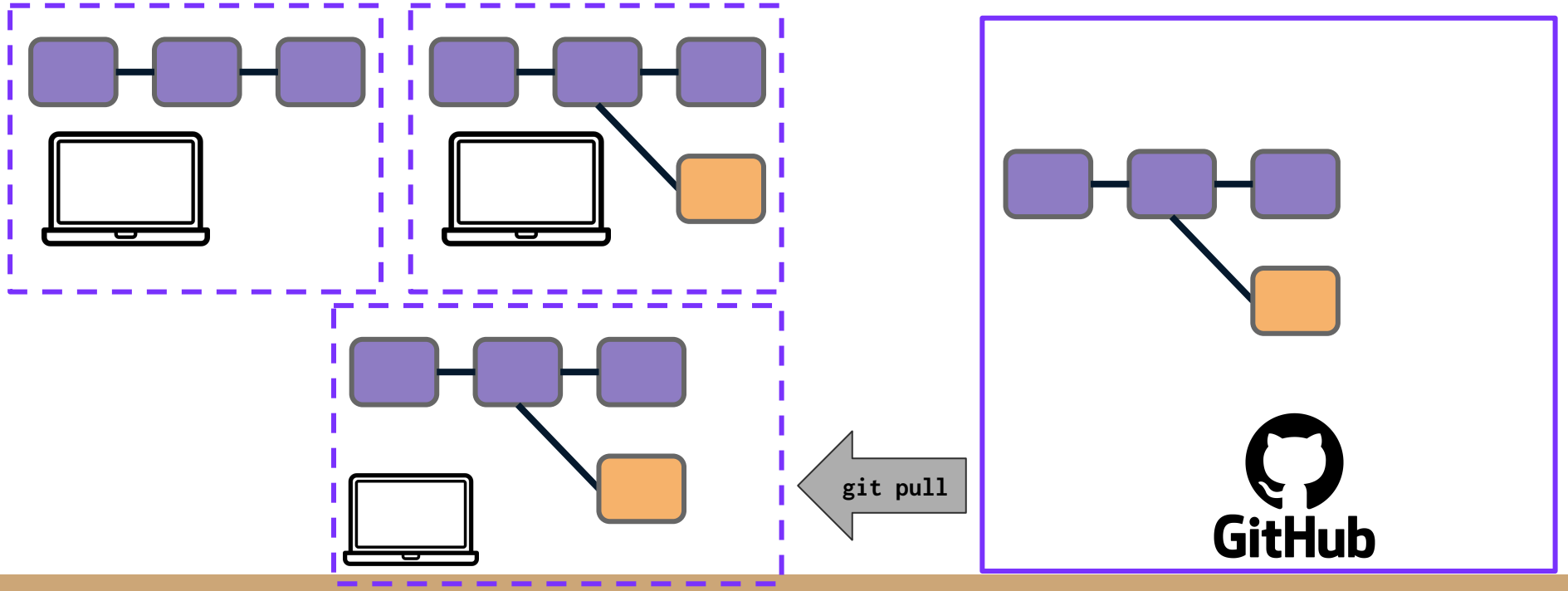
GitHub and Git in Practice

- **Branch Based Development**



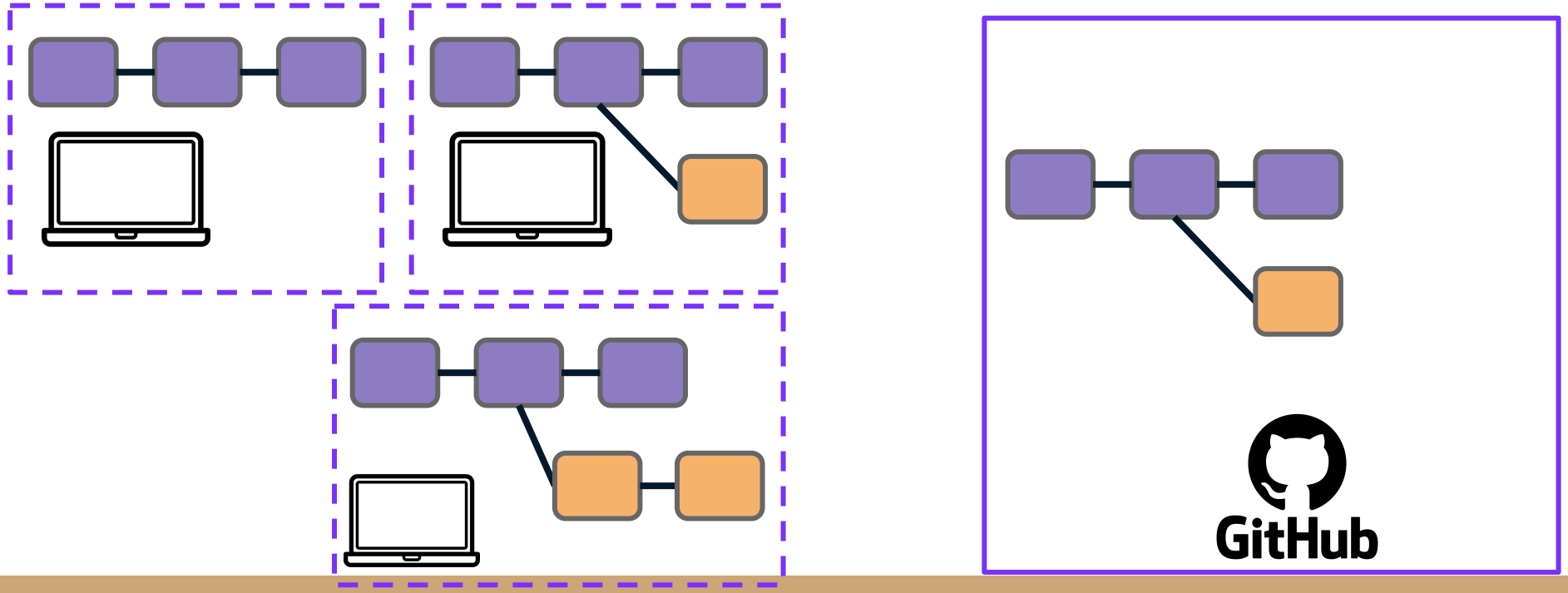
GitHub and Git in Practice

- Branch Based Development



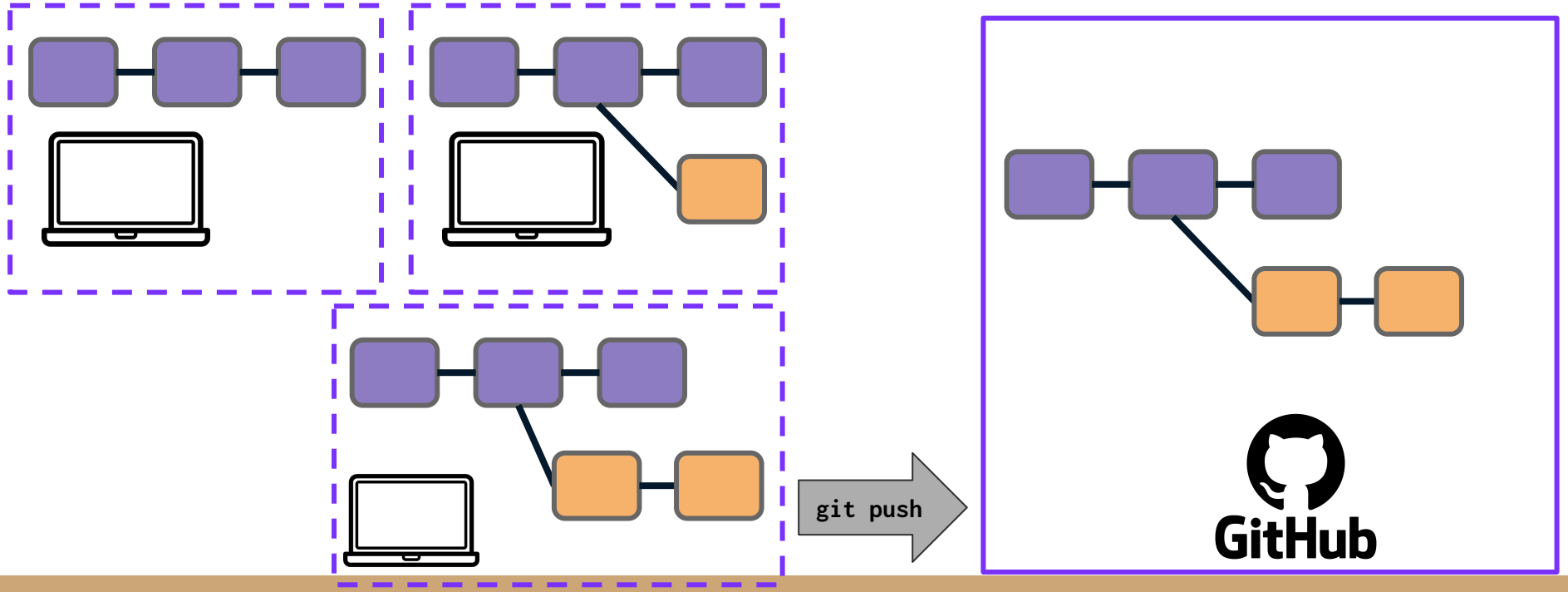
GitHub and Git in Practice

- Branch Based Development



GitHub and Git in Practice

- Branch Based Development



GitHub and Git in Practice

- **Branch Based Development**
  - Pros:
    - Allows for multiple collaborators to work on the same codebase and develop new features together at the same time.
    - Allows for an infinite number of experiments and features (within reasonable limits).

GitHub and Git in Practice

- **Branch Based Development**
  - Cons:
    - Needs a defined process in place to accept merges into the main/master branch.
    - Fortunately, we will later learn about **pull requests** as a way to manage these merges.

GitHub
Pull Request

GitHub and Git in Practice

- **GitHub Pull Requests**
  - Pull requests are a service feature of GitHub and is actually not native to Git (competing platforms also provide pull request features).
  - They allow for an organized review of new merges into a branch.

GitHub and Git in Practice

- **GitHub Pull Requests**
  - Code review processes will vary between organizations, but pull requests are often the foundation of the code review process.
  - GitHub has the ability to give different levels of permissions for approving pull requests or allowing a merge.

Forking on GitHub

GitHub and Git in Practice

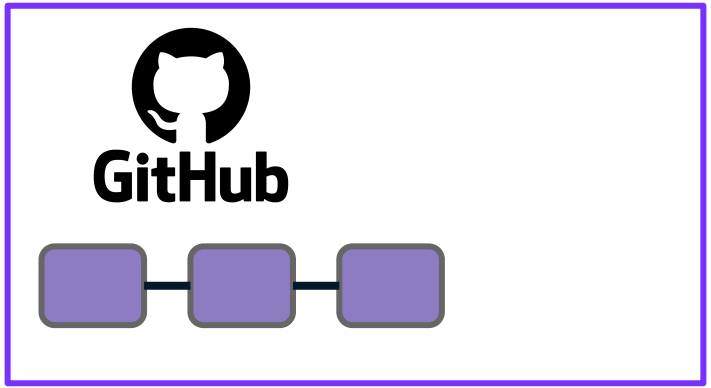
- Now that we understand pull requests, let's explore the workflow of using a "fork" which is a GitHub feature, creating a copy of a repository on GitHub for your own GitHub account.
- Then you can clone that version locally, make changes, and create pull requests to merge those changes.

GitHub and Git in Practice

- Forking only creates the copy on your GitHub, you then need to clone this copy to your local machine, make updates/changes, and then create the pull request.
- Any changes you push to GitHub would only get pushed onto your own copy of the repository.

GitHub and Git in Practice

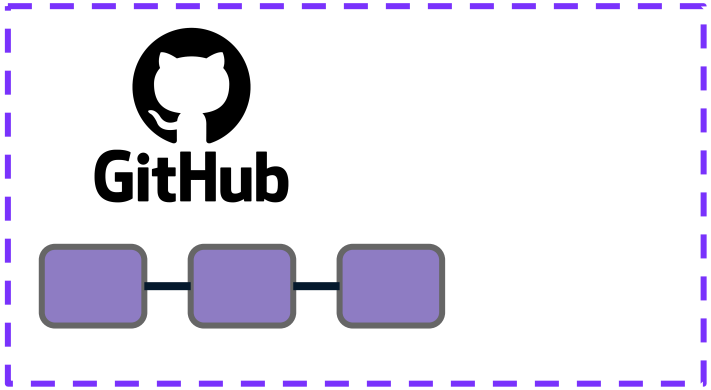
- **Original Repository**



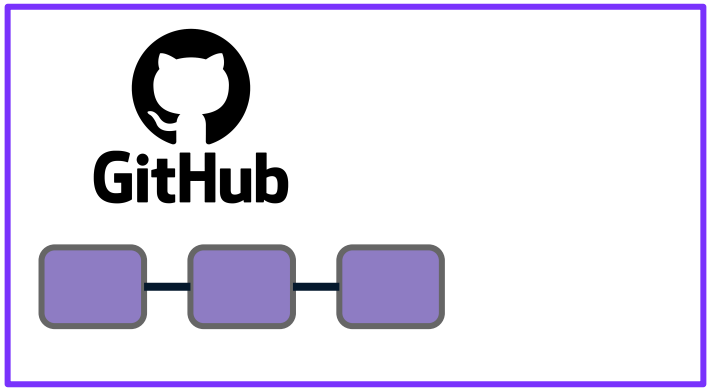
Original Repository

GitHub and Git in Practice

- Fork a copy on GitHub



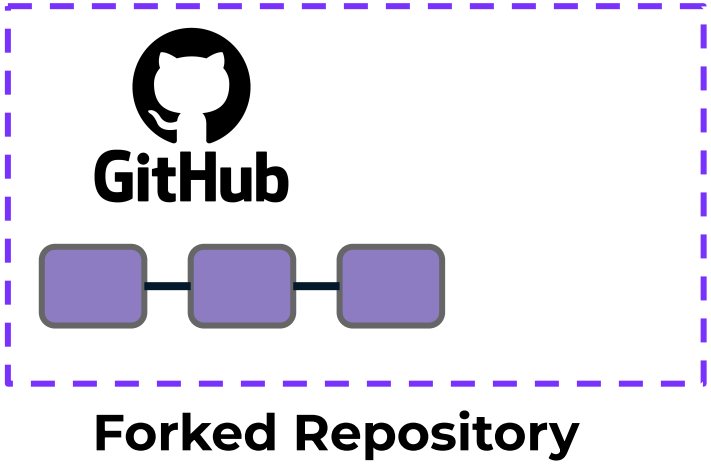
Forked Repository



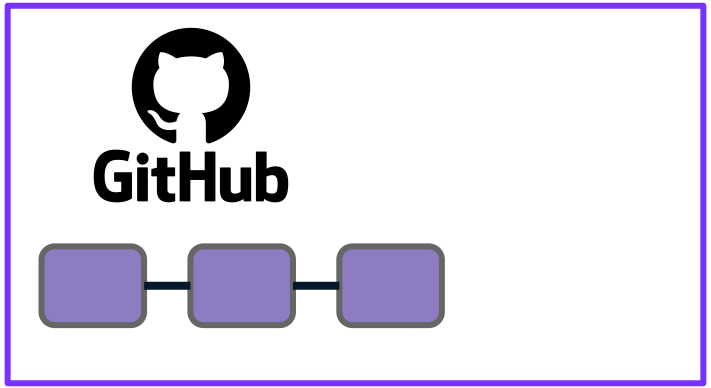
Original Repository

GitHub and Git in Practice

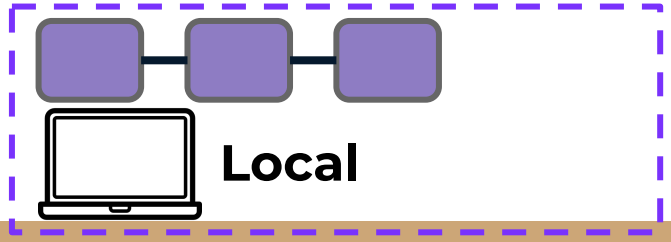
- Clone *your* forked copy locally



Forked Repository

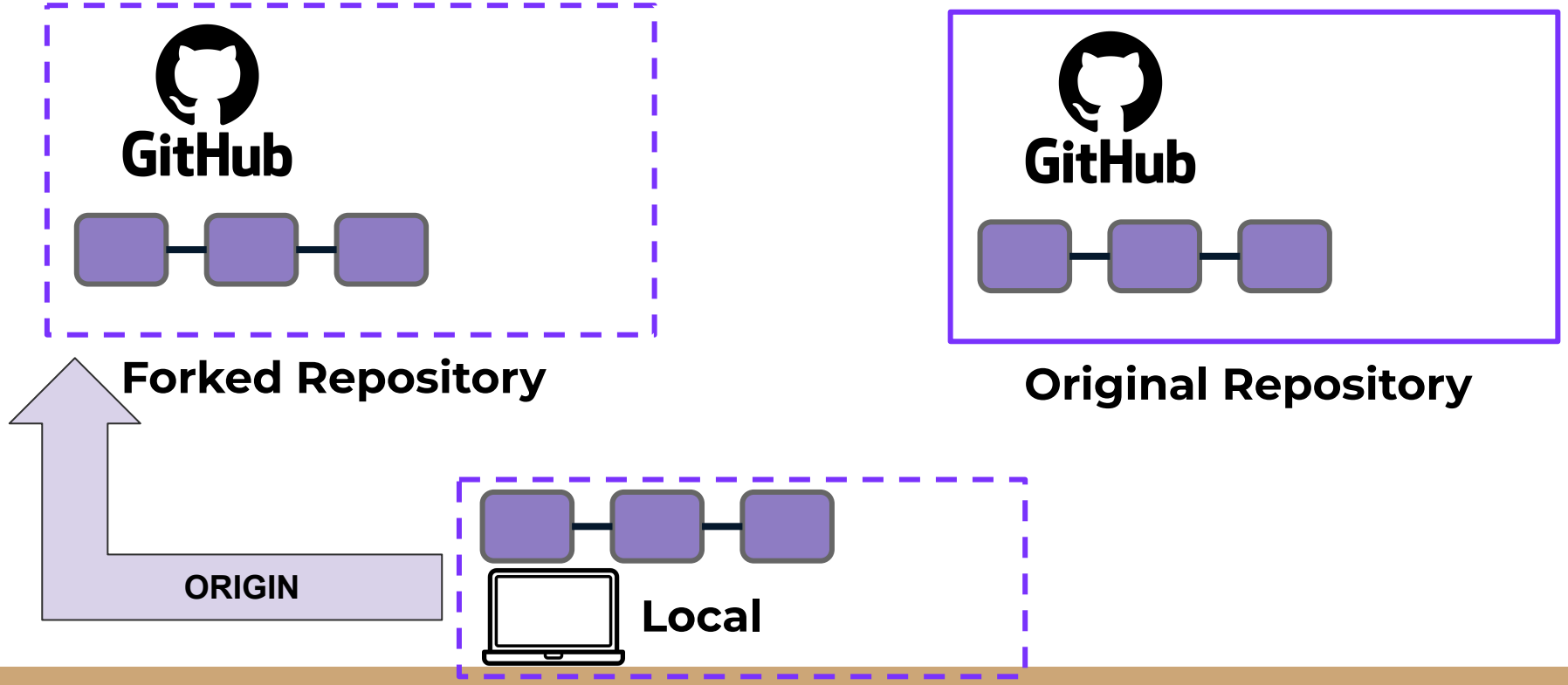


Original Repository



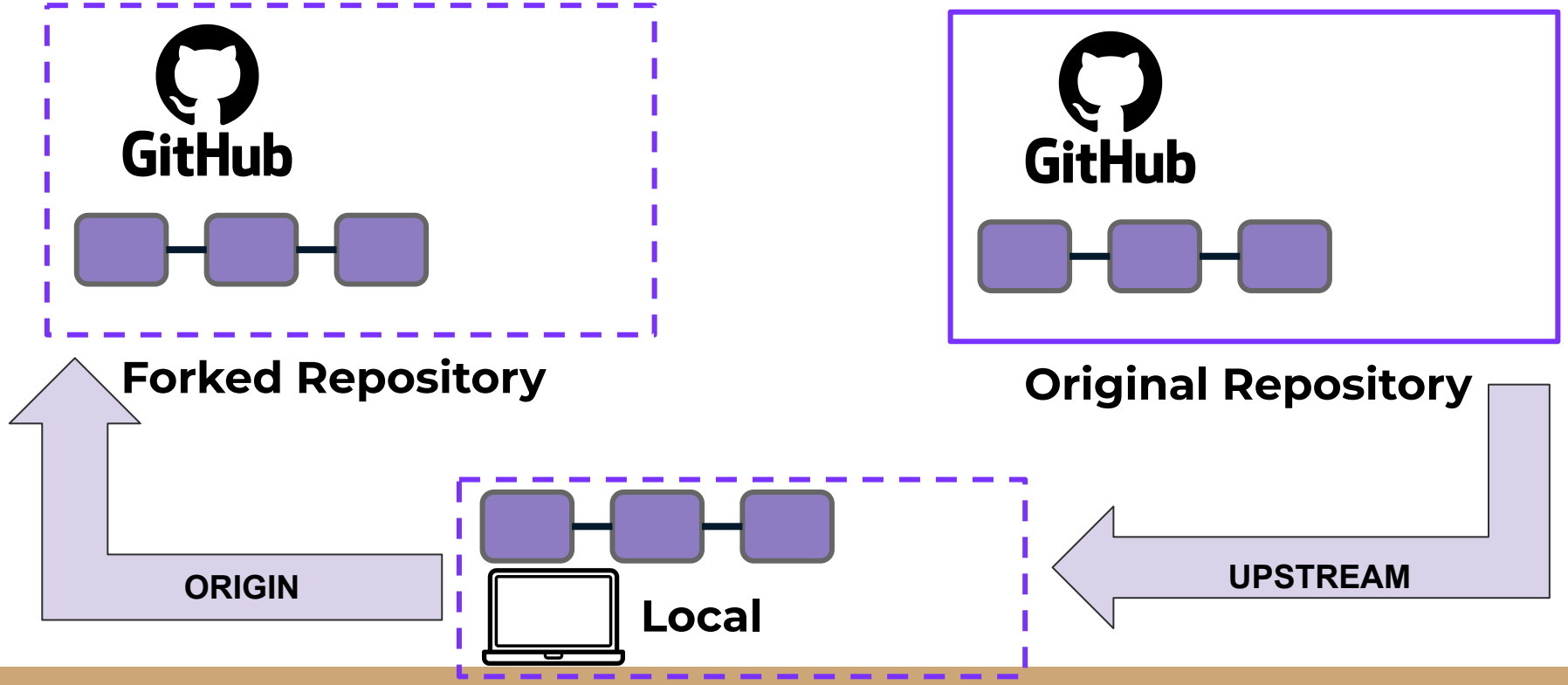
GitHub and Git in Practice

- Set up your remote origin



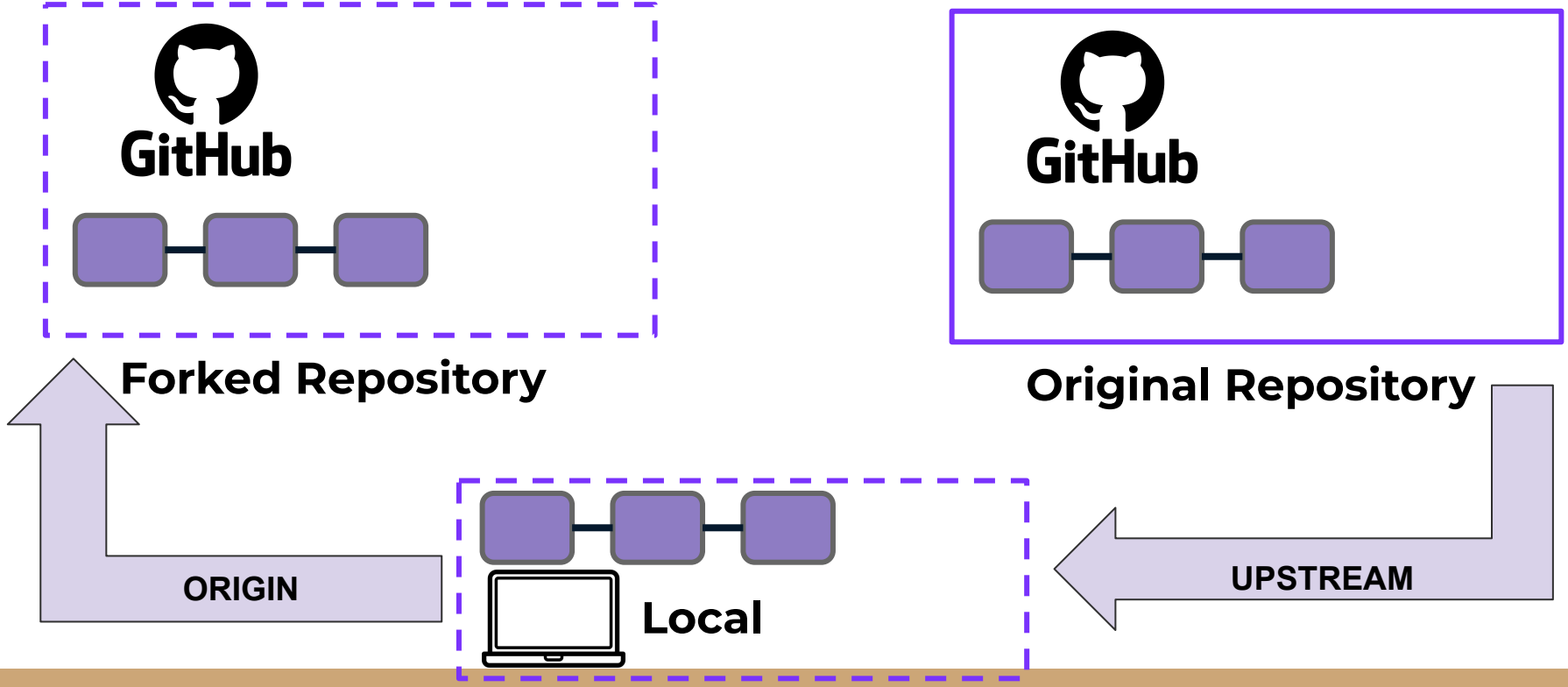
GitHub and Git in Practice

- And set up your remote upstream



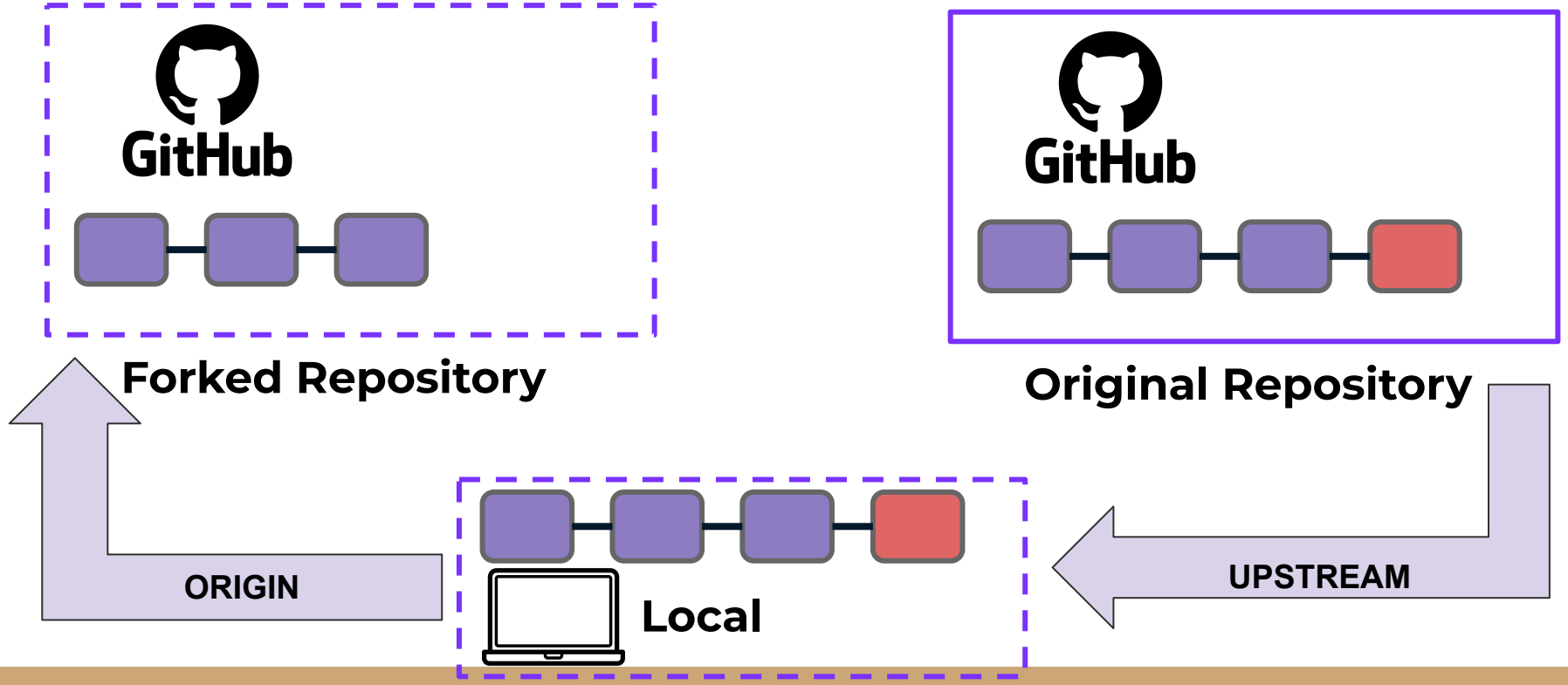
GitHub and Git in Practice

- Upstream allows you to update locally



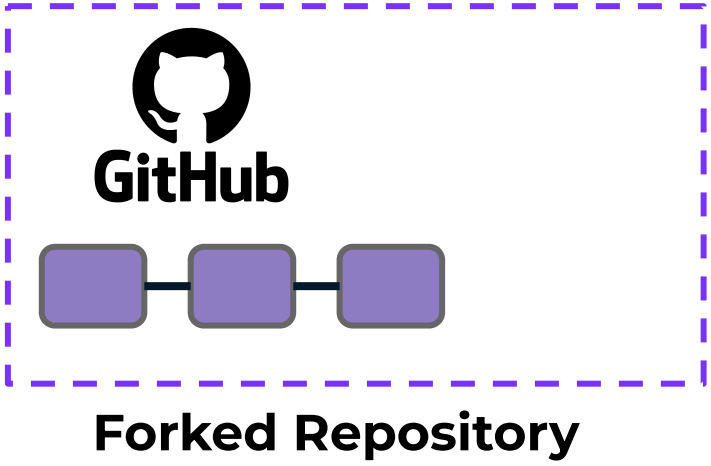
GitHub and Git in Practice

- Update from upstream remote

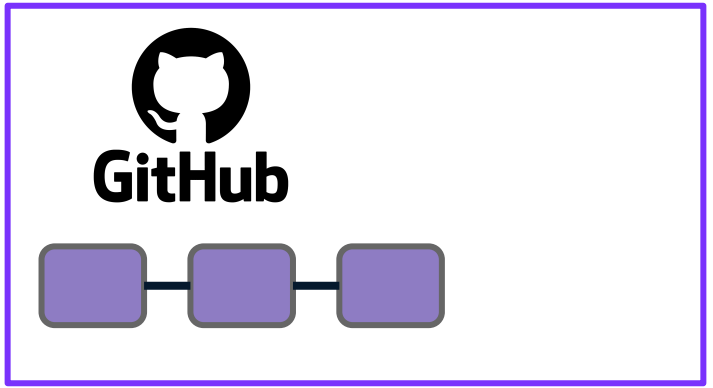


GitHub and Git in Practice

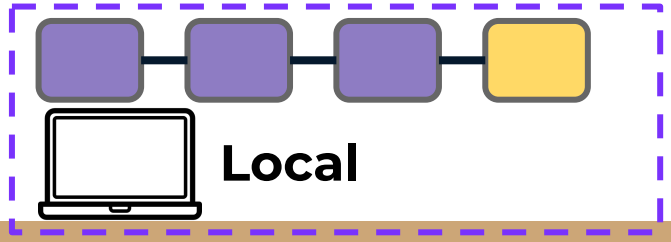
- Now work on changes locally



Forked Repository

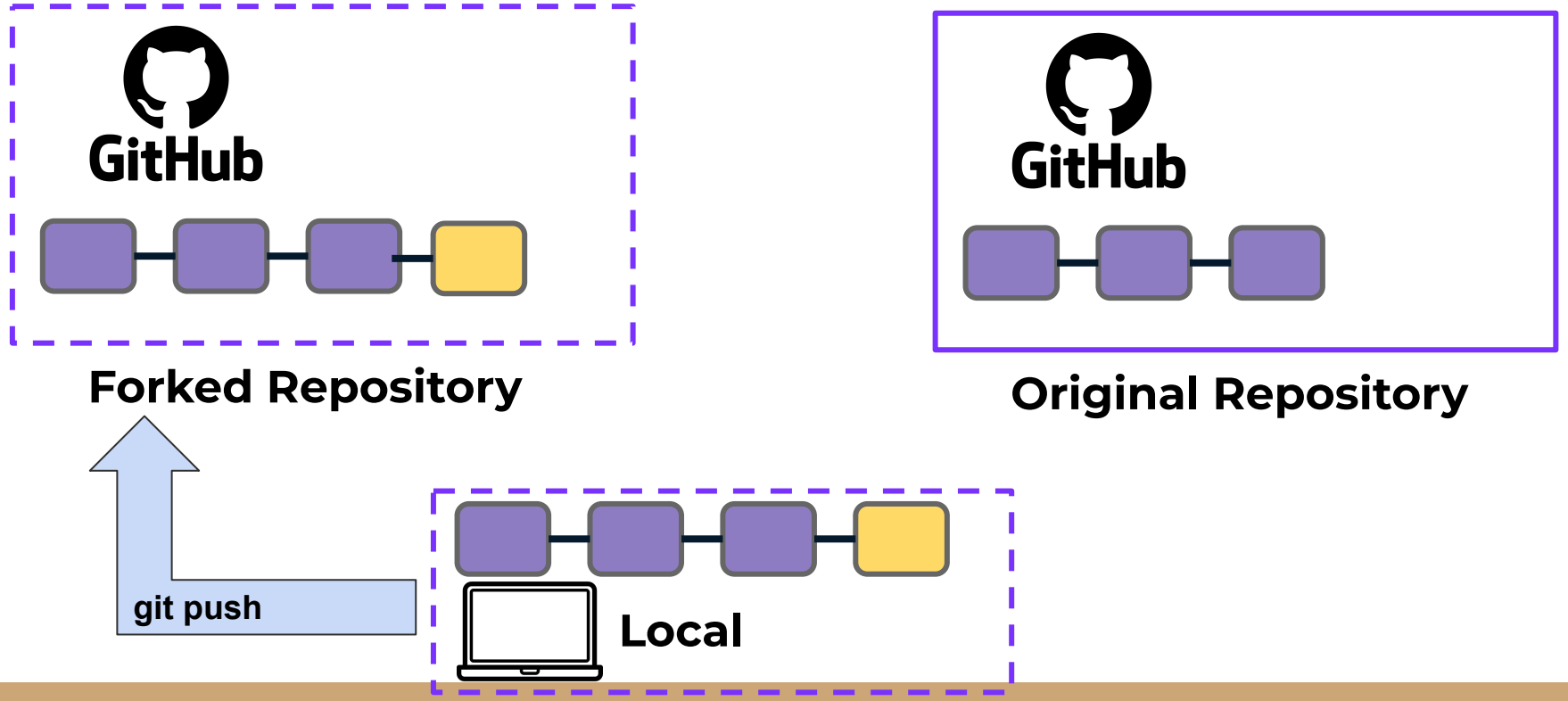


Original Repository



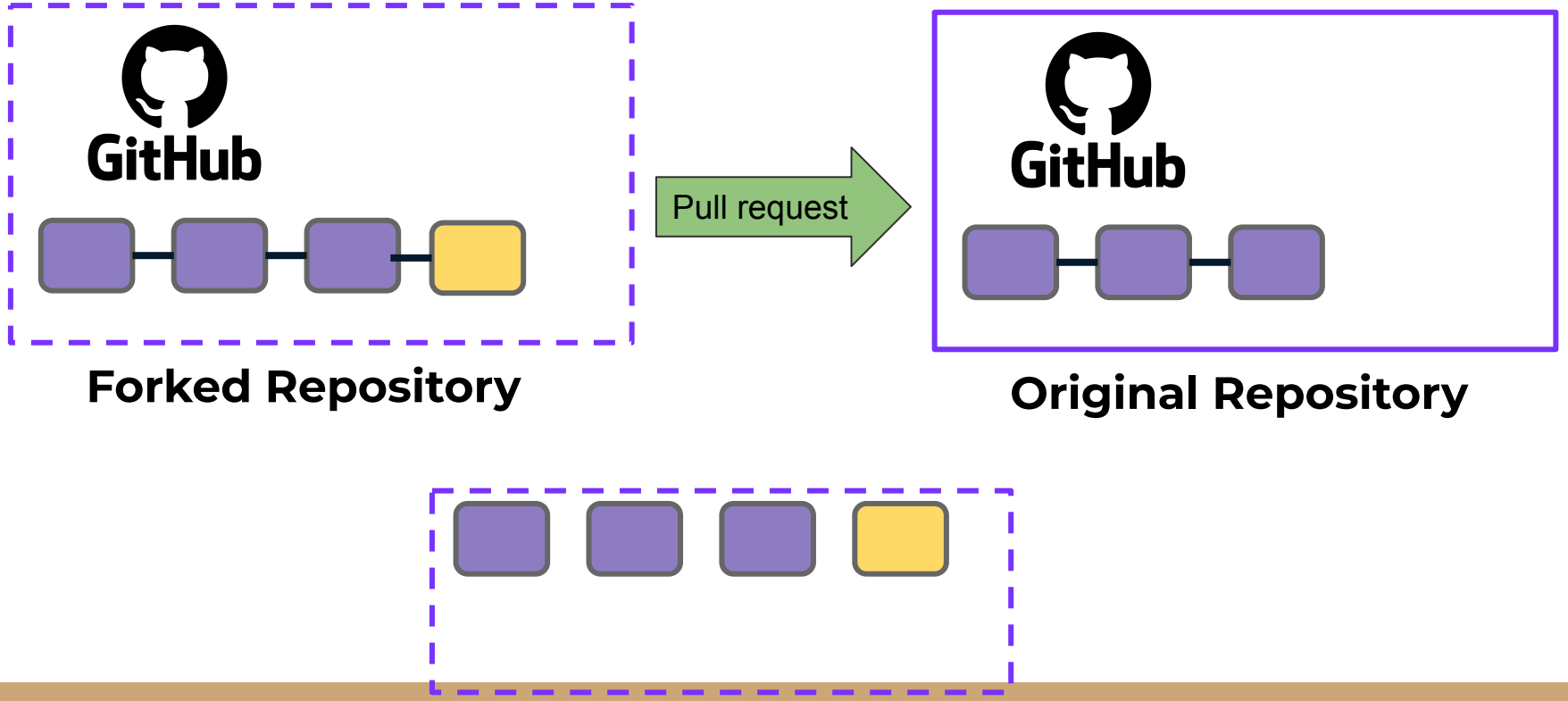
GitHub and Git in Practice

- Push to origin



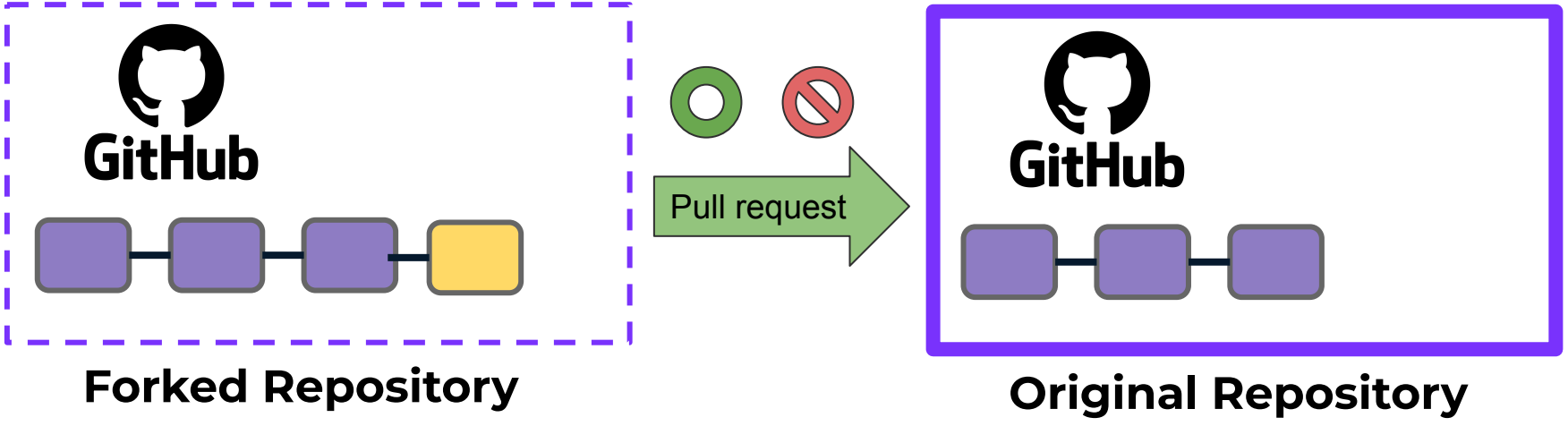
GitHub and Git in Practice

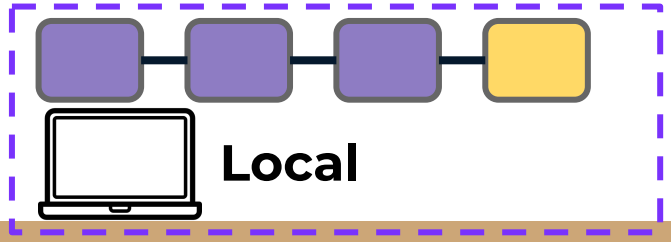
- Submit Pull Request on GitHub



GitHub and Git in Practice

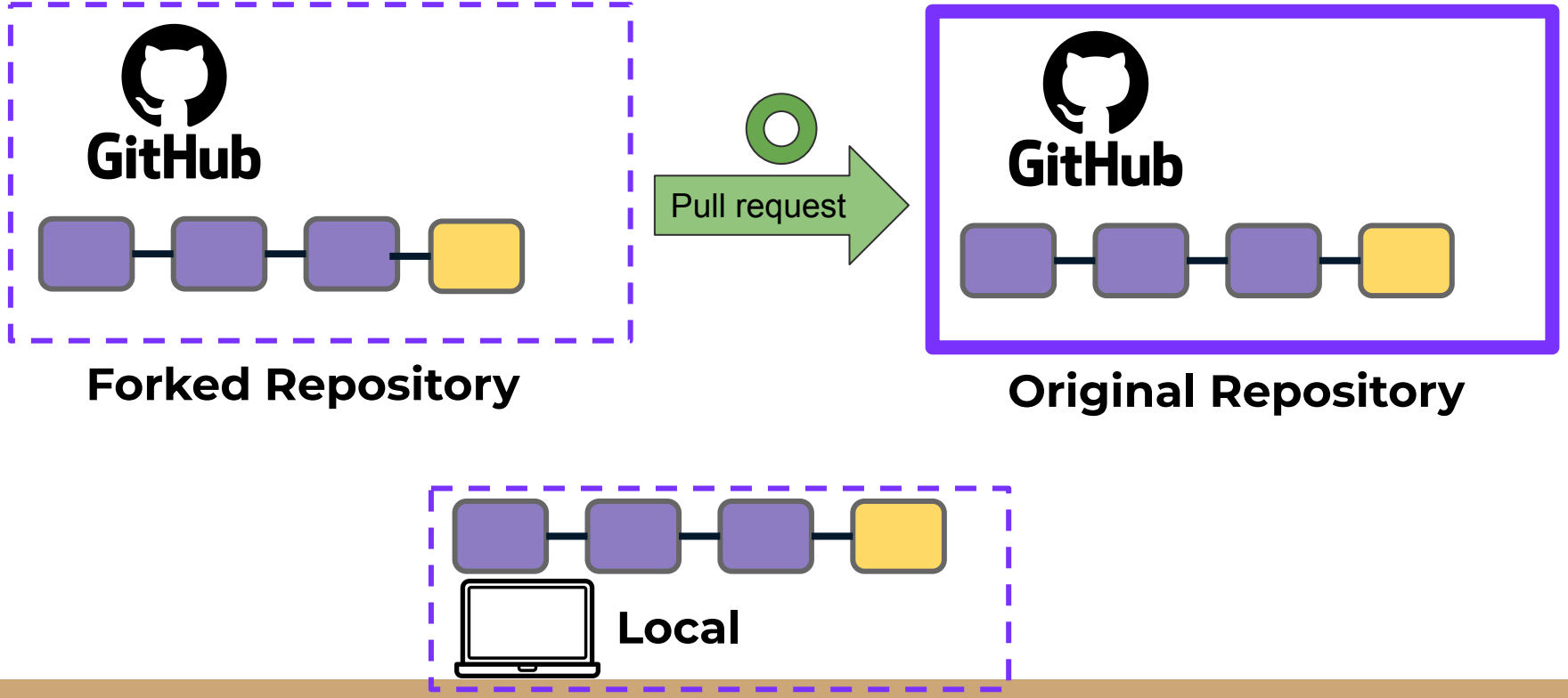
- Owner can decide to merge or not





GitHub and Git in Practice

- Can accept changes



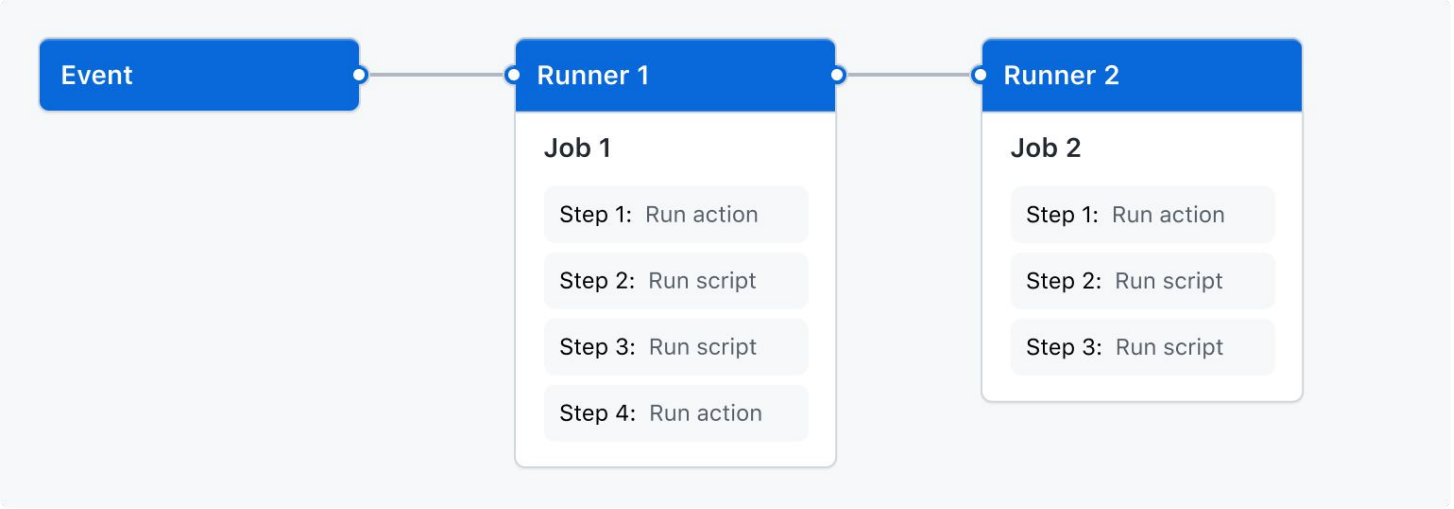
GitHub Actions

GitHub and Git in Practice

- GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline.
- You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

GitHub and Git in Practice

- GitHub Actions are executed upon an event, and the “runners” of jobs can be stacked on top of each other:



GitHub and Git in Practice

- If you work on large projects with many users on GitHub, you'll often see GitHub actions in use to help maintain the codebase.
- The actions themselves are YAML files with instructions for what GitHub should do upon an event.

GitHub and Git in Practice

- GitHub Actions YAML File

YAML

```yaml
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{ github.event_
      - run: echo "🐧 This job is now running on a ${{ runner.os }} server hoste
      - run: echo "🔎 The name of your branch is ${{ github.ref }} and your repo
      - name: Check out repository code
        uses: actions/checkout@v3
      - run: echo "💡 The ${{ github.repository }} repository has been cloned to
      - run: echo "🖥️ The workflow is now ready to test your code on the runner.
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🍏 This job's status is ${{ job.status }}."
```

GitHub and Git in Practice

- You can see the full syntax for the YAML files in the documentation:
  - **https://docs.github.com/en/actions**

Exercise and Solution

GitHub and Git in Practice

- Find an open source project
- Fork and Clone it and then attempt to improve the documentation (or code if you’re up to it!)
- Submit your pull request, and mention you are new to this in your PR!