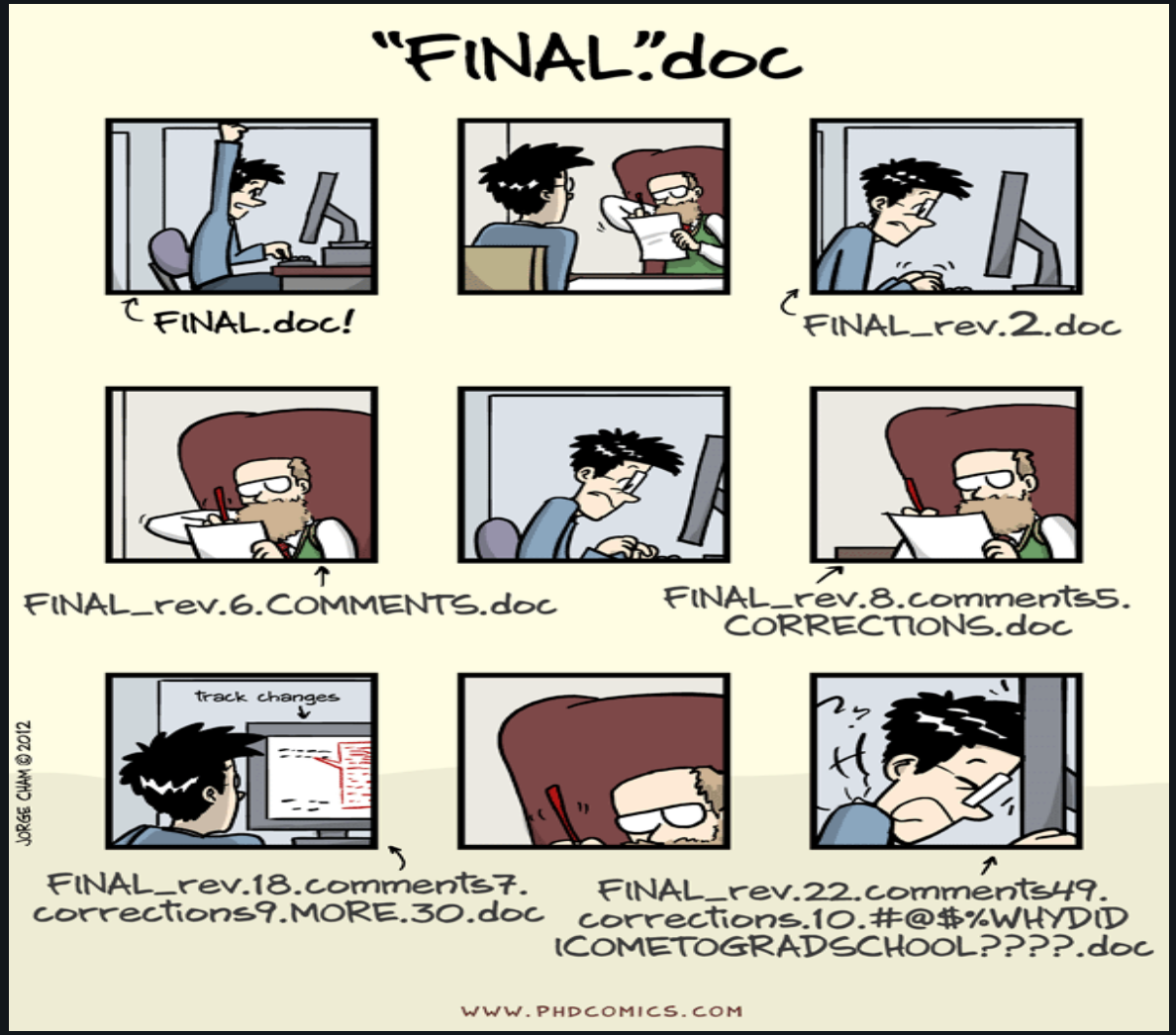
Git & GitHub

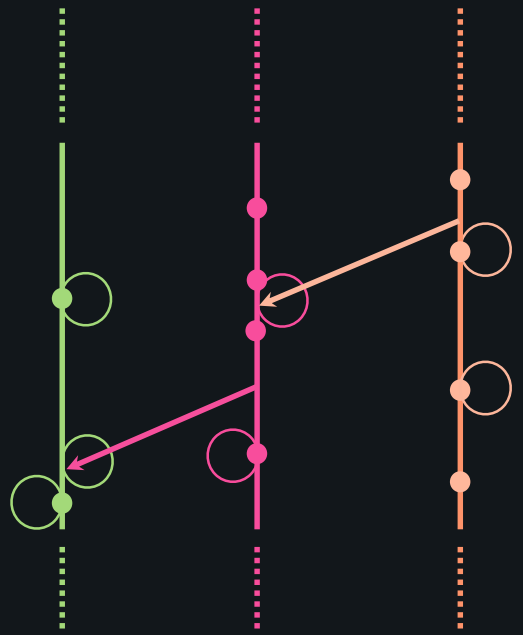
General Overview


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
JORGE CHAM © 2012
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
WWW.PHDCOMICS.COM

What is Git?

Git is a **version control system** for tracking changes in code.

Git is an **open source**, distributed version control system designed for speed and efficiency.



What is GitHub?

GitHub is a web-based platform that uses Git for

1. hosting repositories
2. enabling collaboration
3. issue tracking
4. project management.

Note: It's widely used for open-source development and team collaboration.

Working with Commits

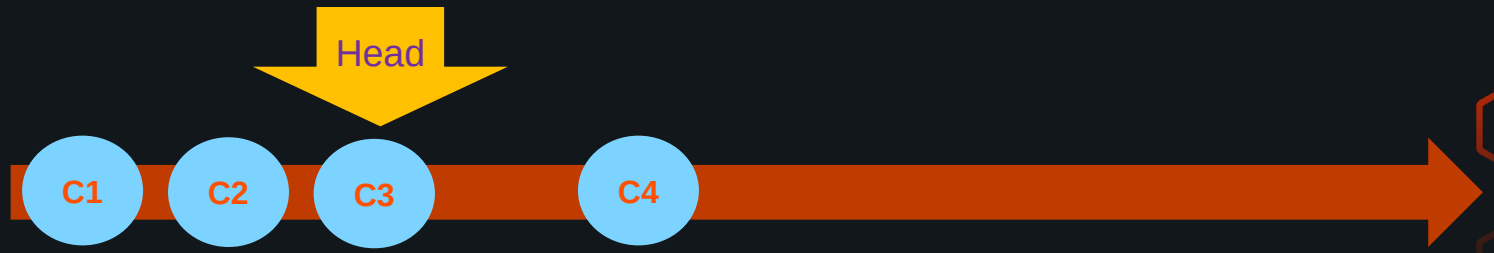
Creating Commits

1 – **Staging**

- git add <file(s)> : Stage changes for next commit.

2 – **Committing**

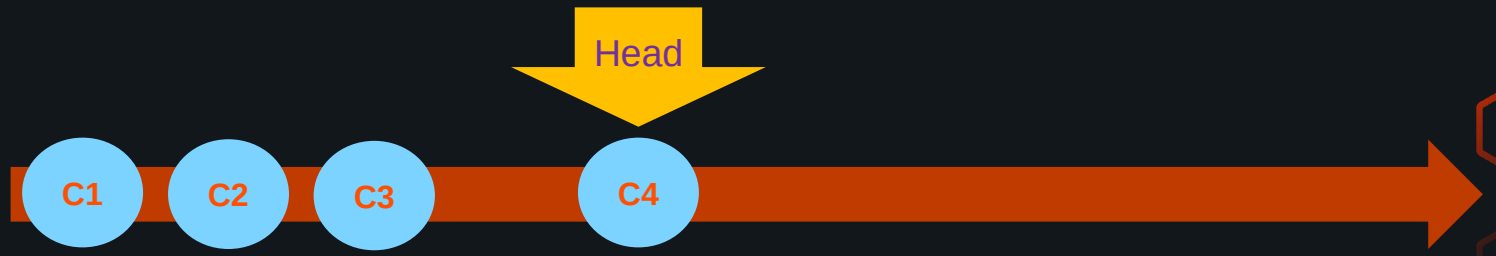
- git commit –m “message” : Create a commit that includes all staged changes



Working with Commits

Moving between commits

- git checkout <id>: Temporarily move to another commit.

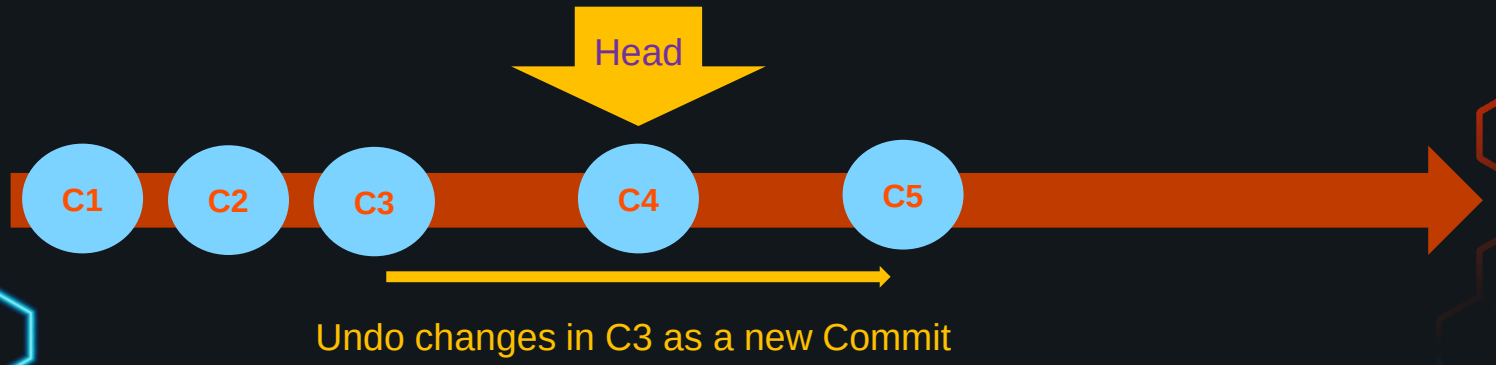


Working with Commits

Undo Commits

1 – **Reverting**

- git revert <id> : Revert changes of commit by creating a new commit.



Working with Commits

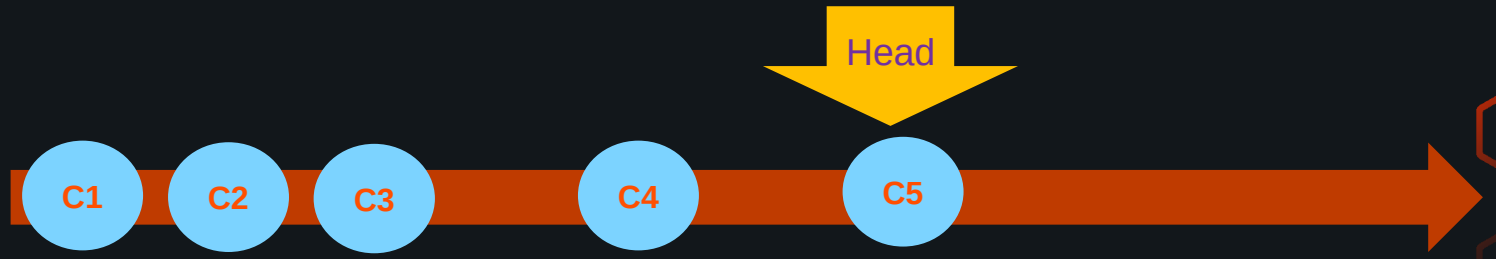
Undo Commits

1 – **Reverting**

- git revert <id> : Revert changes of commit by creating a new commit

2 – **Deleting Permittly**

- git reset --hard <id>: Undo changes by deleting all commits since <id>



Common Commands

git init	Initialize a Git repository (only required once per project)
git add < file(s) >	Stage code changes (for the next commit)
git commit –m "…"	Create a commit for the staged changes (with a message)
git status	Get the current repository status (e.g., which changes are staged)
git log	Output a chronologically ordered list of commits
git checkout <id>	Temporarily move back to commit <id>
git revert <id>	Revert the changes of commit <id> (by creating a new commit)
git reset <id>	Undo commit(s) up to commit <id> by deleting commits

Branching

What is a branch?

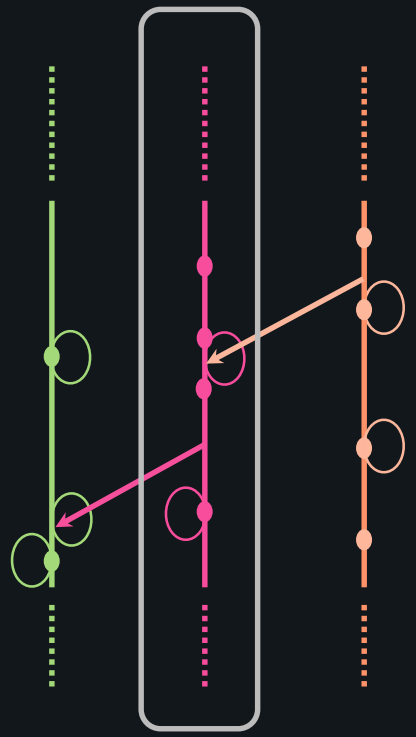
In Git, a branch is a separate line of development that allows you to work on a specific set of changes without affecting the **main – also called Development -** or other branches.

- Each branch represents an **independent snapshot** of the project's codebase, allowing developers to work on different **features** or **bug fixes** concurrently

What is a branch?

Main/Branching:

Master/Main Branch: The default branch in Git is often called "master" or "main." It represents the latest stable state of the project.



What is a branch?

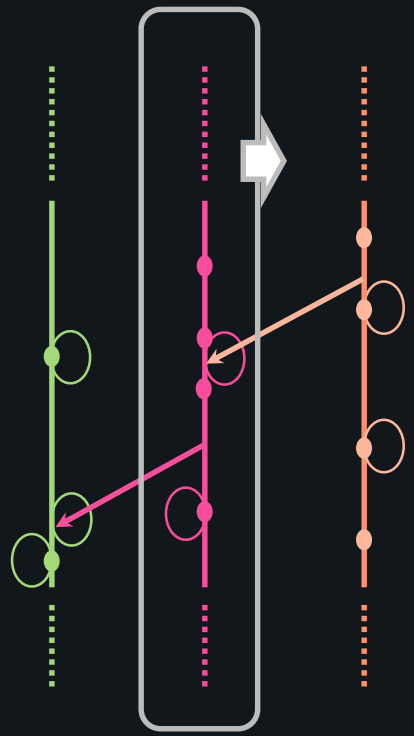
Creating Branches:

Developers create branches to work on new features or bug fixes. The primary branch is typically branched off to create a new branch.

- git branch new-feature

Alternatively, you can create and switch to a new branch in one command.

- git checkout -b new-feature

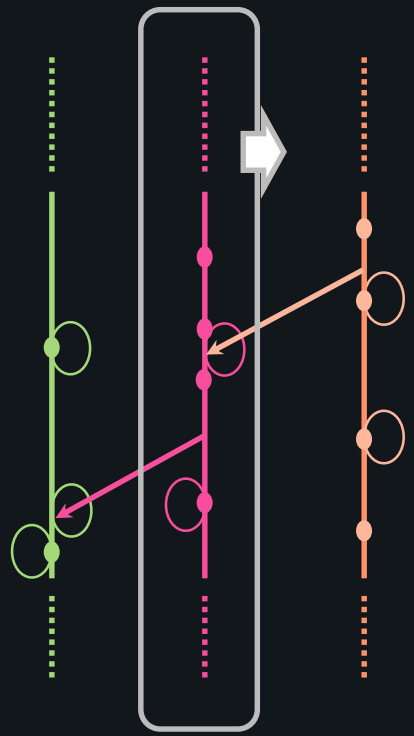


What is a branch?

Switching Between Branches:

Use the git checkout command to switch between branches:

- git checkout new-feature
- git switch new-feature

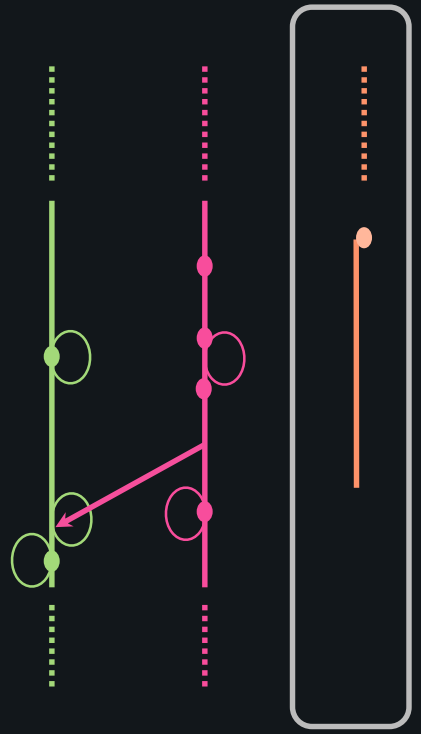


What is a branch?

Merging Branches:

Once a feature or bug fix is complete, the changes from the branch are merged back into the main branch

- git checkout **main**
-



What is a branch?

Merging Branches:

Once a feature or bug fix is complete, the changes from the branch are merged back into the main branch

- git checkout **main**
- git merge **new-feature**

GitHub Collaboration

1- As Collaborators

The Repo owner adds people are collaborators to their projects, so that they can clone the repo, and add their features.

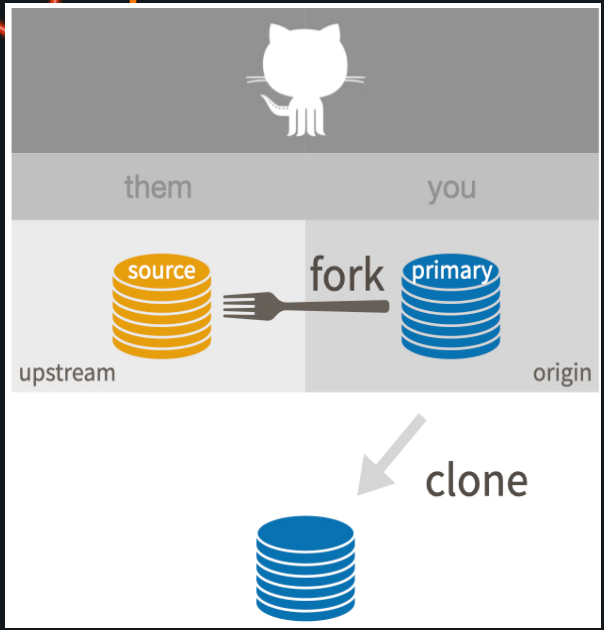
Then they Submit a pull request, to merge there changes to the main Branch - which is mostly protected by the owner using protection rule, to keep code as bug free as possible.

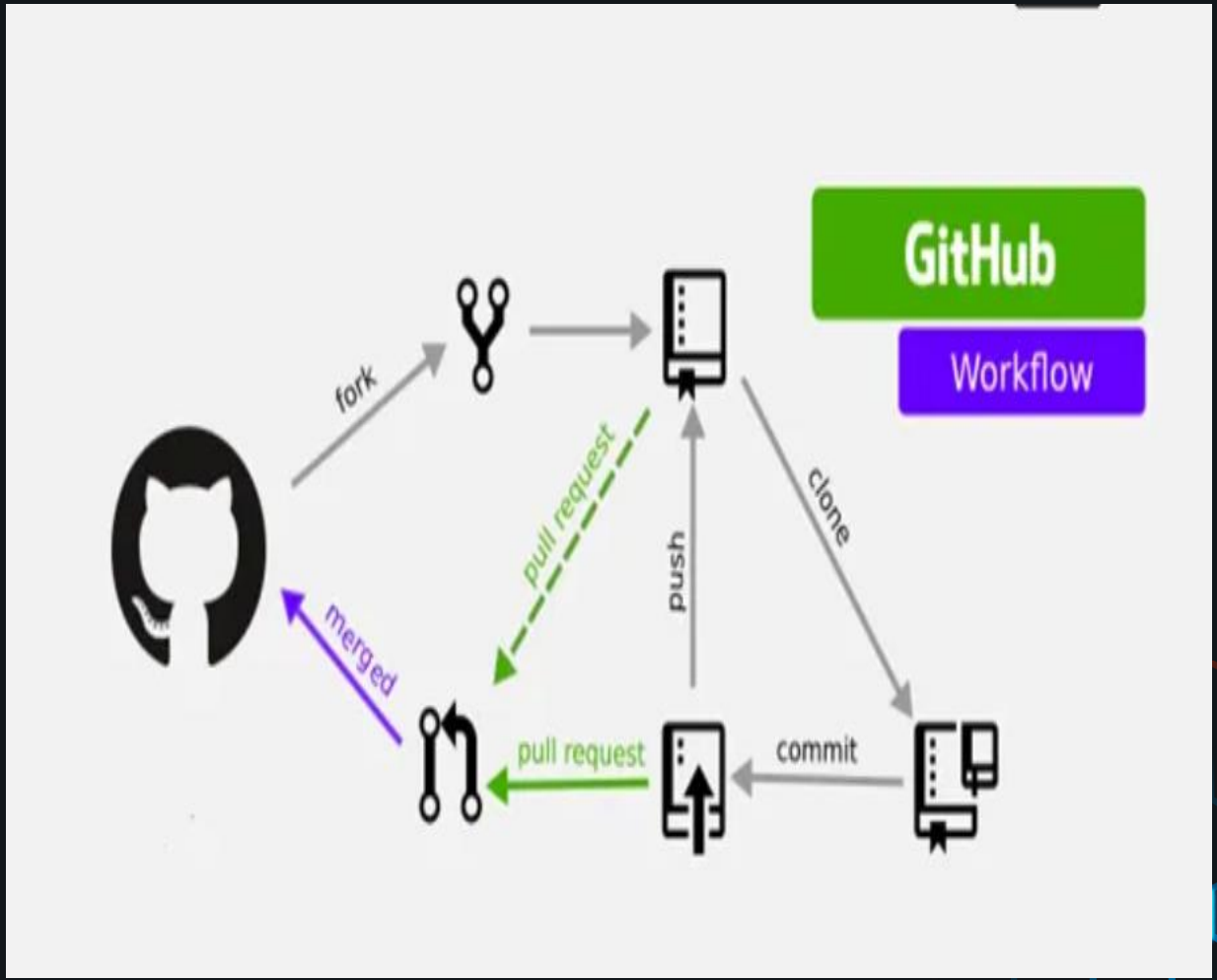
GitHub Collaboration

1- Forking

It is not practical to add everyone as a collaborator, instead, the repo can be forked and features are added to the forked repo – which is basically mine == I am the owner

Then they Submit a pull request, to merge there changes to the main Branch across the forks


them
you
source
fork
primary
upstream
origin
clone


GitHub
Workflow
fork
pull request
push
clone
merged
pull request
commit

Forks and Pull Requests

How To Fork a repository?

1. Go to the Target Repository
2. Press Fork
3. Create A Fork
4. Clone the Repo To Your Device

5. Make Our Changes/Features (**in main or in a new branch**)
6. Push The Changes

7. Compare Forks -> Make a Pull Request

GO TO NOTION

How we will work with GitHub

How we will work

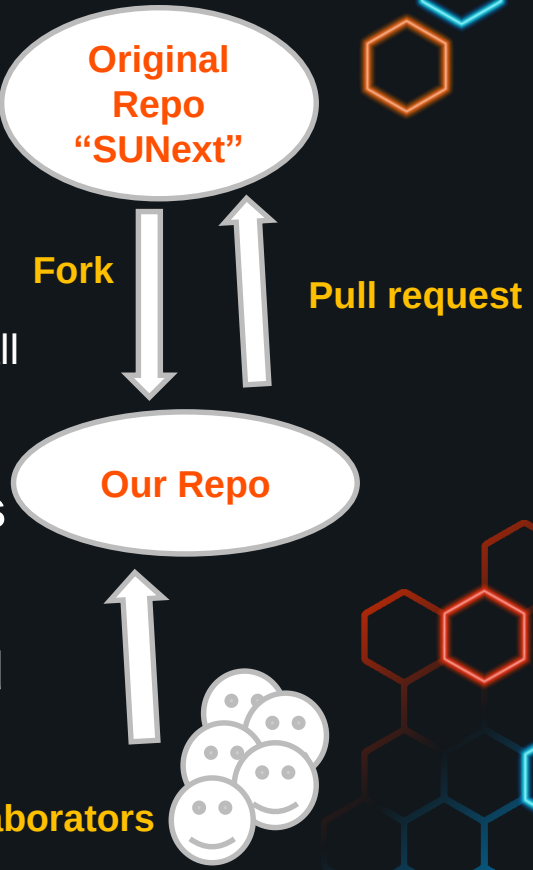
External Work Flow

1 – **Fork the SUNext Repo**

- The team leader will be the repo owner.
- He will also keep the repo **up to date** by pulling all the new changes in the Original Repo.

2 – The Team Members will be added as Collaborators.

3 – After Code “Features” is Tested and reviewed, the owner will Submit a pull request.



How we will work

Internal Work Flow

1 – **We will create Branches For Each Feature**

- Basic doctypes, Setup, Constraints Setup, Reports, AI integration, Manual Edit (Vue Drag & Drop) … etc.

2 – We will work in Micro Teams (2-3), for developing certain Features.

- The Remaining Members will Test and Review The Work, and Vis-versa.

3 – We will Use Wikis to document our Screens

- **But Can WE?**

What is a Wiki?

rfearing / gotta-spark-em-all

Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

Wiki

Home

Ricardo Fearing edited this page on Jul 14 · 2 revisions

Edit | New Page

SparkBox Pokedex

This project was first created as part of the interview process at Sparkbox and has continued to be utilized as an example project for Wiki CircleCI implementation.

Developer Documentation
NPM Packages & Tools Used
Coding Decisions & Standards
Building & Running the App
How to Contribute

Pages 5

Find a Page...

- Home
- How To Contribute
- Packages
- Running The App
- Standards

+ Add a custom sidebar

Clone this wiki locally

https://github.com/rfea

Can We Use Wikis to document Our Work?

NO,

Because the repo won't be public

But the alternative is better

We will use NOTION

What is Notion?

Notion is a powerful all-in-one productivity tool with many uses such as:

1. Note-taking and Documentation
2. Task and Project Management
3. Knowledge Base and Wiki
4. Personal Organization
5. Team Collaboration
6. Knowledge Management
7. Content Creation and Publishing
8. Personal CRM and Contact Management

GO TO NOTION

Thanks!

Moving on to Agile Scrum