Raj Saha
cloudwithraj.com
Cloud With Raj
cloudwithraj
linkedin.com/in/rajdeep-sa-at-aws/

Instructor Bio:
Sr. Solutions Architect @ aws
Published Udemy/Pluralsight author
Public speaker
Author of multiple AWS official blogs
Previously - Distinguished Cloud Architect @Verizon

Opinions are my own

Git Fundamentals

- ✓ Git What and Why
- ✓ Git vs GitHub
- ✓ Git Workflow
- ✓ Comparing Files
- ✓ Git Branch and Merge
- ✓ Clone, Remote Branches
- ✓ Fetch, Pull
- ✓ Master or Main
- ✓ Dealing with GitHub Branches
- ✓ Browsing History and Commits
- ✓ Browsing History with Git Graph
- ✓ Removing Files

Real World Git with GitHub

- ✓ Pull Request
- ✓ Fork
- ✓ Real World GitHub Workflow
- ✓ Git Pull vs Pull Request
- ✓ Keeping Forked Repository Updated
- ✓ Merge Conflicts
- ✓ Ignoring Files
- ✓ Markdown Guide
- ✓ GitHub Issues
- ✓ GitHub Issues for Open Source
- ✓ GitHub Webhook vs API
- ✓ GitHub Discord Integration
- ✓ GitHub Jenkins Integration
- ✓ Git Branching Strategies – Trunk vs Git Flow

Git Continued

- ✓ Roll Back Changes with Git Revert
- ✓ Roll Back Changes with Git Reset
- ✓ Rebase and Rebase vs Merge
- ✓ Squash Merge
- ✓ Reordering Commits
- ✓ Cherry Picking
- ✓ Git Stash

Git and GitHub Interview Prep

- ✓ GitHub Tips to get Noticed by Recruiters
- ✓ 5 GitHub Showcase Projects for Jobs
- ✓ How to Find Good Open-Source Projects
- ✓ Git Best Practices for Interview
- ✓ Basic – Interview Q/A
- ✓ Intermediate – Interview Q/A
- ✓ Advanced – Interview Q/A
- ✓ Git Commands Cheat Sheet

Git – What and Why

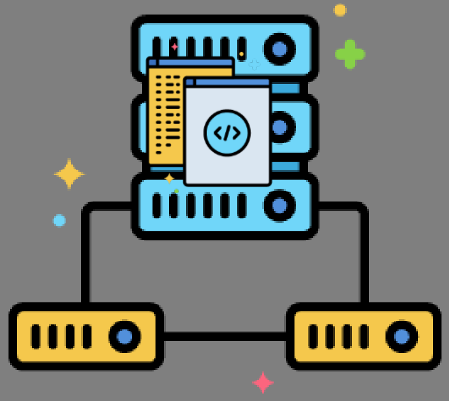
Raj Saha
cloudwithraj.com
Cloud With Raj

Before Version Control System

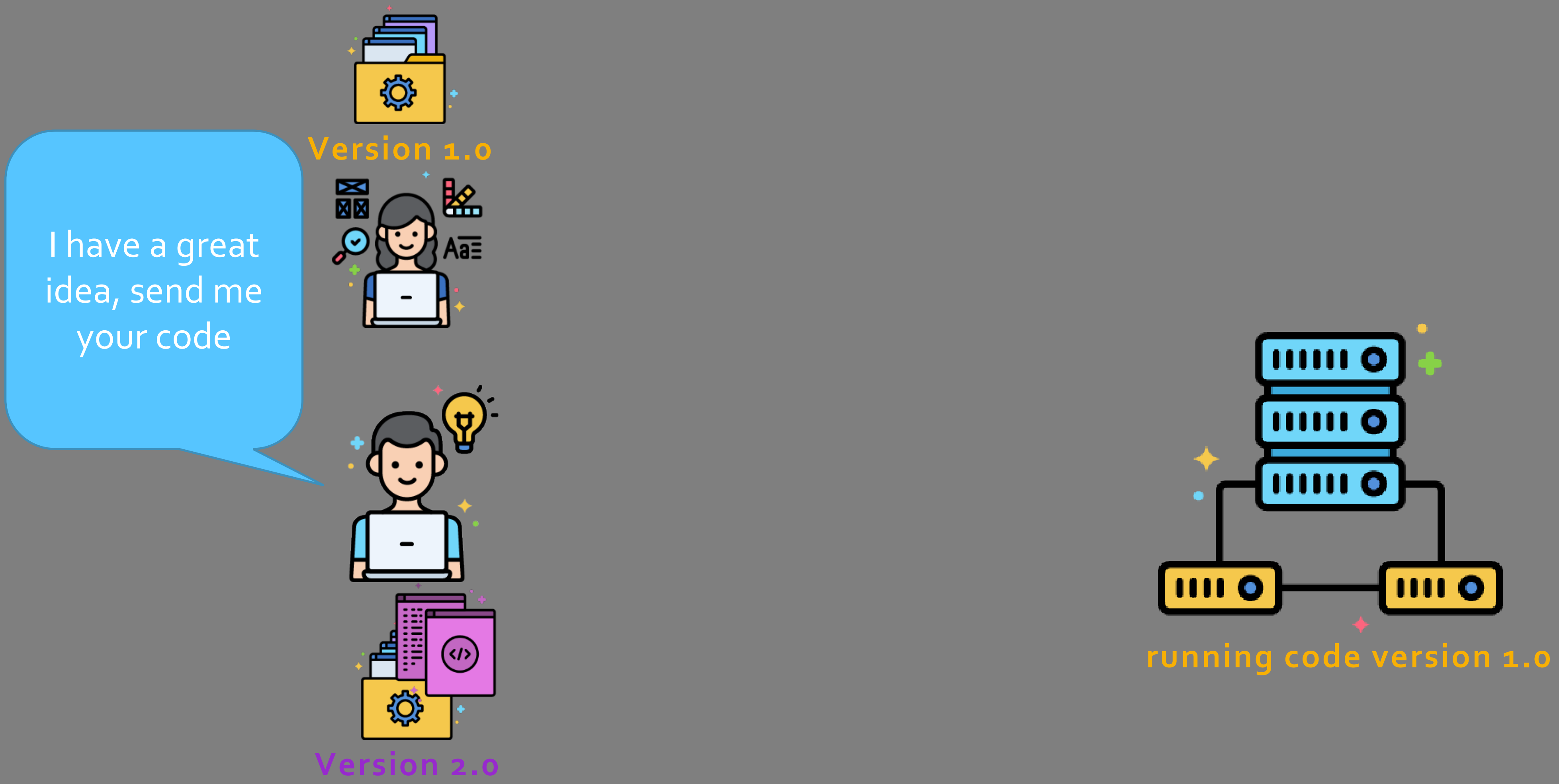
Before Version Control System

Version 1.0

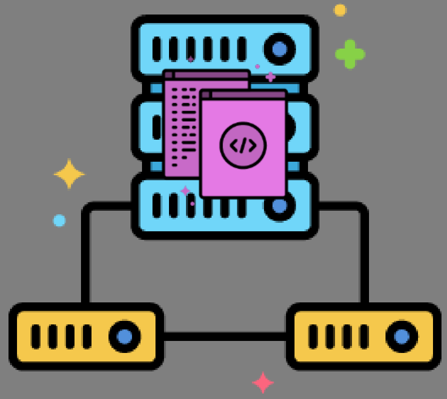
Before Version Control System



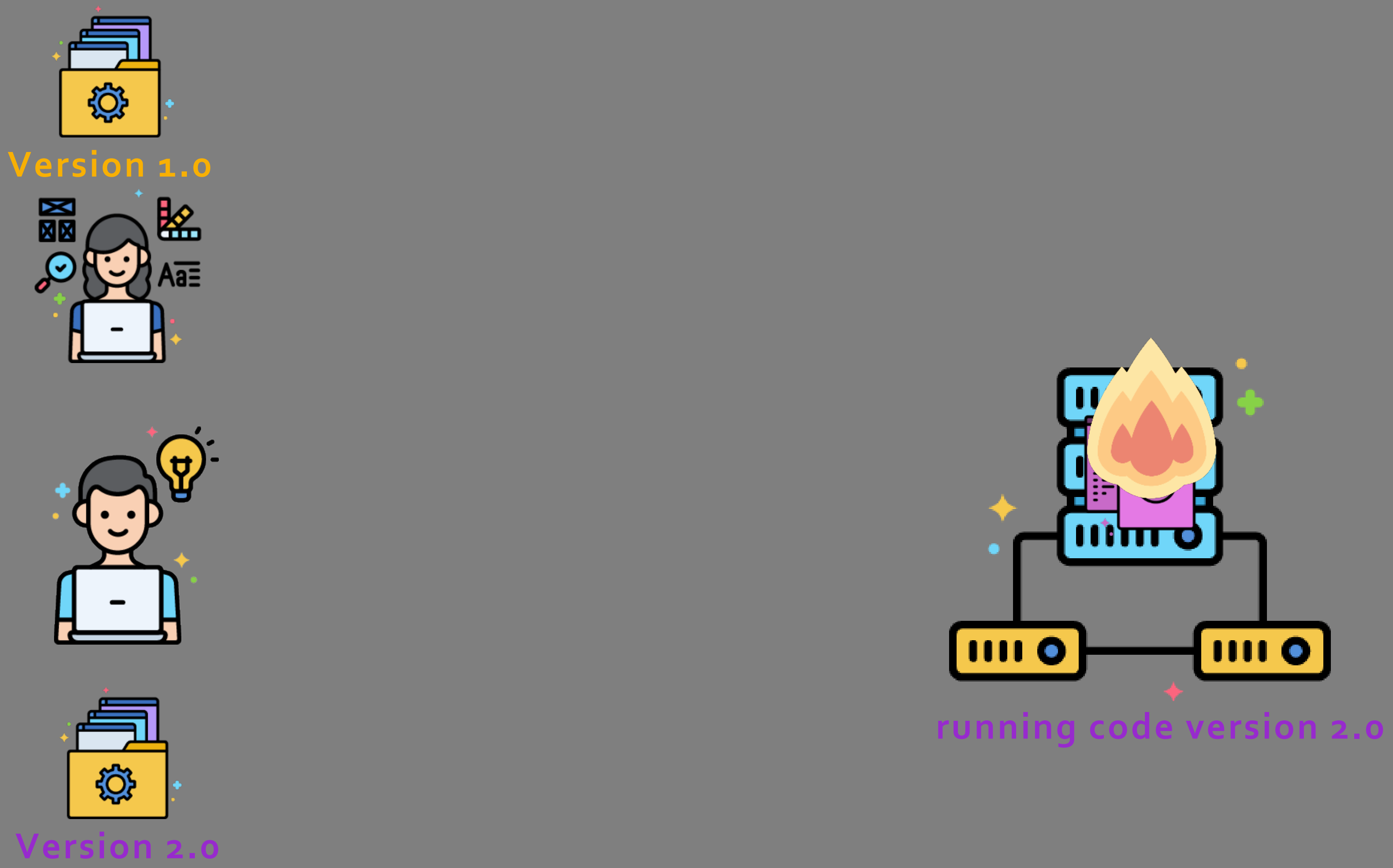
Before Version Control System



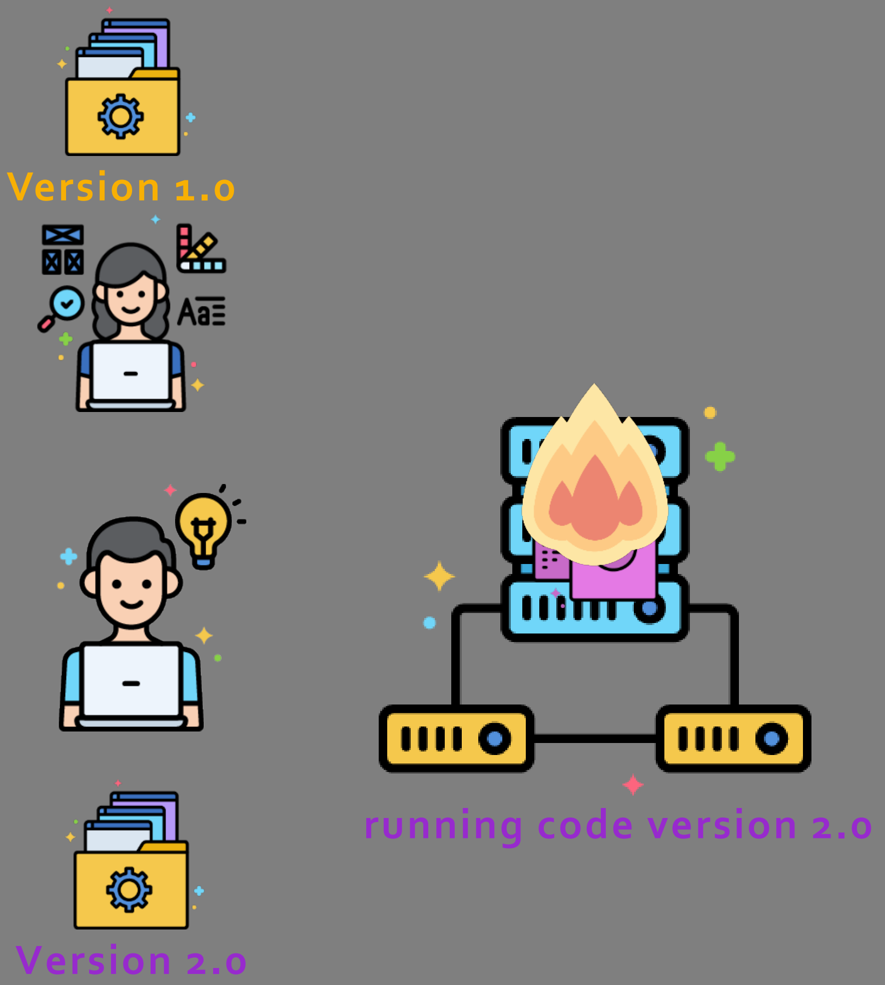
Before Version Control System



Before Version Control System

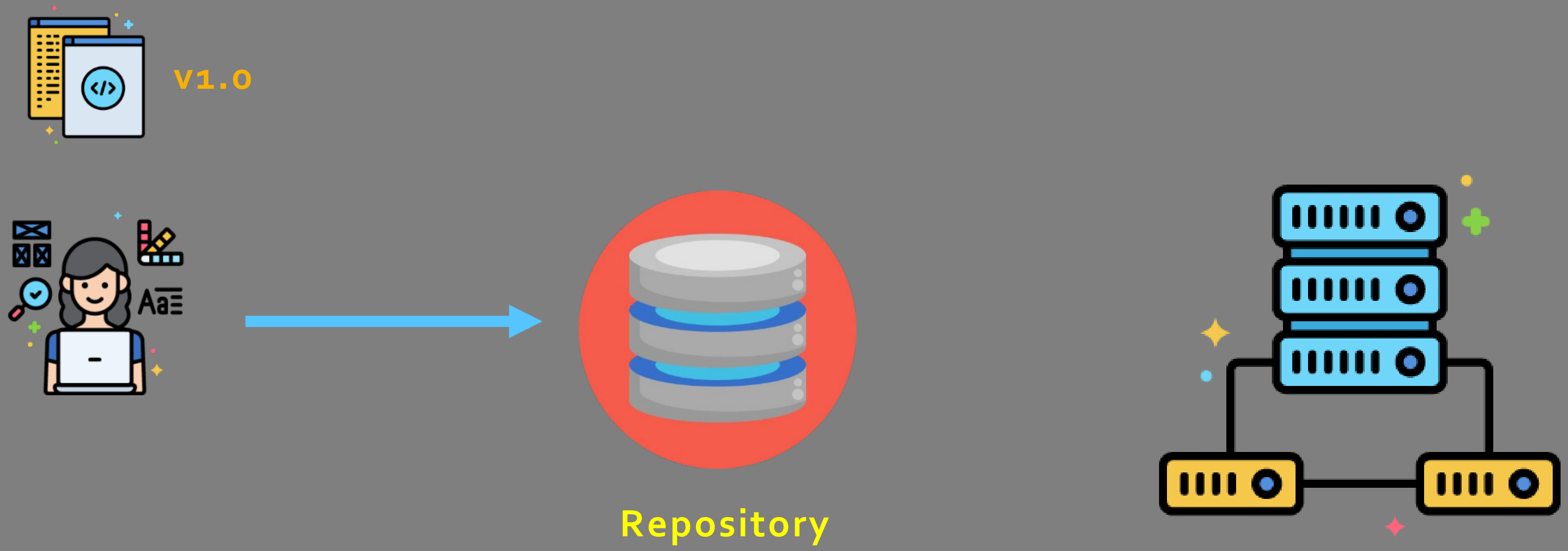


Before Version Control System

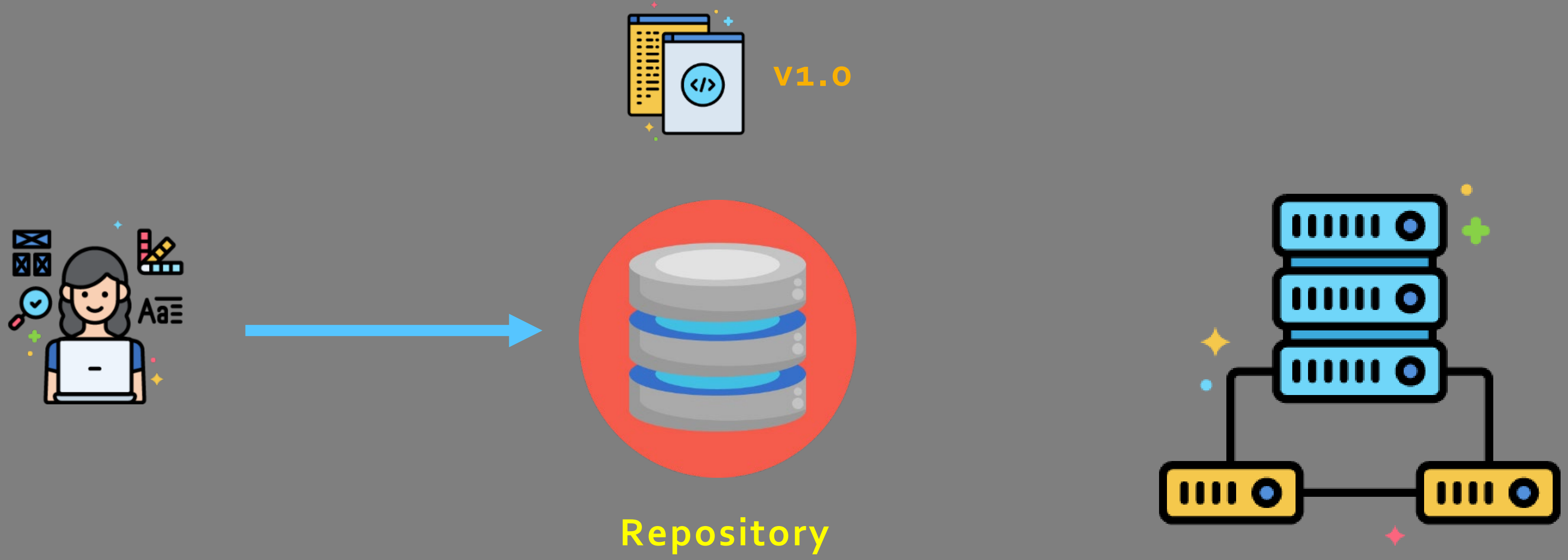


- Rollback is time consuming
- No audit tracking
- Not scalable for large teams

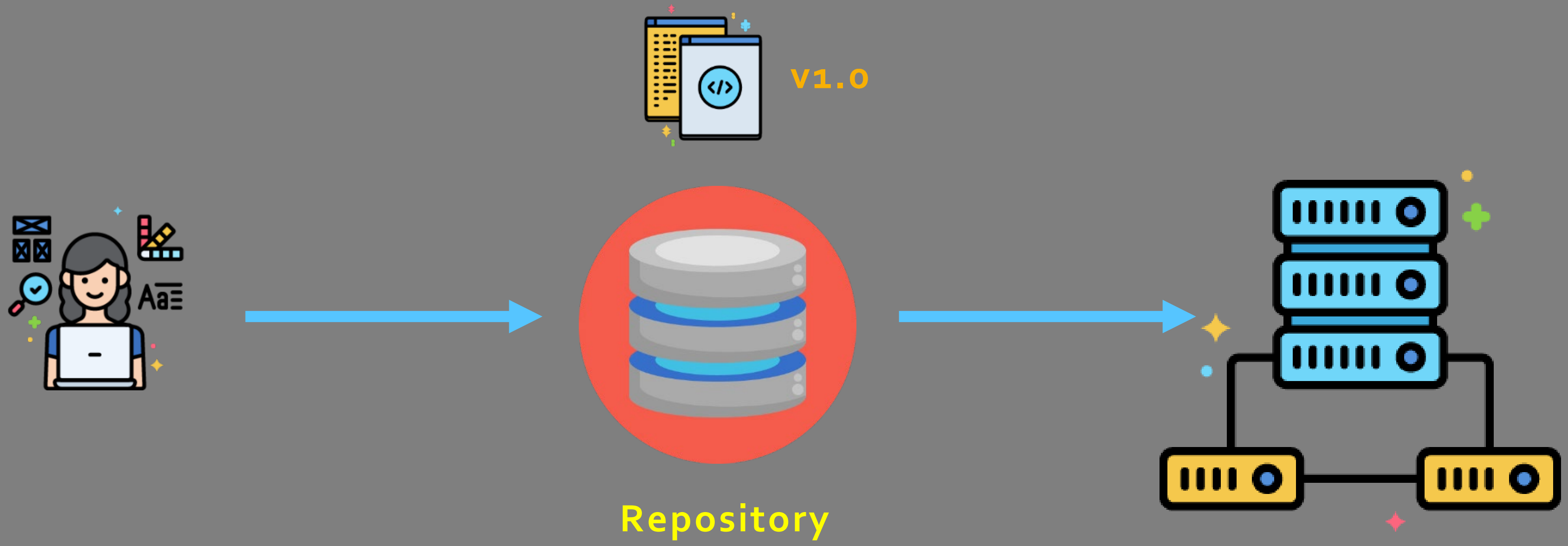
Version Control System



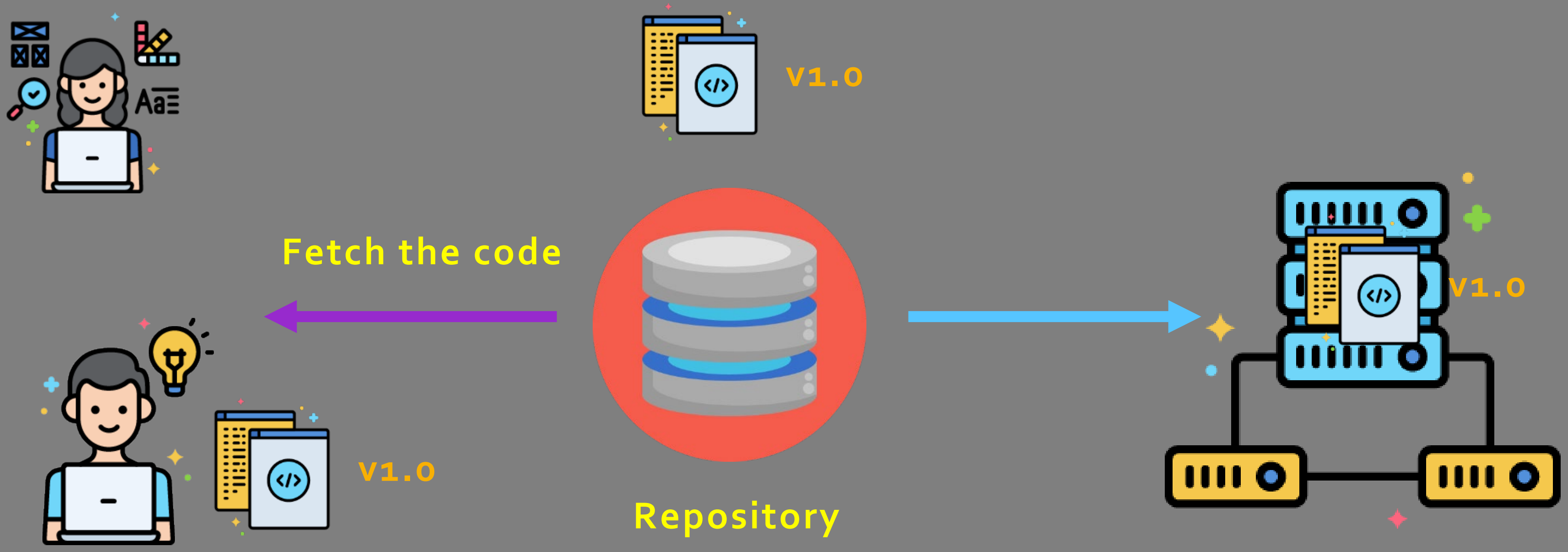
Version Control System



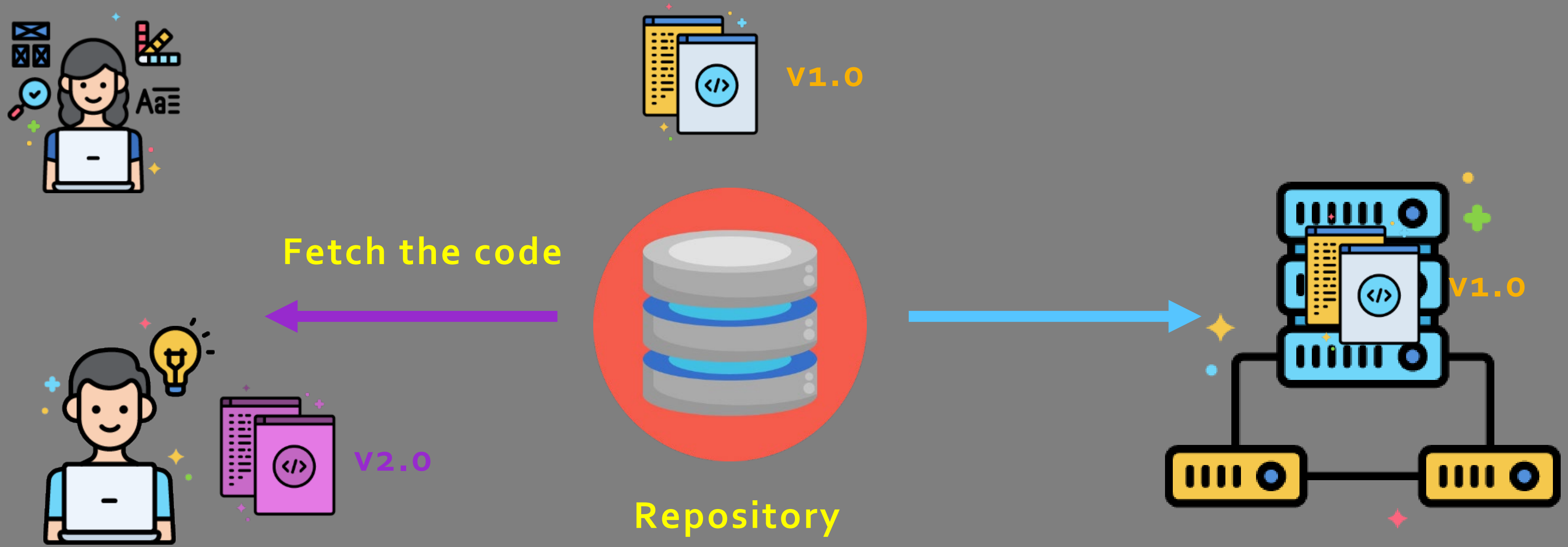
Version Control System



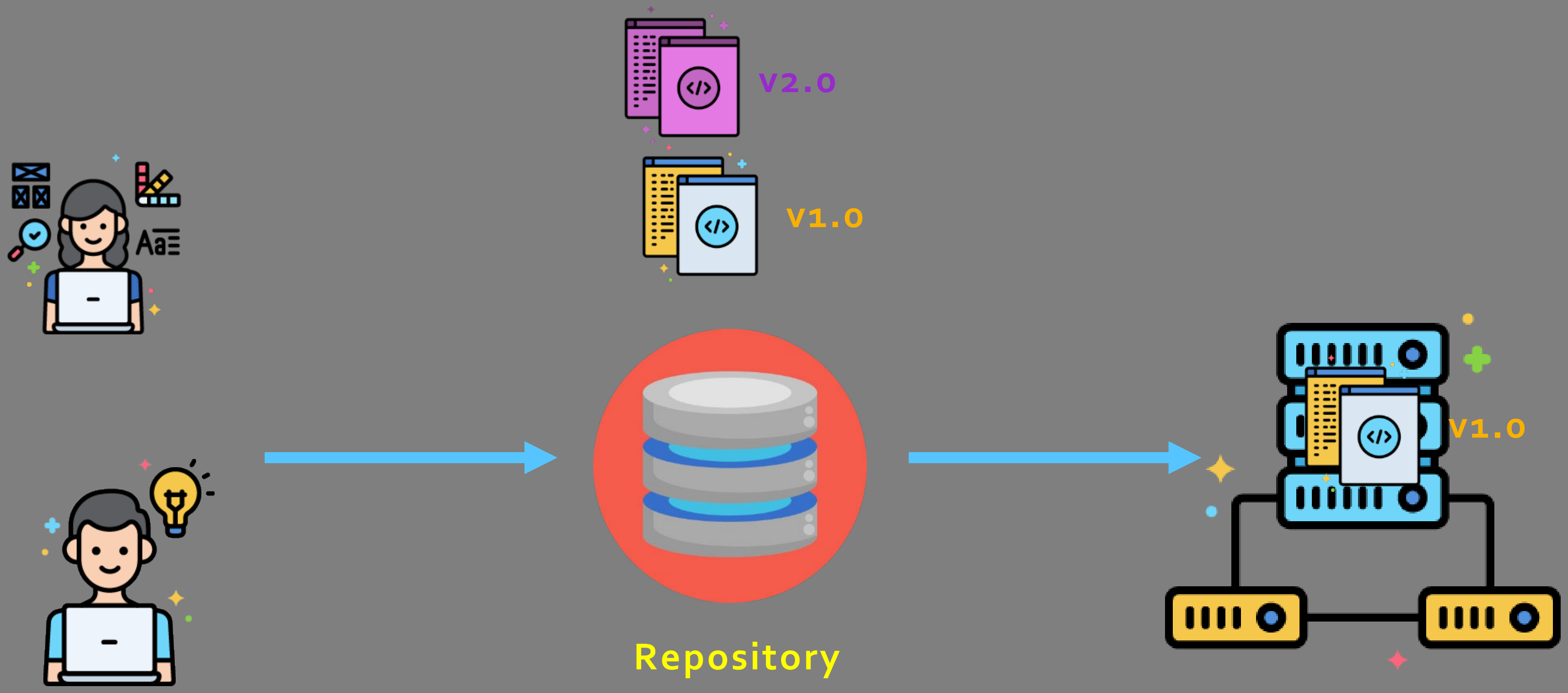
Version Control System



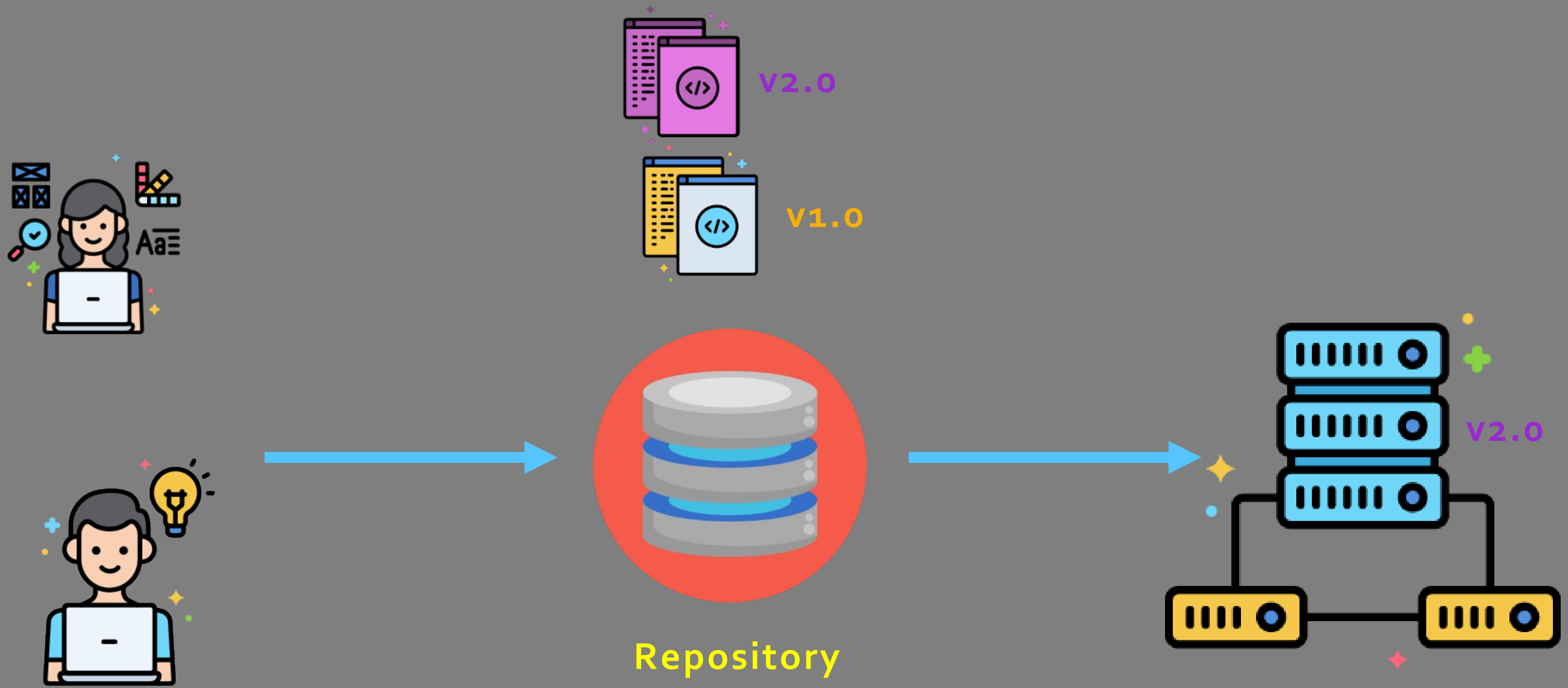
Version Control System



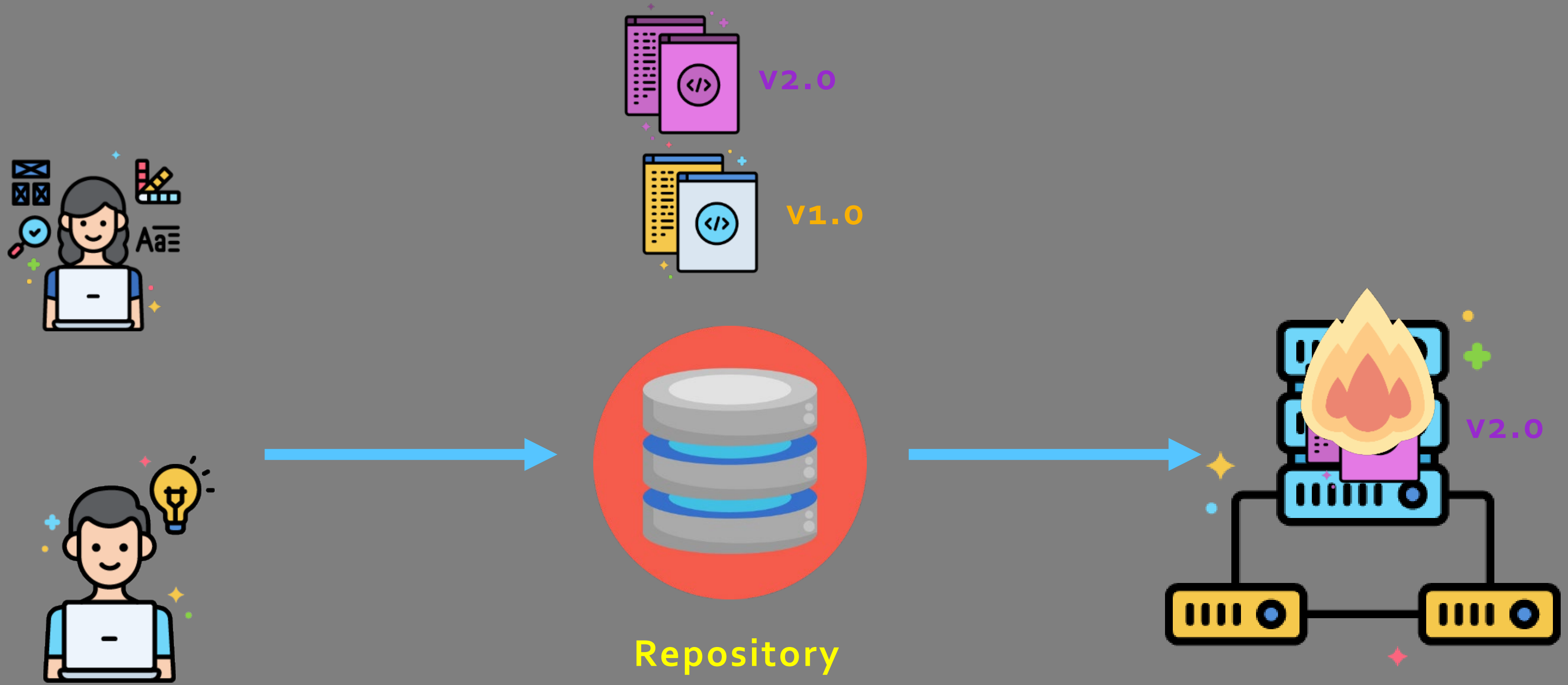
Version Control System



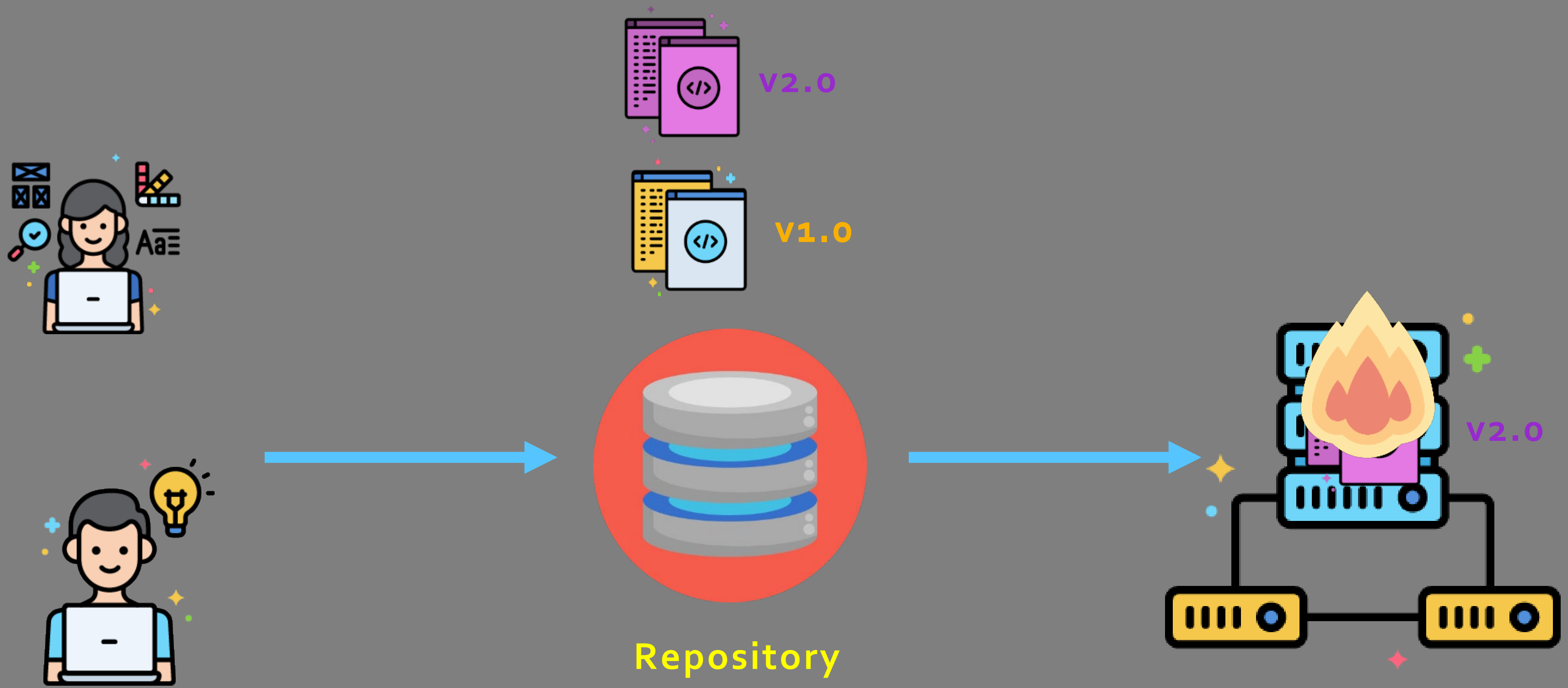
Version Control System



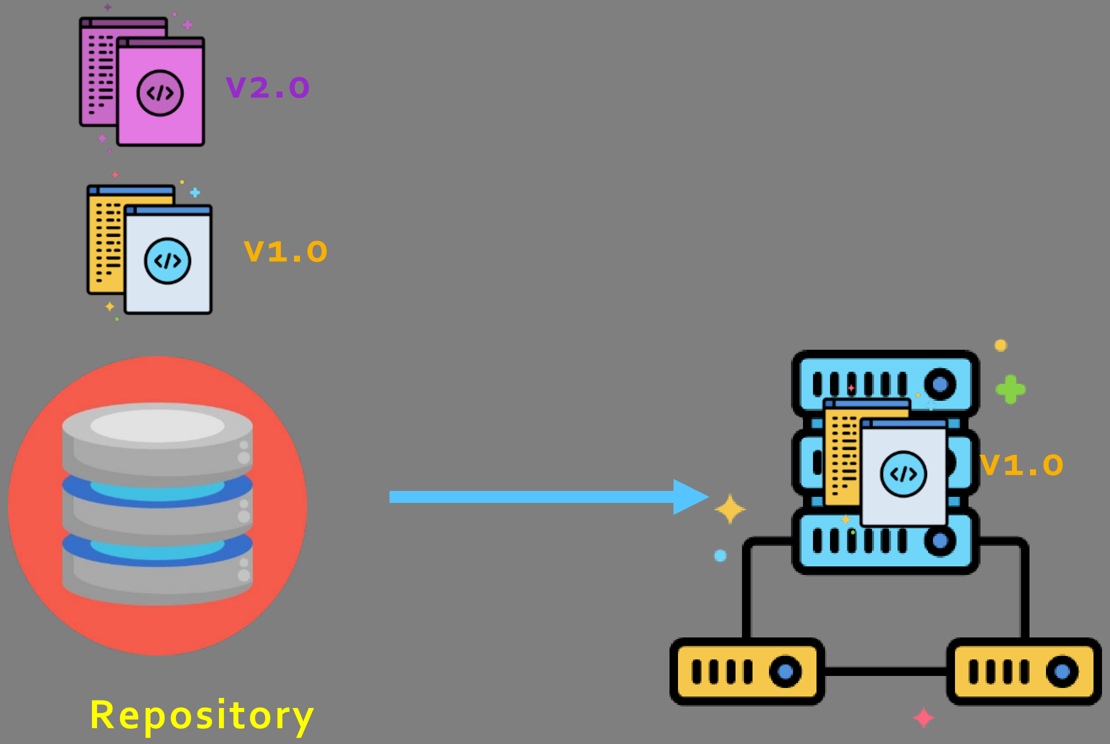
Version Control System



Version Control System



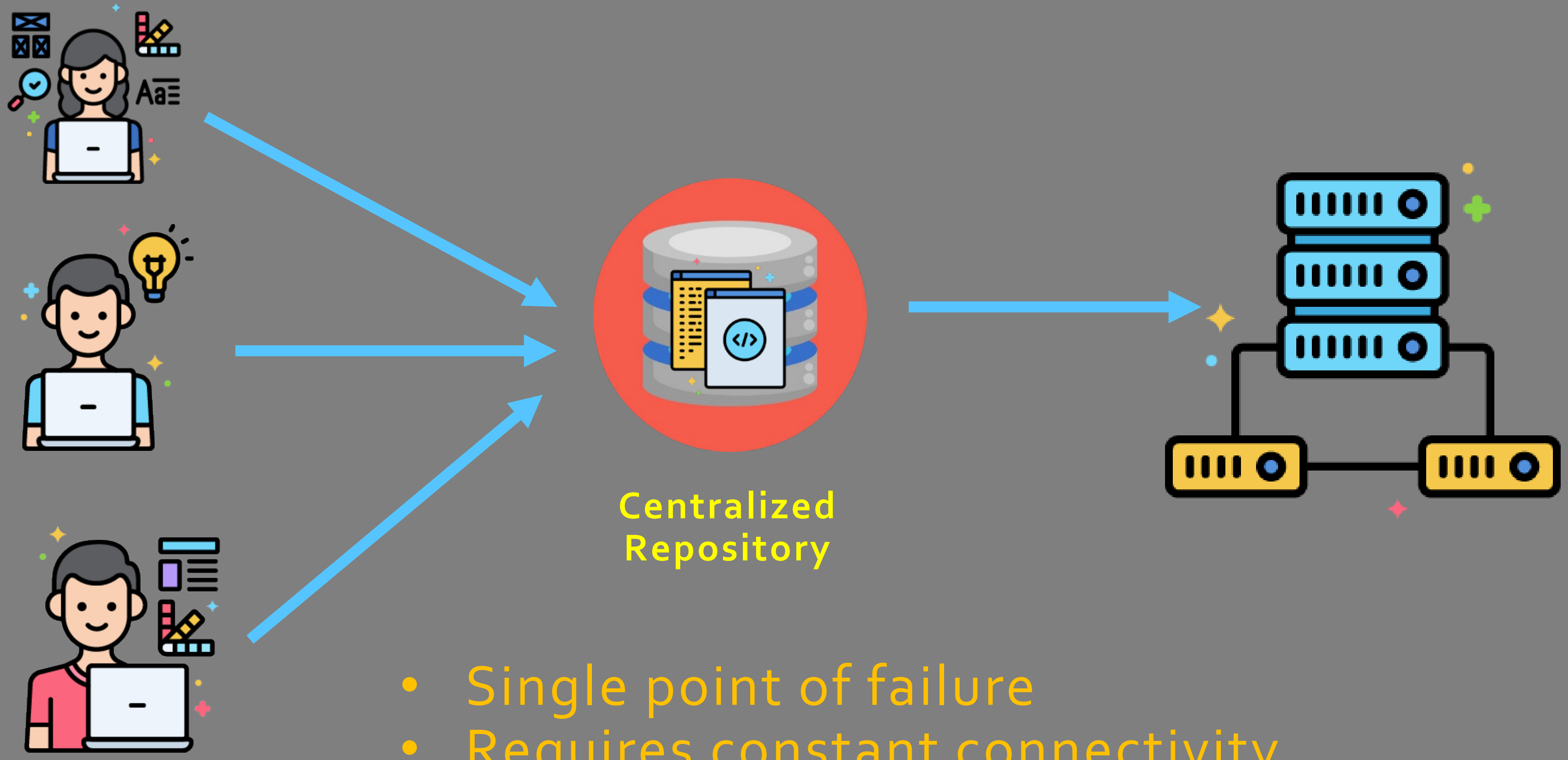
Version Control System - Git



Why Git?

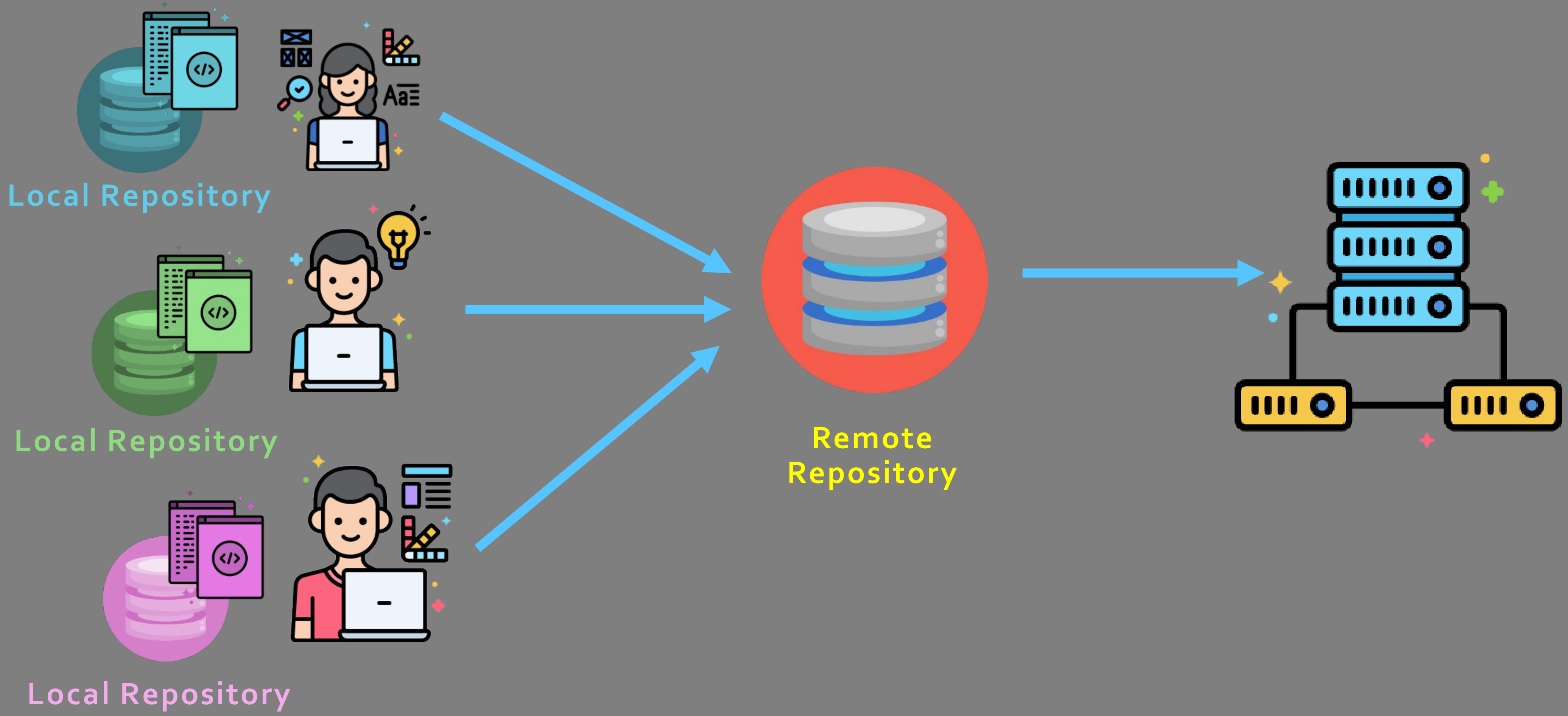
- Distributed

Centralized Version Control System

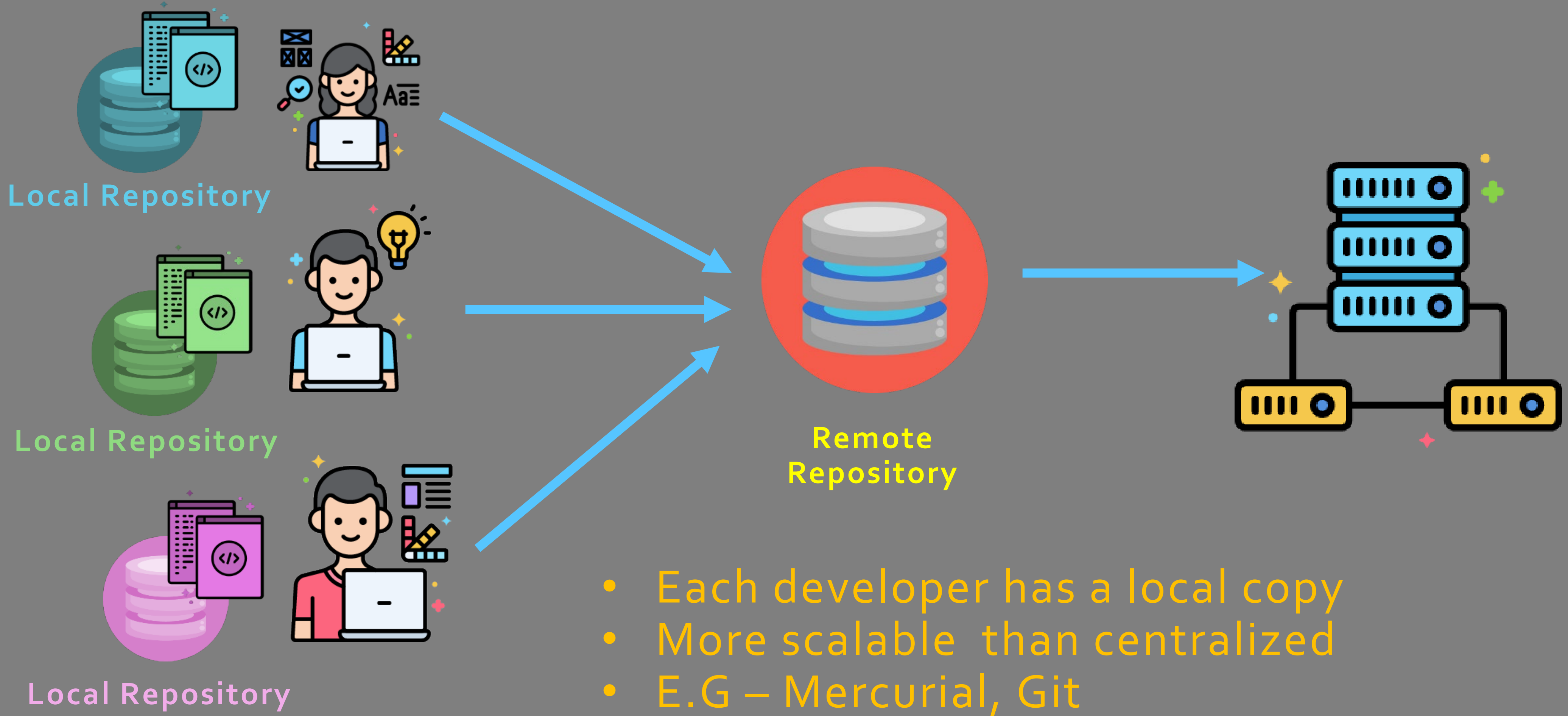


- Single point of failure
- Requires constant connectivity
- E.G – Subversion, Endevor

Distributed Version Control System

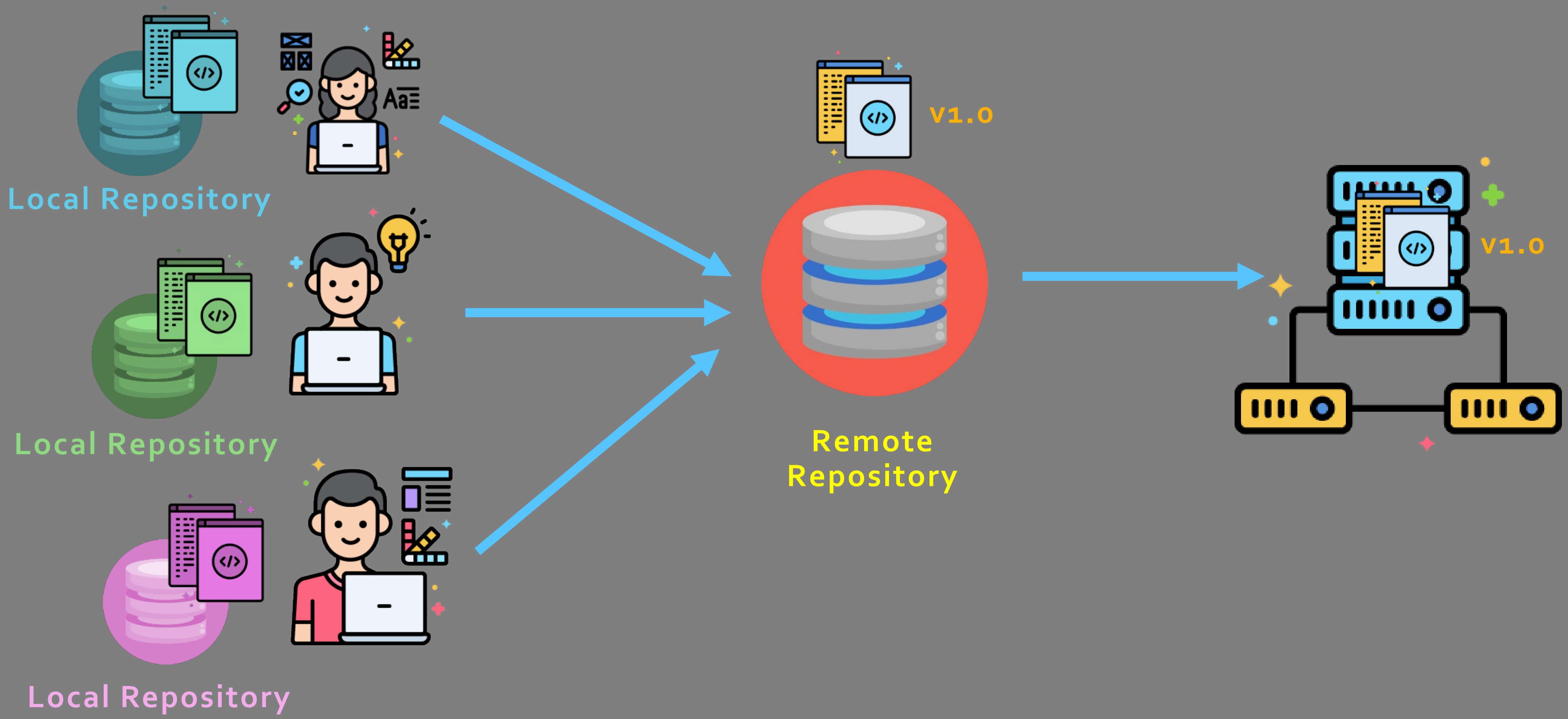


Distributed Version Control System



- Each developer has a local copy
- More scalable than centralized
- E.G – Mercurial, Git

Distributed Version Control System



Version Control System

Why Git?

- Distributed
- Performant
- Detailed audit tracking
- Open source
  - Free!
  - Implemented with Kubernetes GitOps, integration with Jenkins and other DevOps tools
  - GitHub, GitLab, Code Commit are all based on Git

Git vs GitHub

Raj Saha
cloudwithraj.com
Cloud With Raj

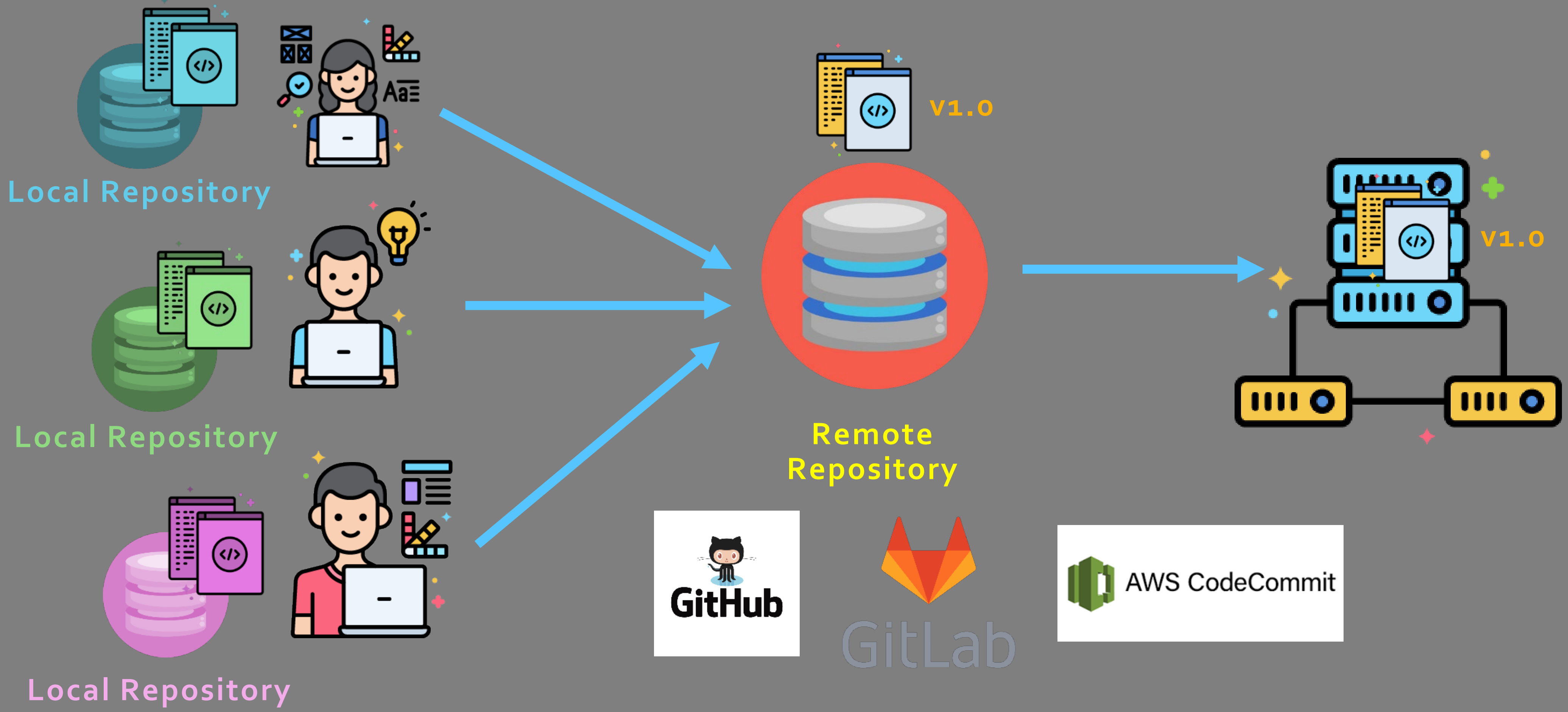
Git Vs. GitHub

- Version Control System
- Installed locally on the system
- Created in 2005, by Linus Torvalds
- Open source, and used in multiple cloud repository services

GitHub

- Git repository hosting services with other features
- Runs on the cloud
- Created in 2008, currently owned by Microsoft
- Not open source, have free and paid tiers

Distributed Version Control System



Ways to use Git

- Command line interface (CLI)
- GUI
- Within DevOps tools

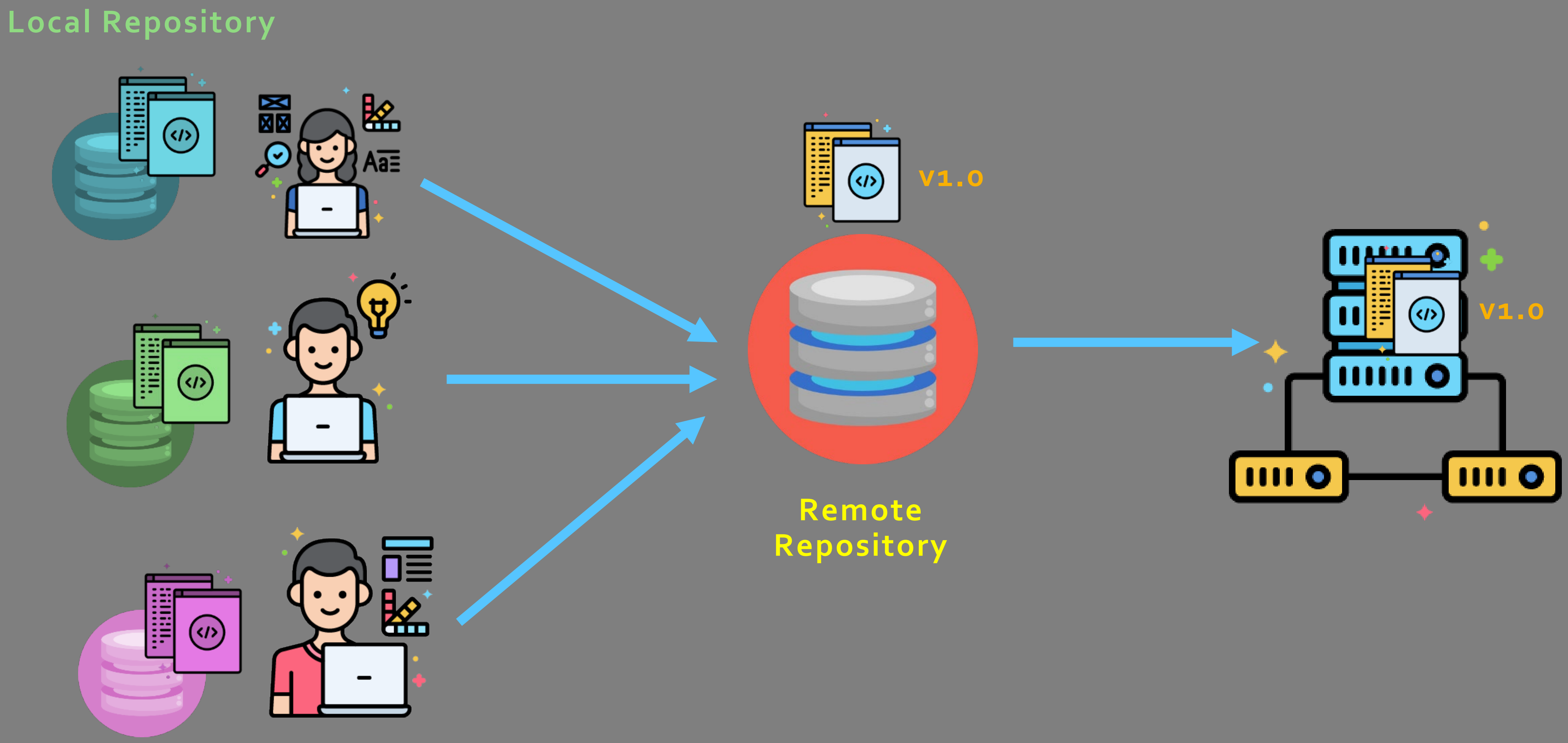
For our Course

- Command line interface (CLI)
  - Visual studio code
- GUI
  - Git Graph
- Repository
  - GitHub
- Within DevOps tools
  - Jenkins

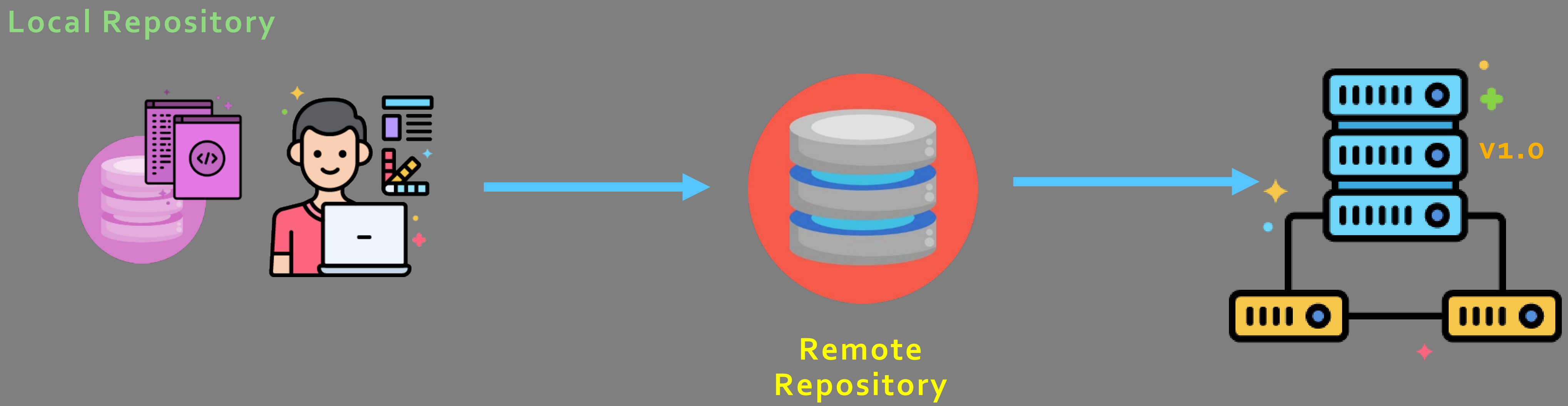
Git Workflow

Raj Saha
cloudwithraj.com
Cloud With Raj

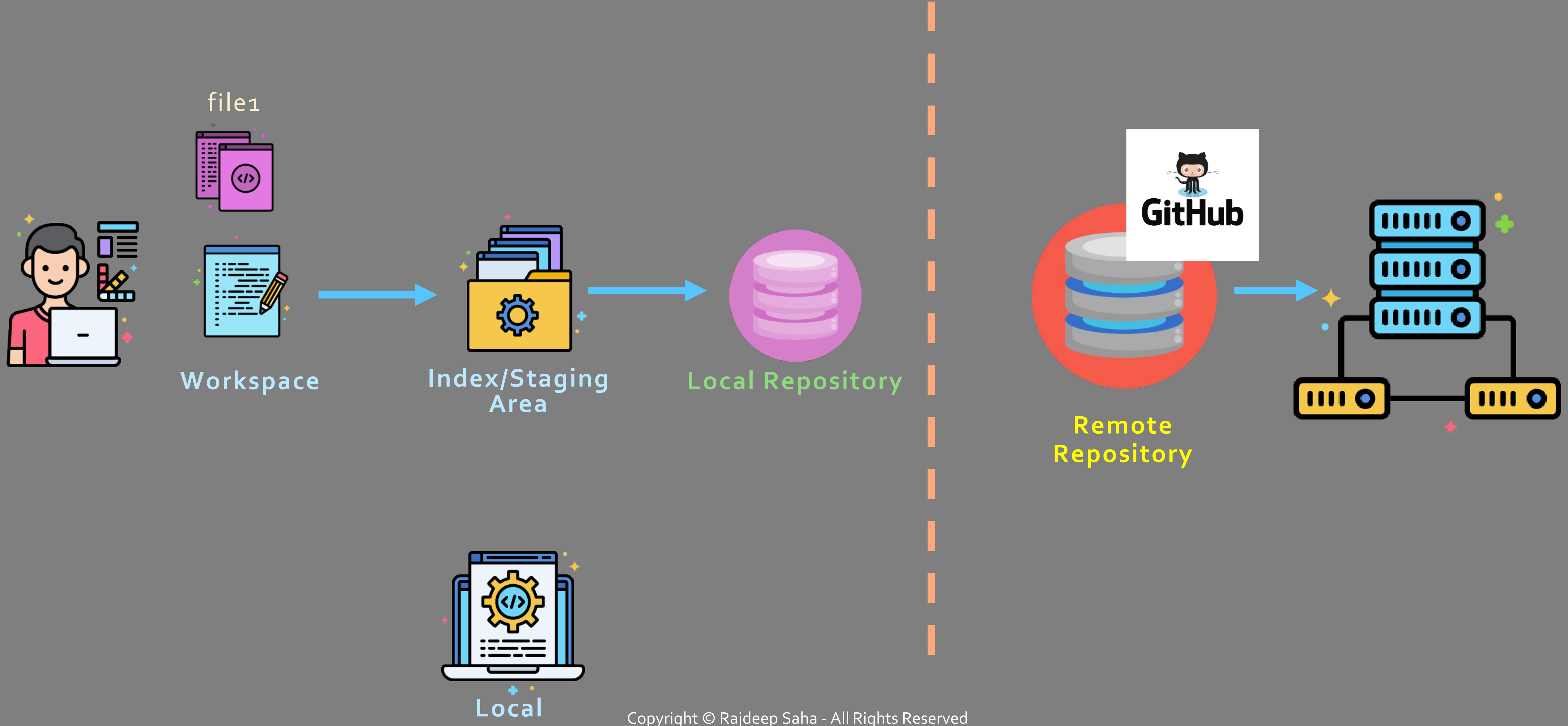
Distributed Version Control System



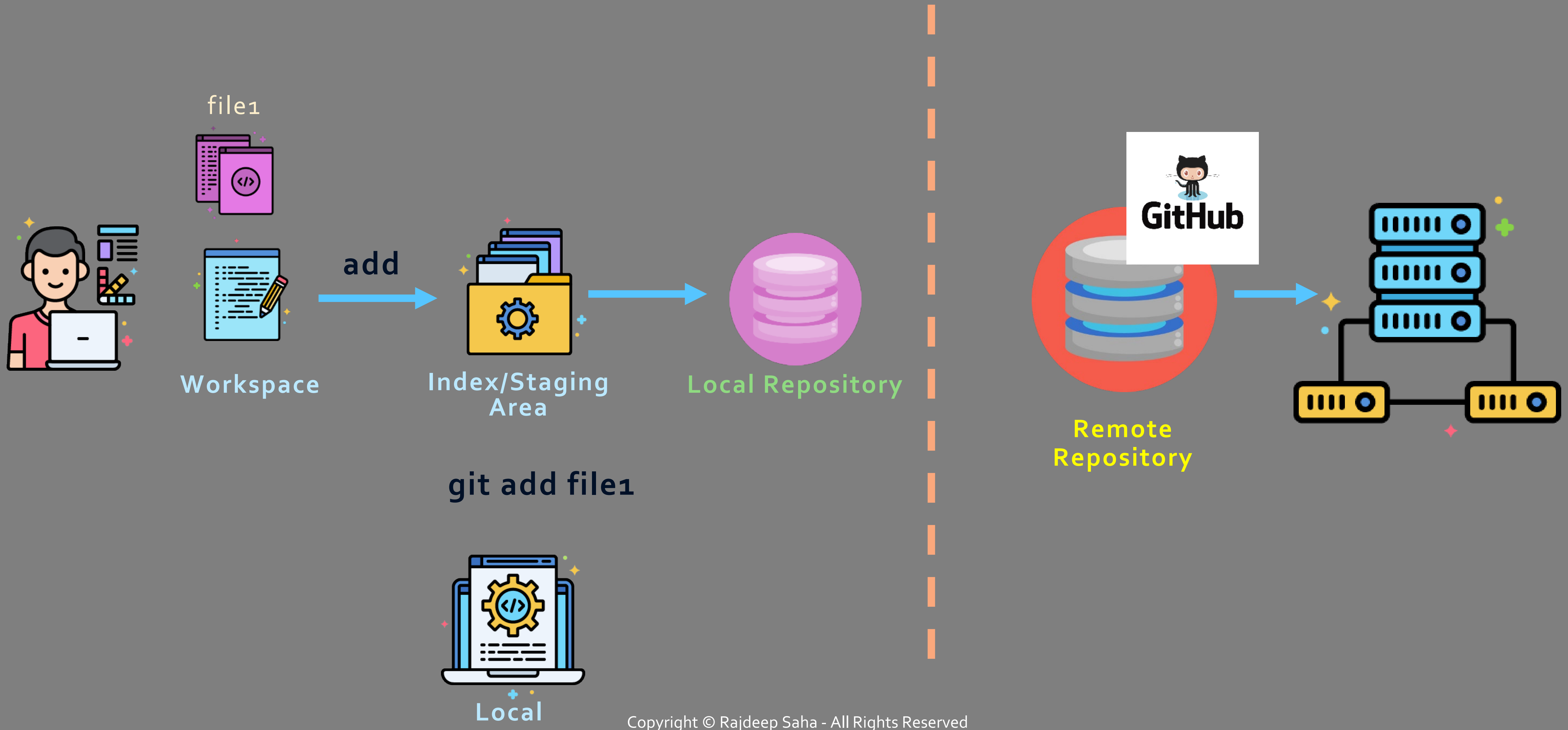
Distributed Version Control System



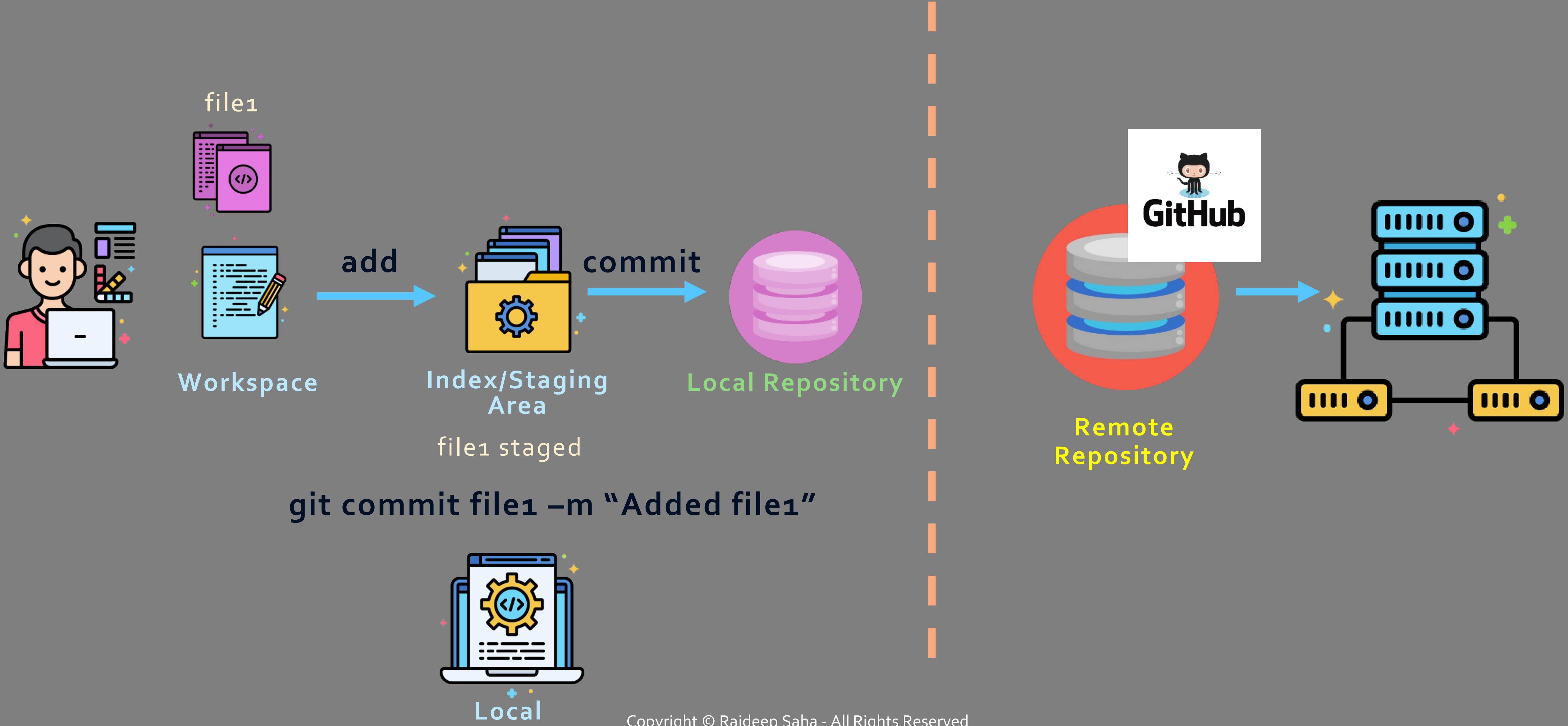
Git Workflow



Git Workflow

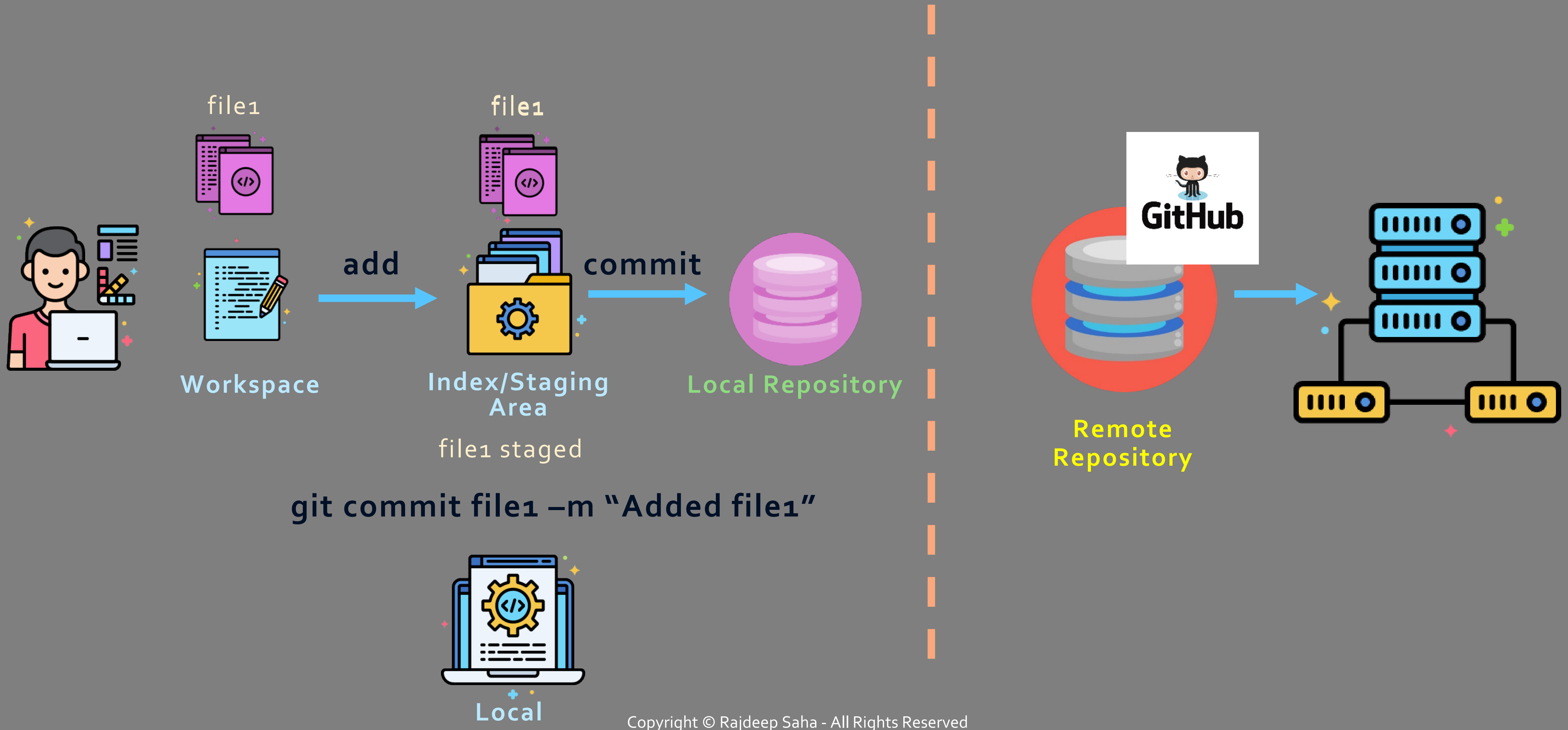


Git Workflow



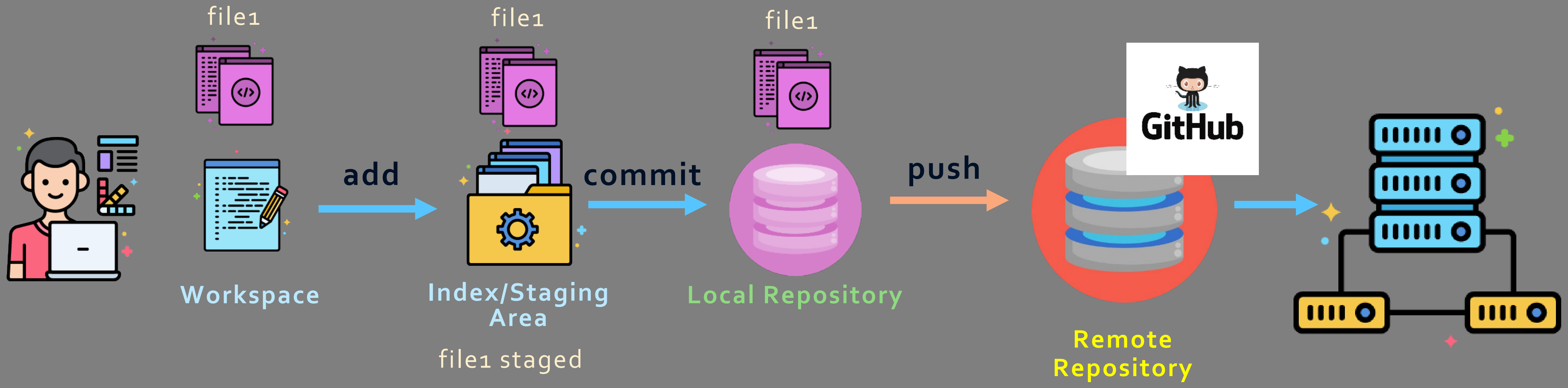
```
git commit file1 –m "Added file1"
```

Git Workflow



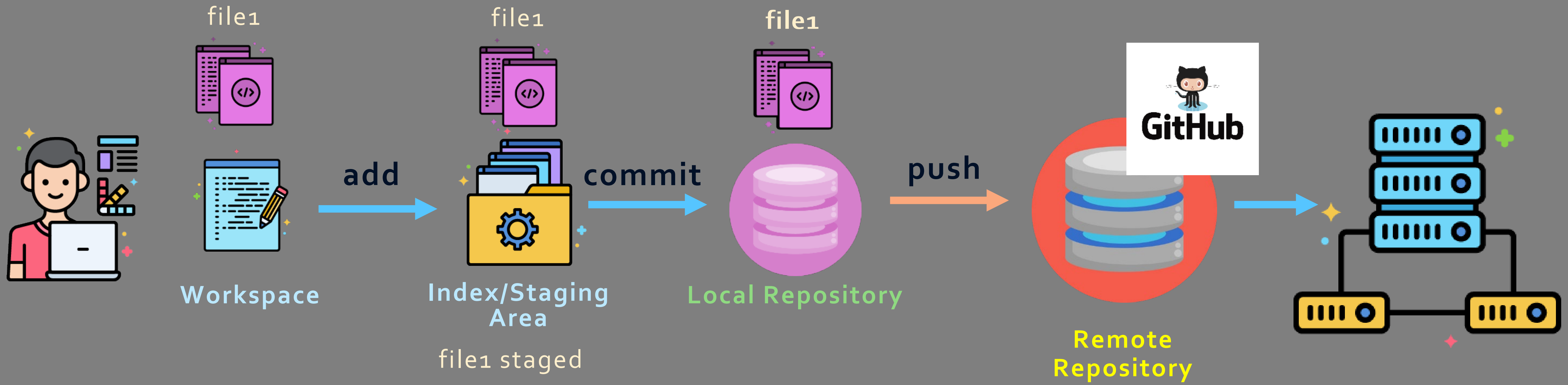
```
git commit file1 –m “Added file1”
```

Git Workflow

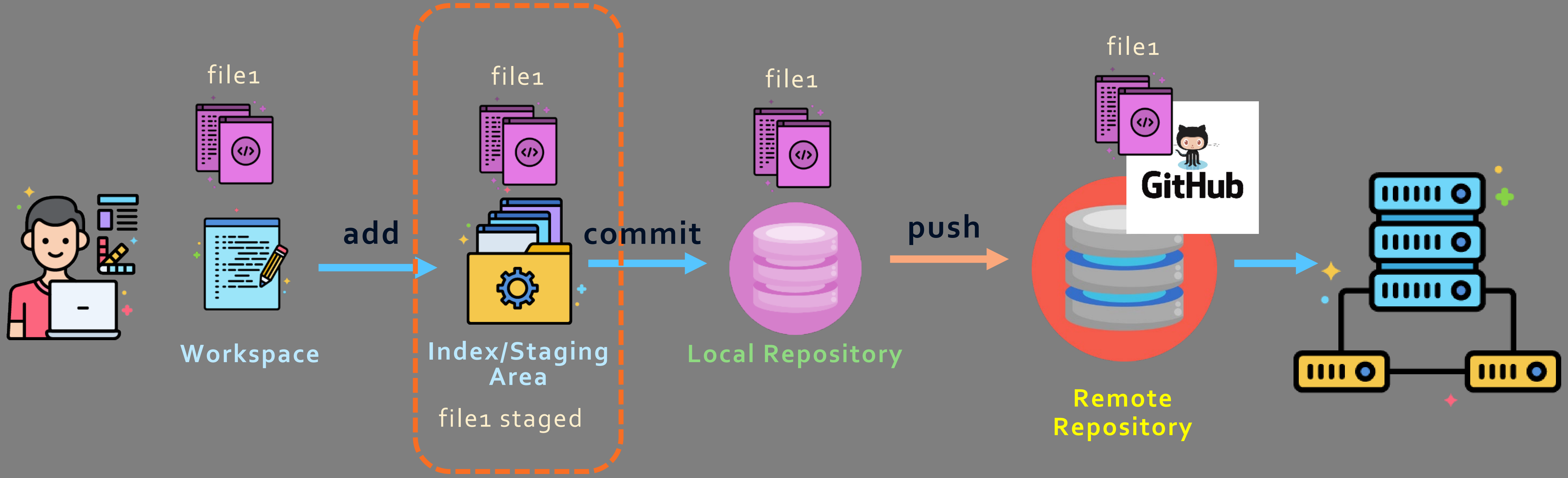


git push

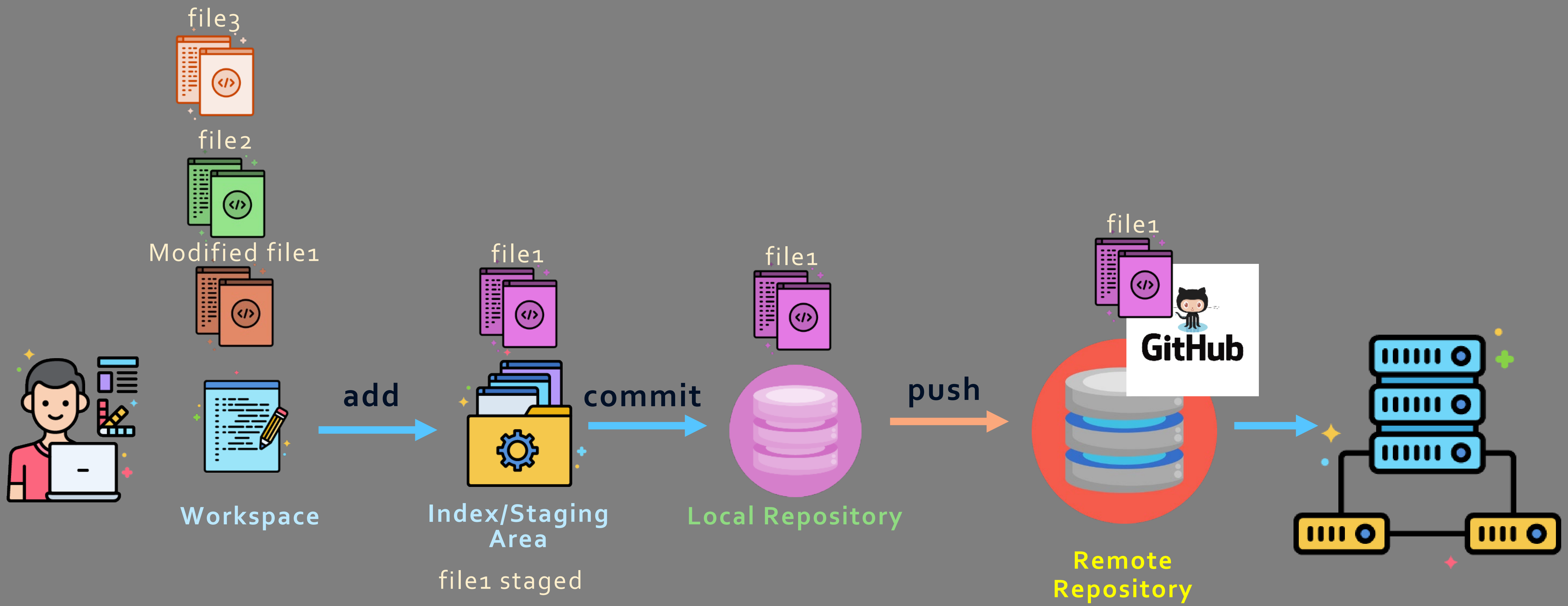
Git Workflow

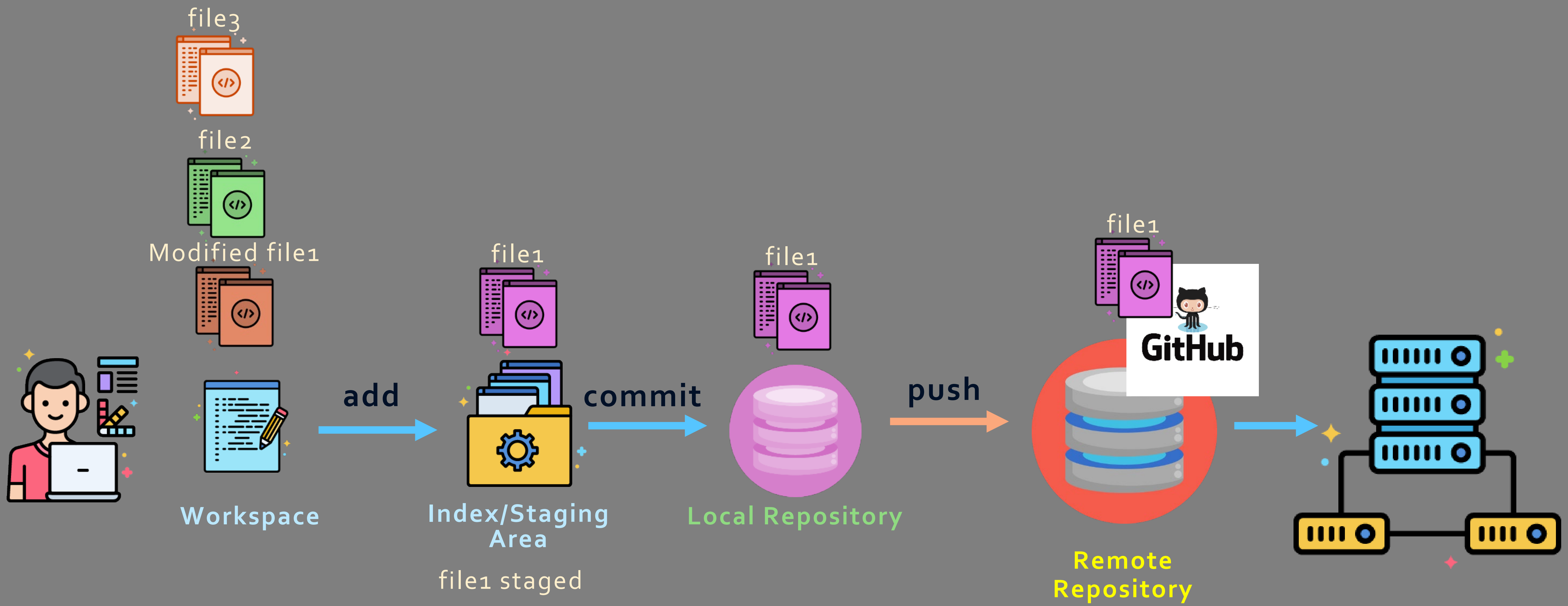


Git Workflow



- Staging area still has the file!

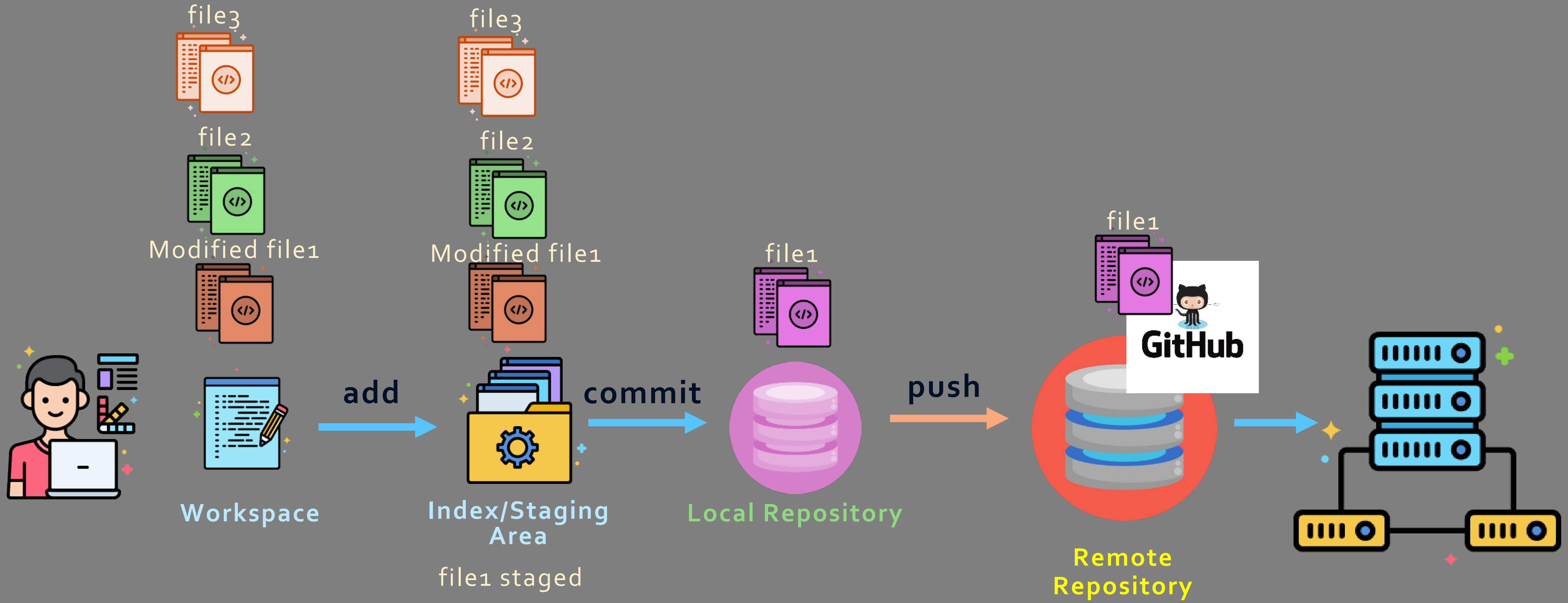

file3
file2
Modified file1
file1
file1
file1
add
commit
push
GitHub
Workspace
Index/Staging Area
file1 staged
Local Repository
Remote Repository



git add file1 file2 file3

git add -A

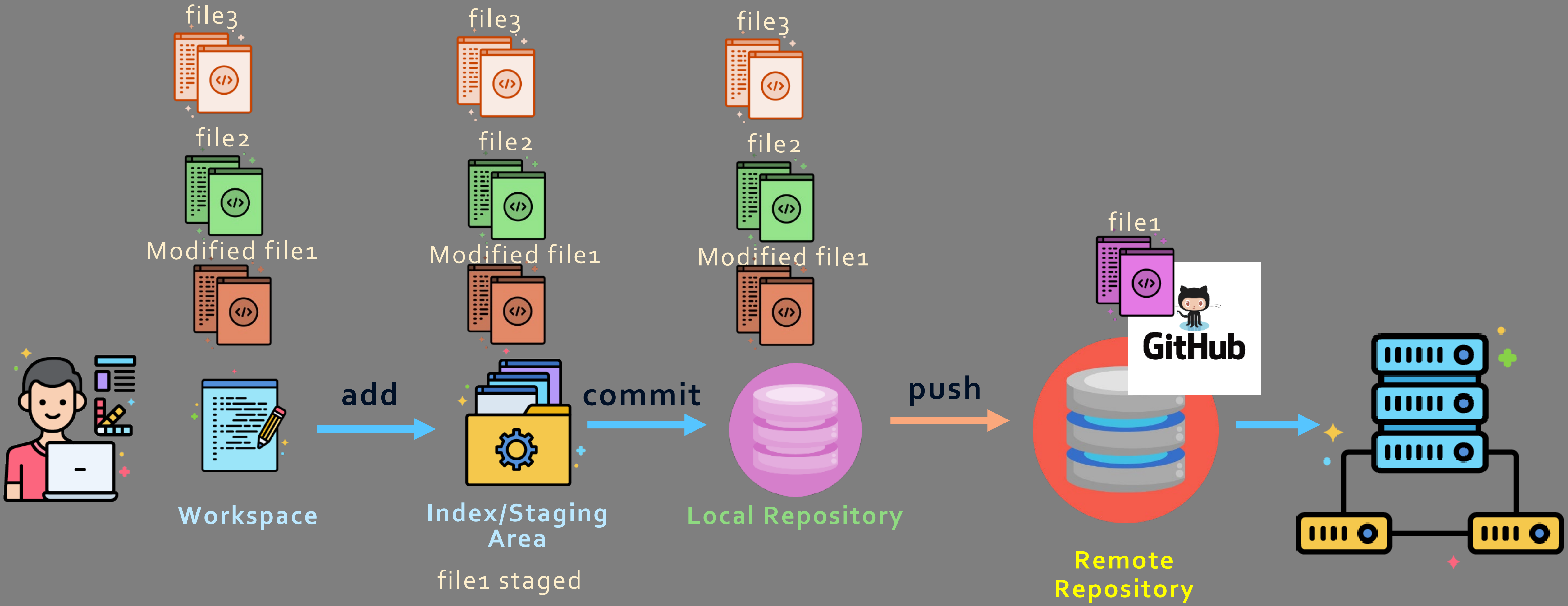
(Stage all new, modified, and <u>deleted</u> files)



git commit file1 file2 file3 –m "Added new feature1"

git commit –m "Added new feature1"

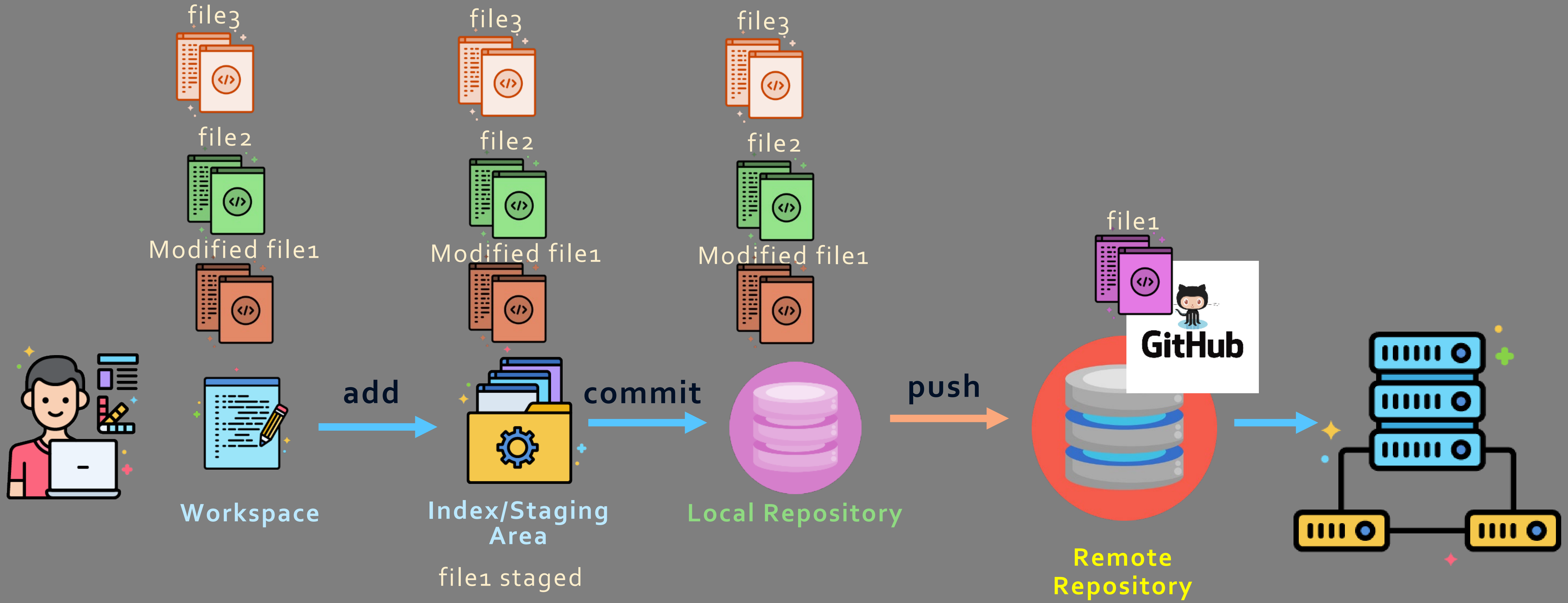
(Move the changes from stage to local repo)



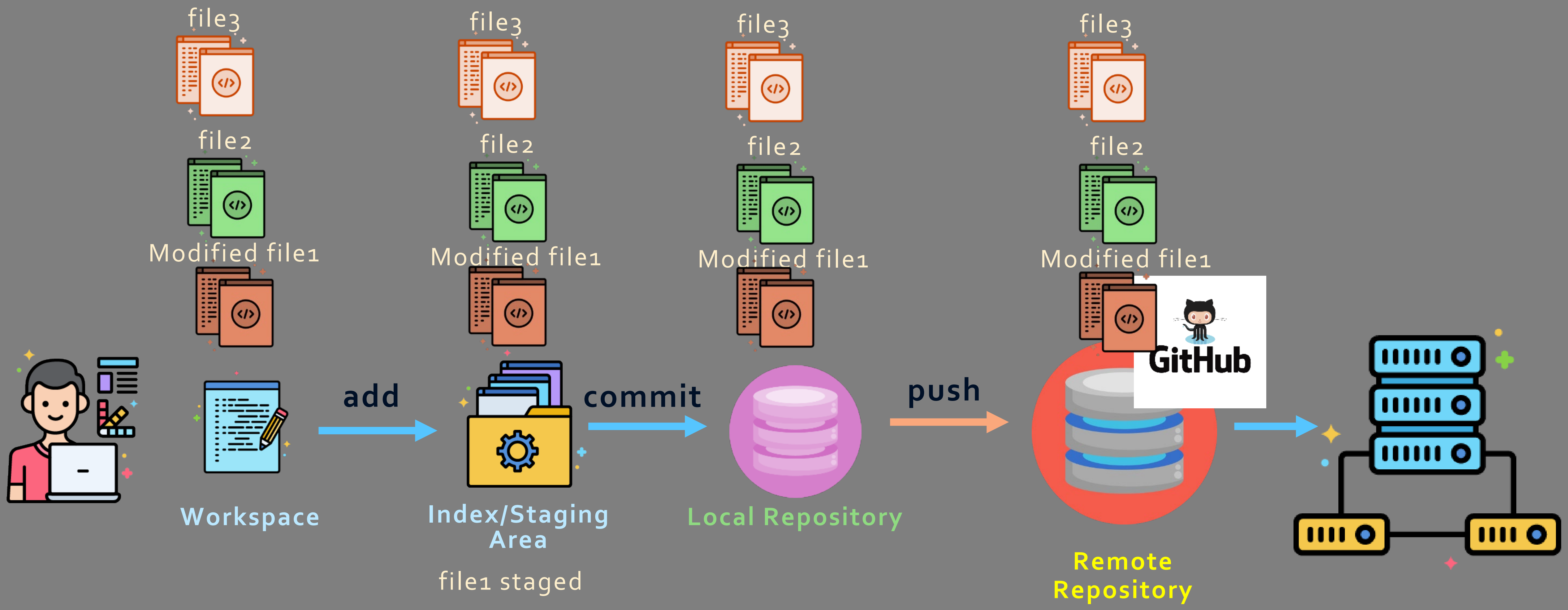
git commit file1 file2 file3 –m "Added new feature1"

git commit –m "Added new feature1"

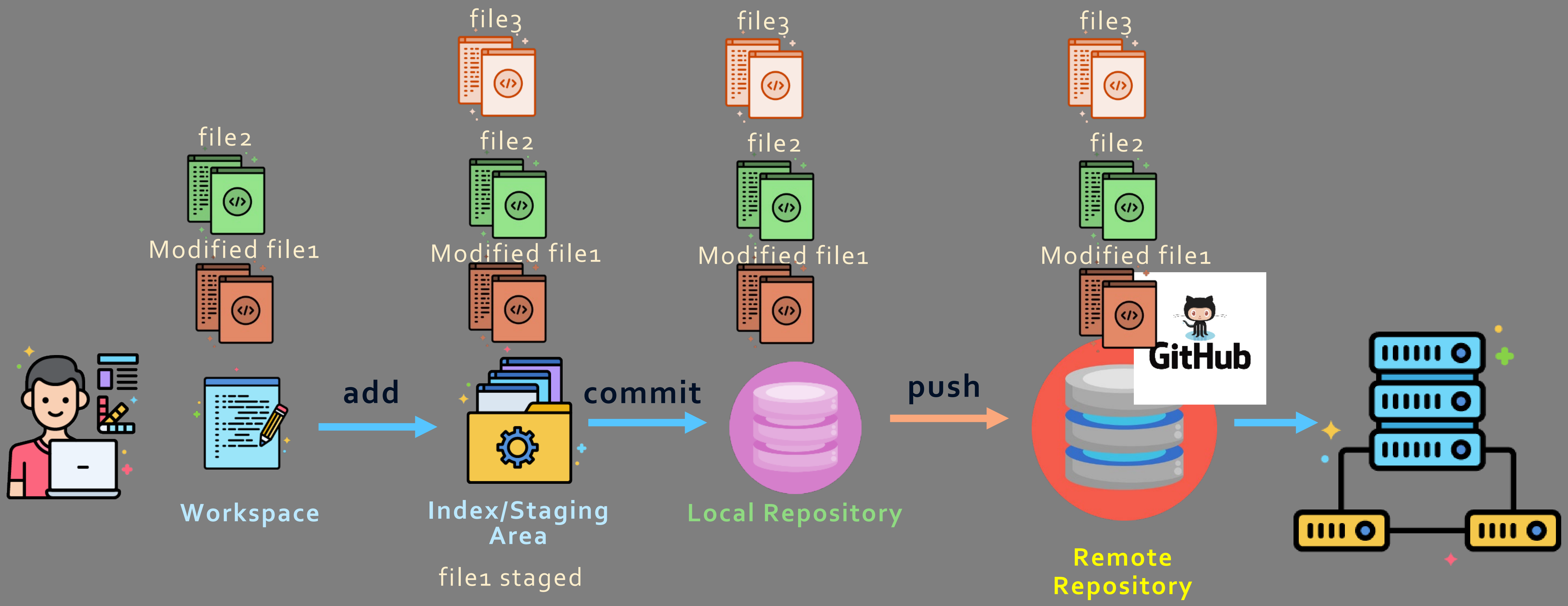
(Move the changes from stage to local repo)


file3
file2
Modified file1
file3
file2
Modified file1
file3
file2
Modified file1
file1
GitHub
add
commit
push
Workspace
Index/Staging Area
file1 staged
Local Repository
Remote Repository

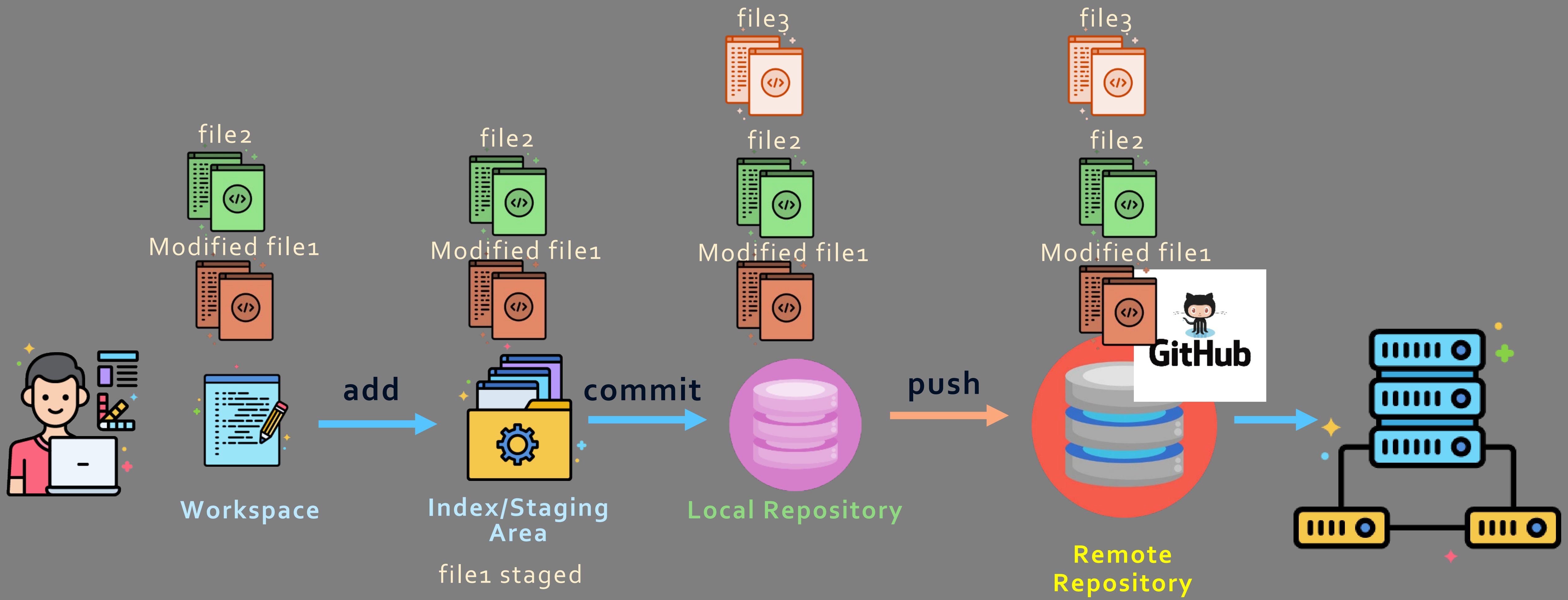
git push


file3
file2
Modified file1
file3
file2
Modified file1
file3
file2
Modified file1
file3
file2
Modified file1
GitHub
add
commit
push
Workspace
Index/Staging Area
file1 staged
Local Repository
Remote Repository

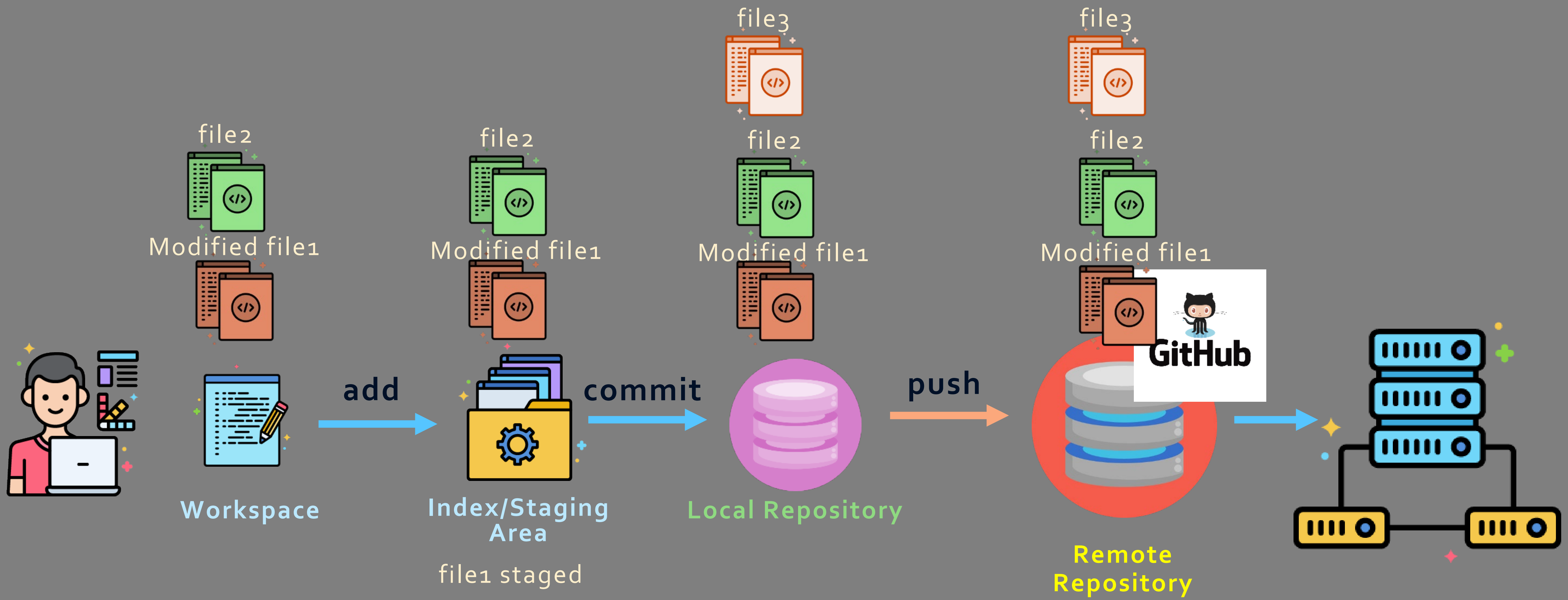
git push



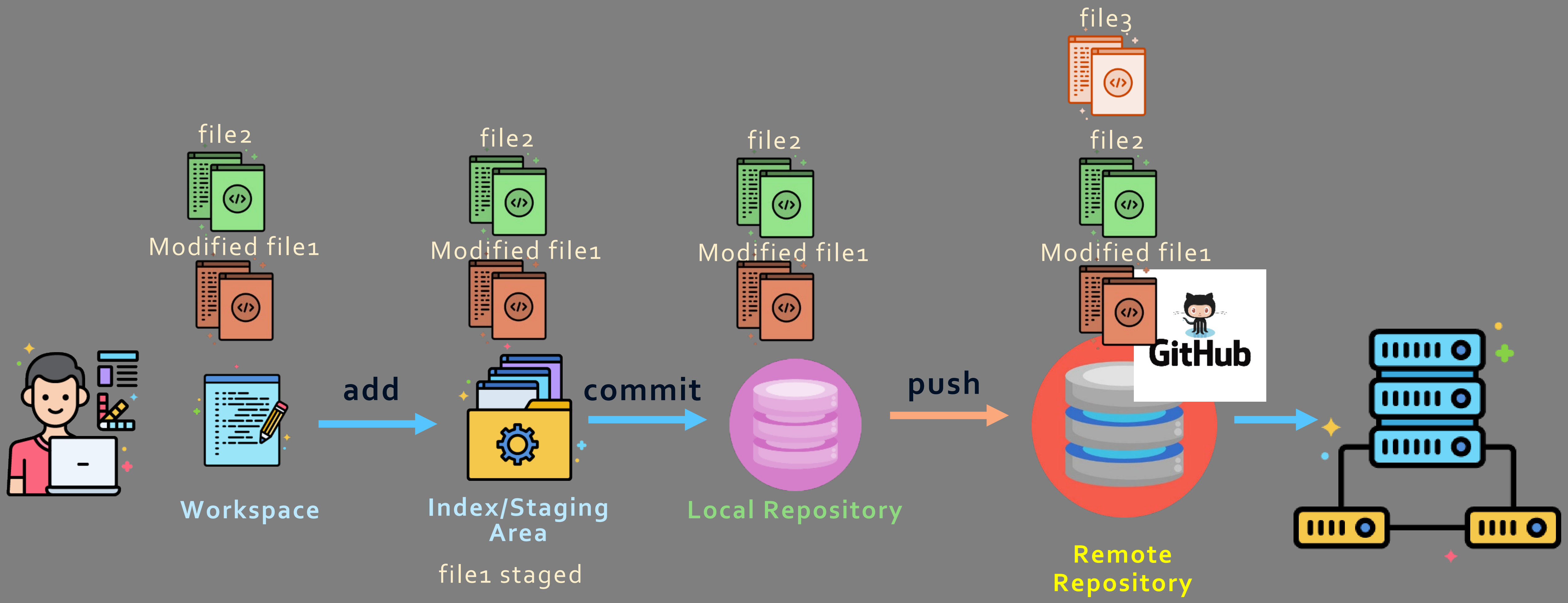
Git add file1
git add -a



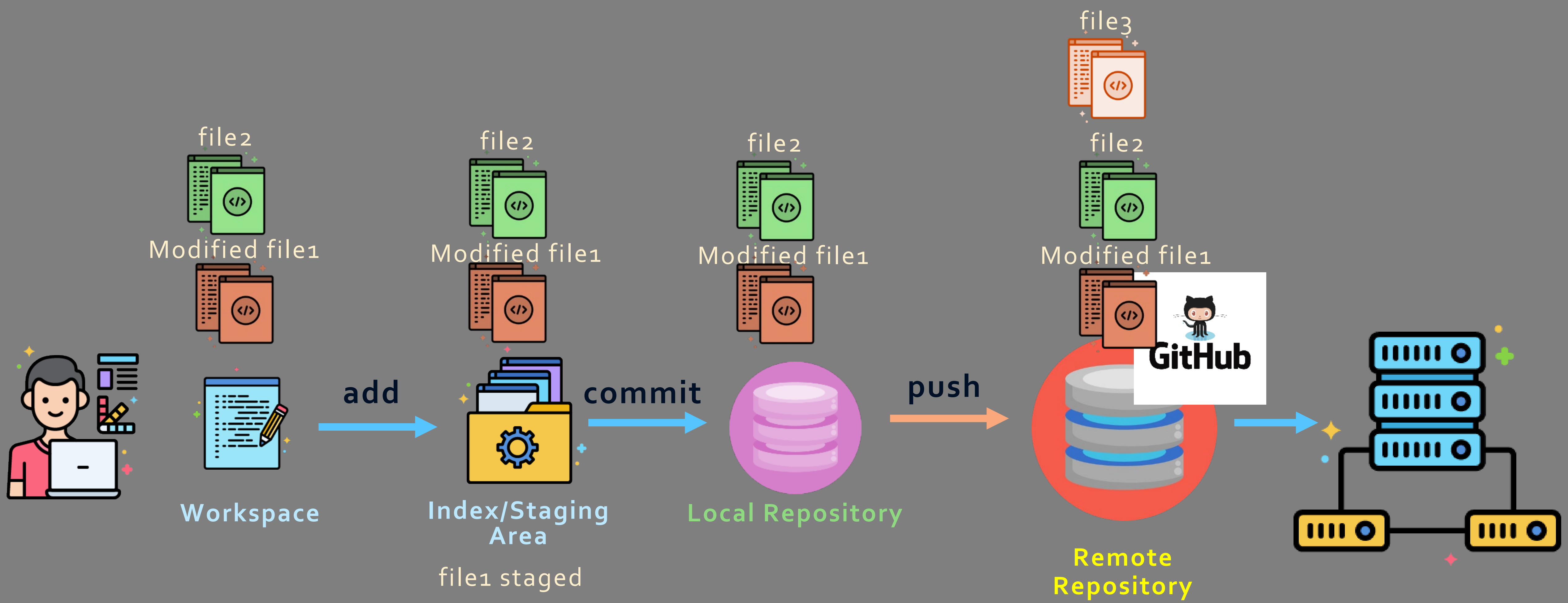
Git add file1
git add -a



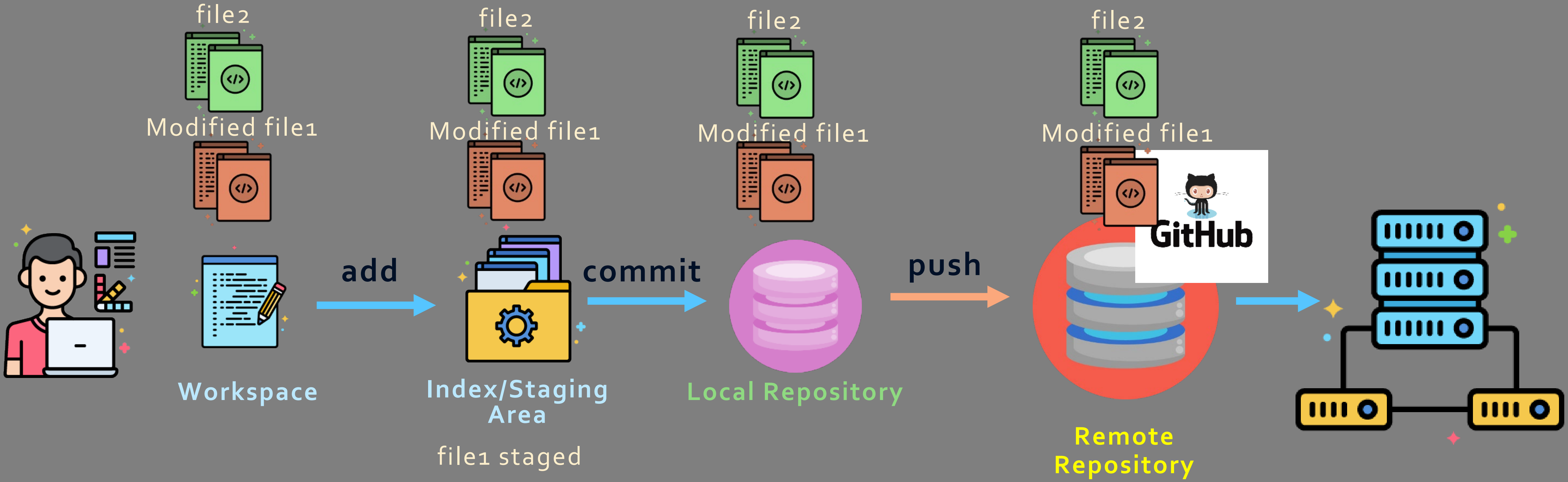
```
git commit –m “Removing file1”
```



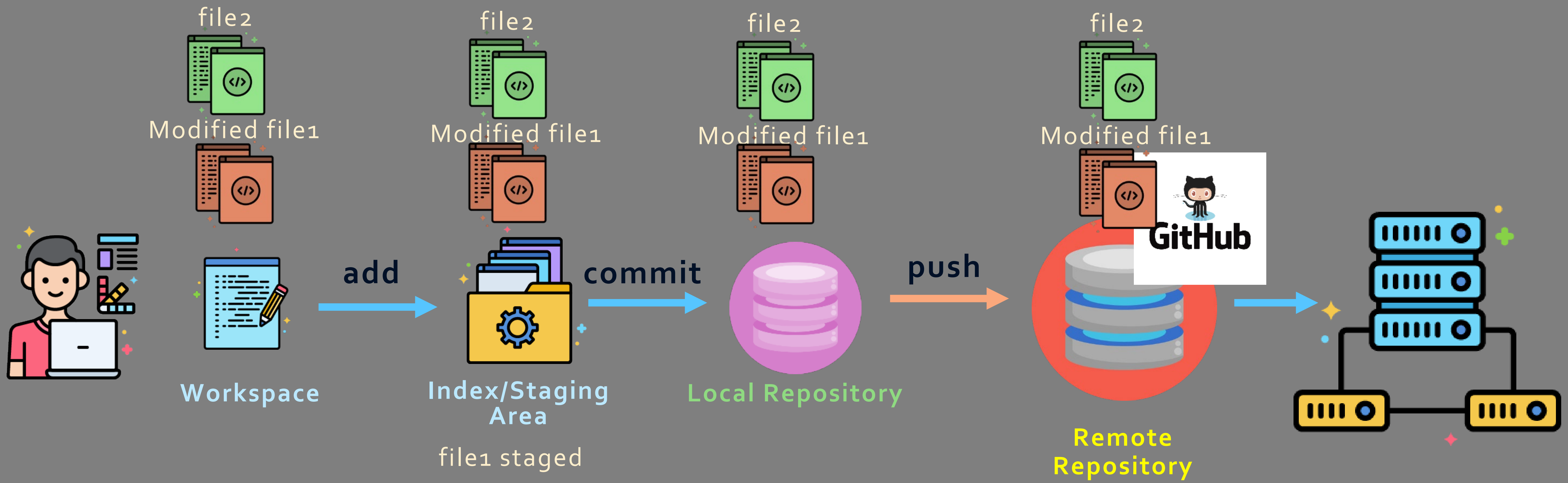
git commit –m "Removing file1"


file3
file2
Modified file1
file2
Modified file1
file2
Modified file1
file2
Modified file1
GitHub
add
commit
push
Workspace
Index/Staging Area
file1 staged
Local Repository
Remote Repository

git push

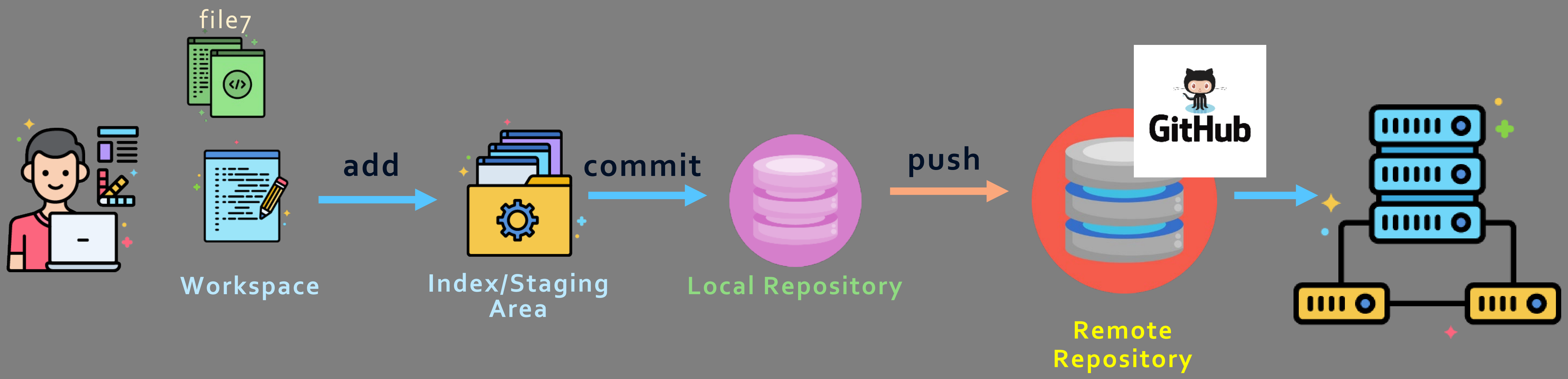

file2
Modified file1
file2
Modified file1
file2
Modified file1
file2
Modified file1
add
commit
push
GitHub
Workspace
Index/Staging Area
file1 staged
Local Repository
Remote Repository

git push



- Each commit will have separate ID, full snapshot with details for audit

Skipping the Staging Area

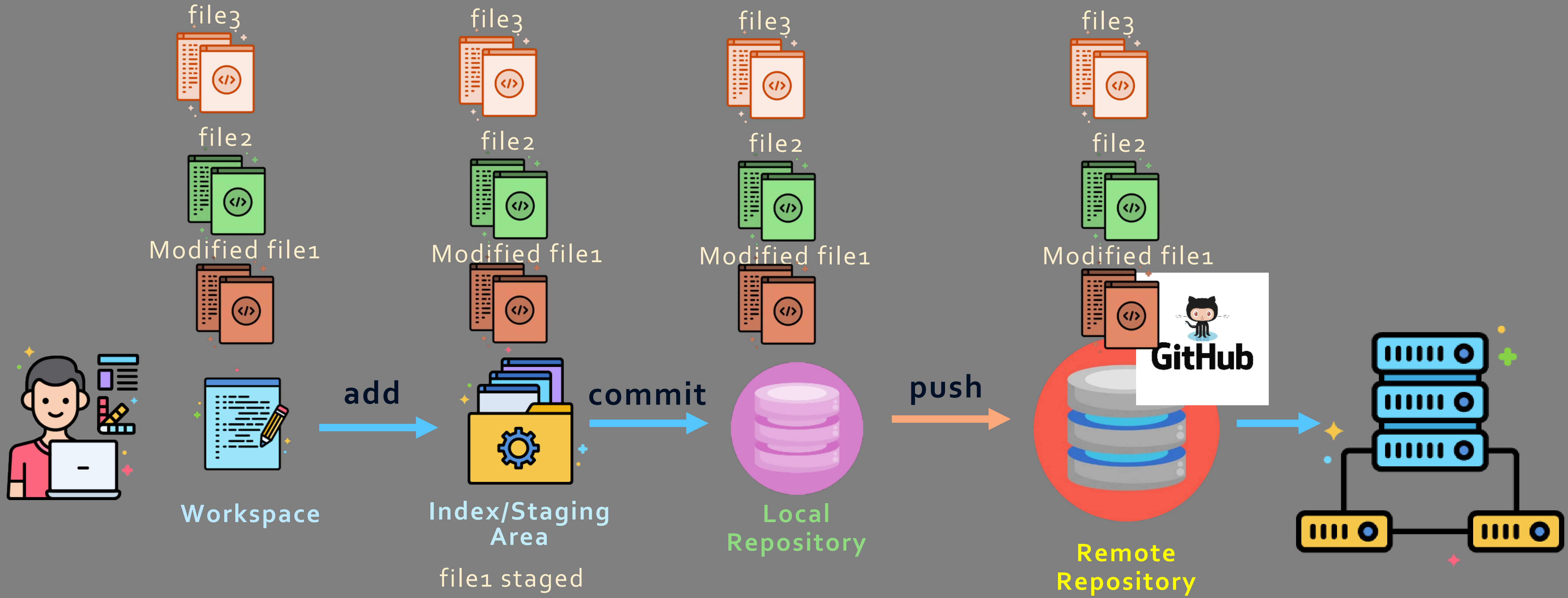


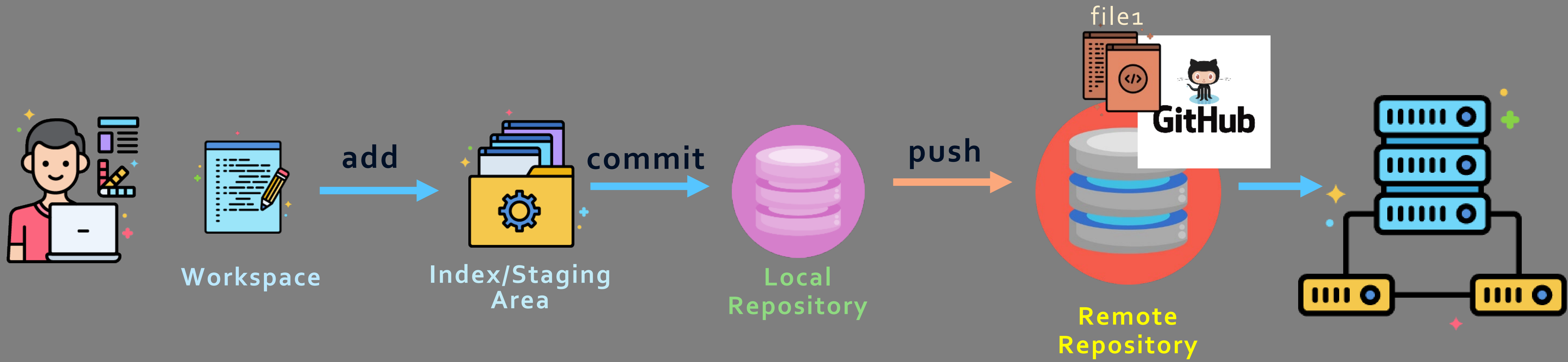
```
git commit –a –m "Adding file7 skipping stage"
git commit –am "Adding file7 skipping stage"
```

❑ NOT RECOMMENDED

Git Branch and Merge

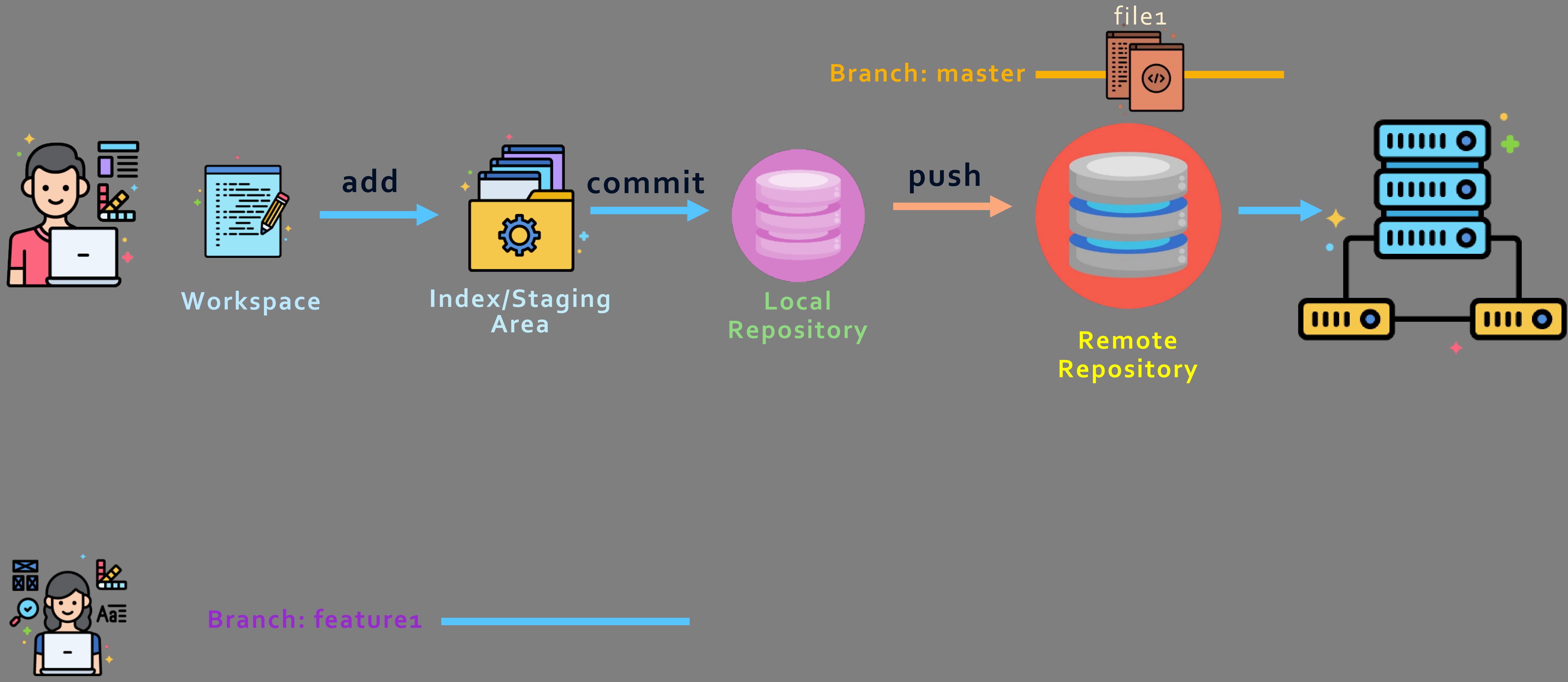
Raj Saha
cloudwithraj.com
Cloud With Raj

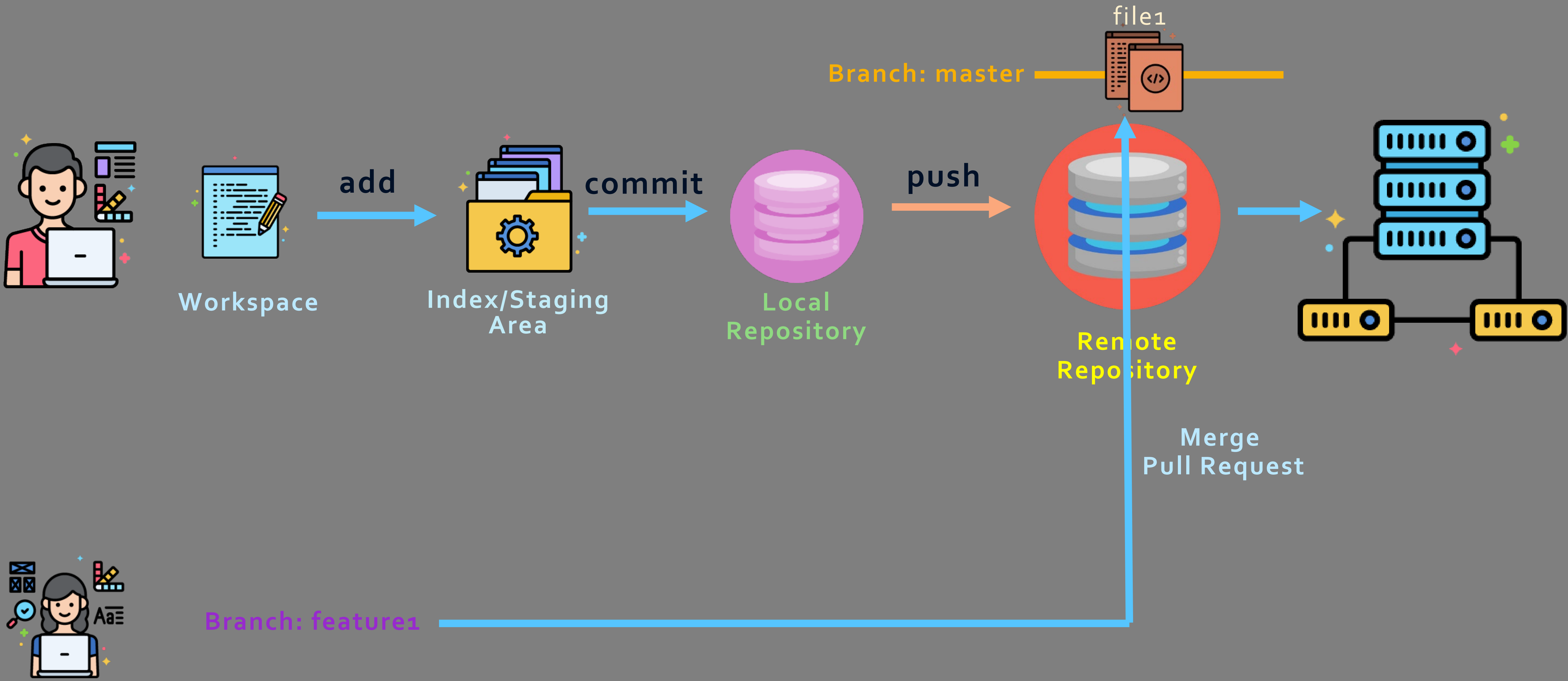

file3
file2
Modified file1
Workspace
add
file3
file2
Modified file1
Index/Staging Area
file1 staged
commit
file3
file2
Modified file1
Local Repository
push
file3
file2
Modified file1
GitHub
Remote Repository

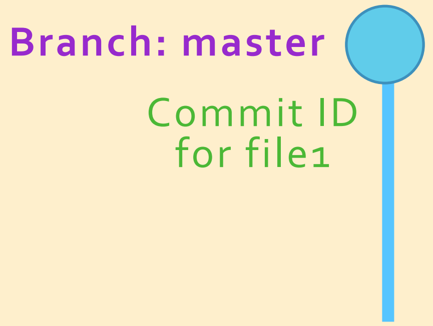


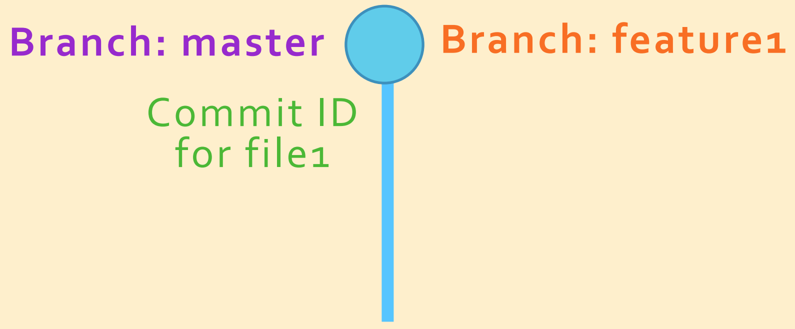
file1
GitHub
add
commit
push
Workspace
Index/Staging Area
Local Repository
Remote Repository

I am working on a new feature feature1


file1
Branch: master
add
commit
push
Workspace
Index/Staging Area
Local Repository
Remote Repository
Branch: feature1

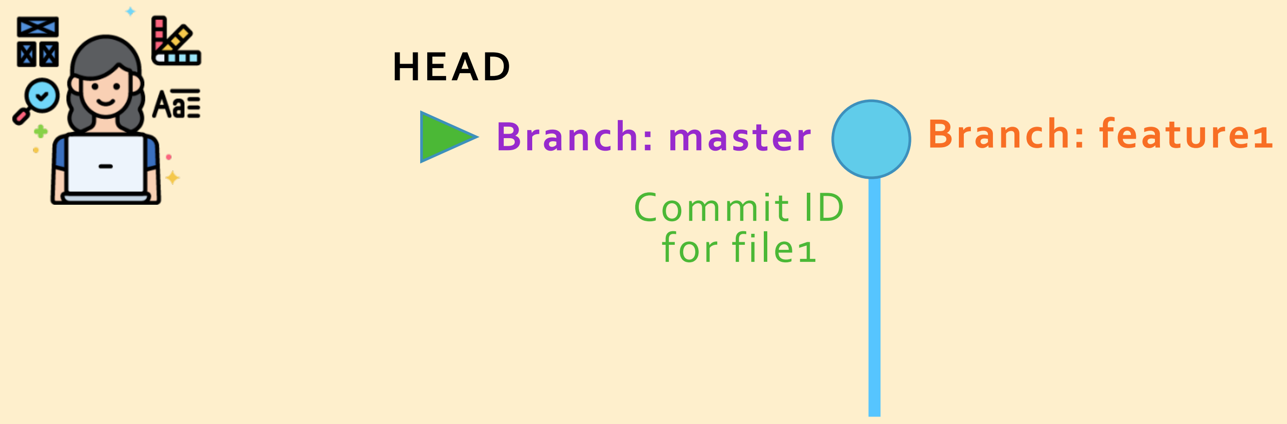

file1
Branch: master
add
commit
push
Workspace
Index/Staging Area
Local Repository
Remote Repository
Merge
Pull Request
Branch: feature1


Branch: master
Commit ID
for file1

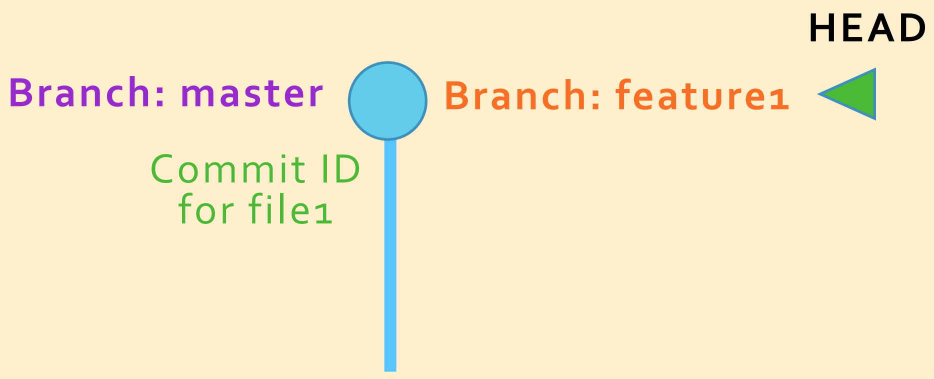


git branch feature1

- Branch can only be created from existing branch
- Important - Branches are references to commit, no code is copied



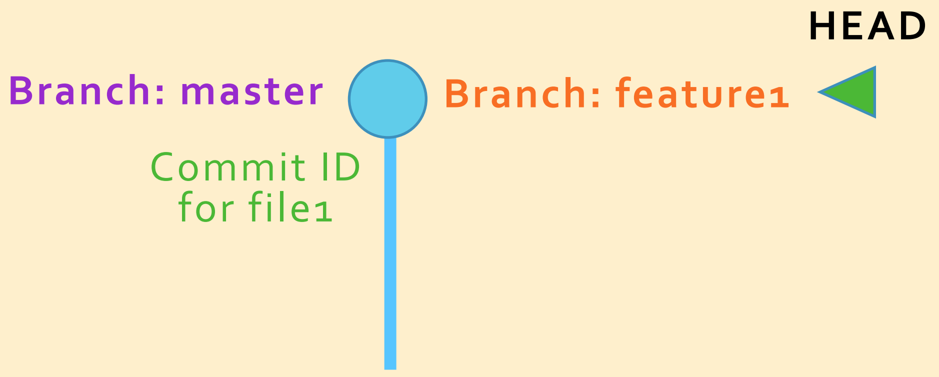
- HEAD determines which branch you are at currently



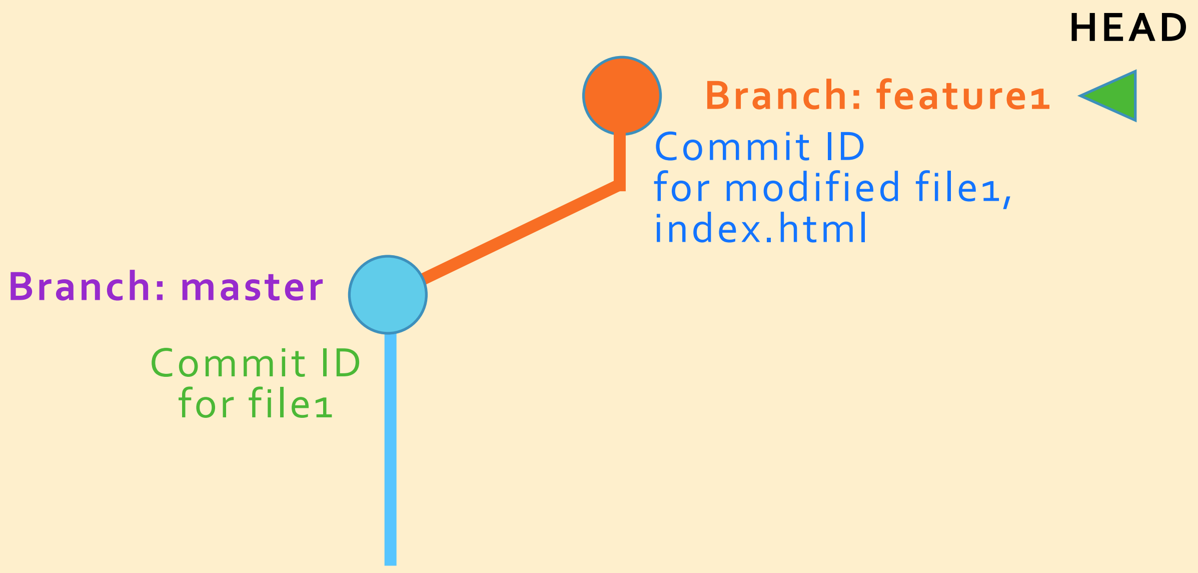
```
git switch feature1

git checkout feature1
```

Modified file1

Added index.html



- ✓ Switch Branch to feature1
- ✓ Modified file1, added index.html
- ✓ Commit to feature1 Branch
- ✓ All of this is in local repo



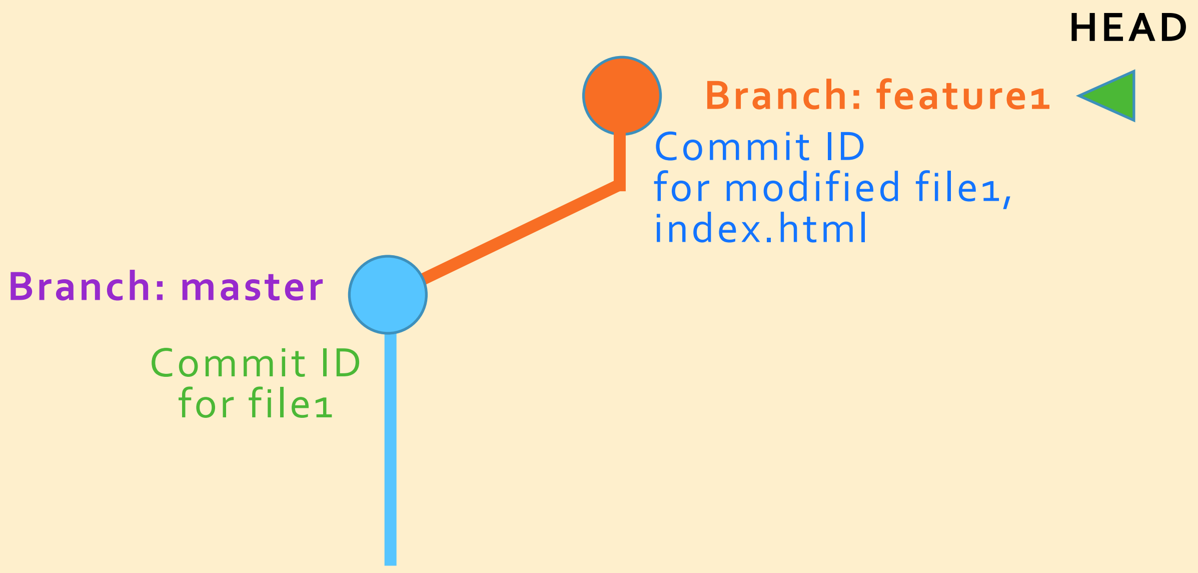
Modified file1

Added index.html

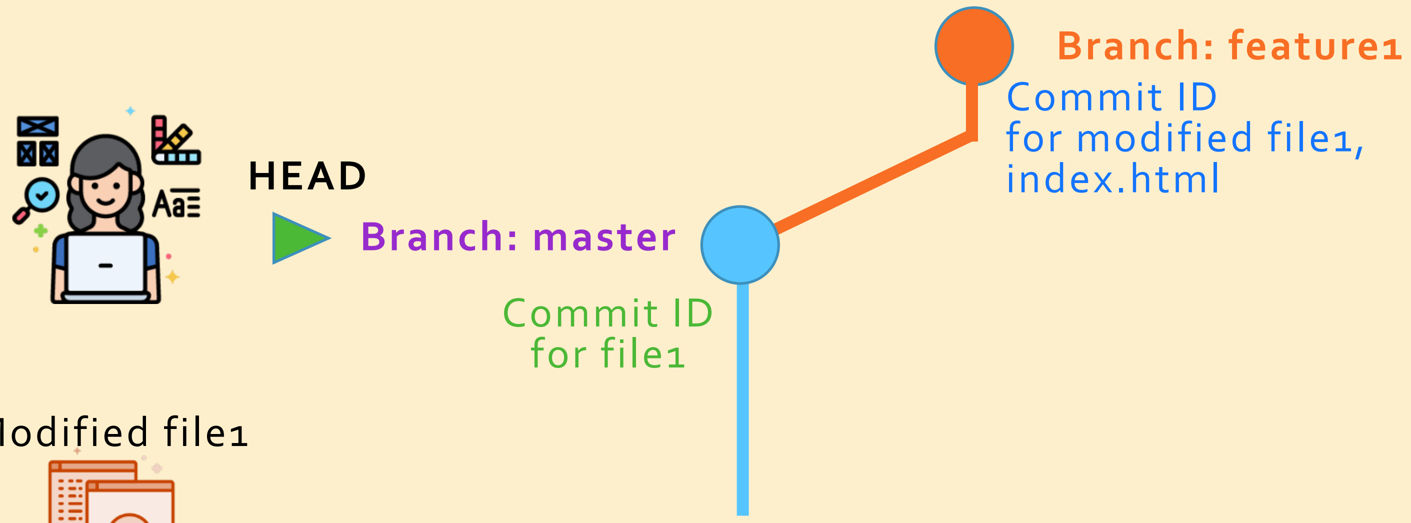
- ✓ Switch Branch to feature1
- ✓ Modified file1, added index.html
- ✓ Commit to feature1 Branch
- ✓ All of this is in local repo

Modified file1

Added index.html



git switch master



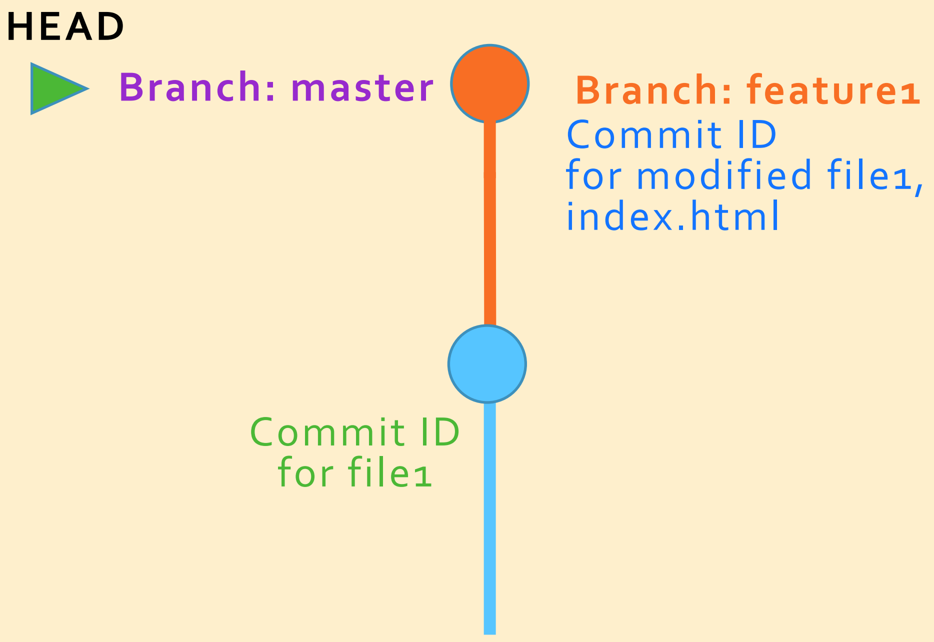
Modified file1

Added index.html

git switch master

git merge feature1

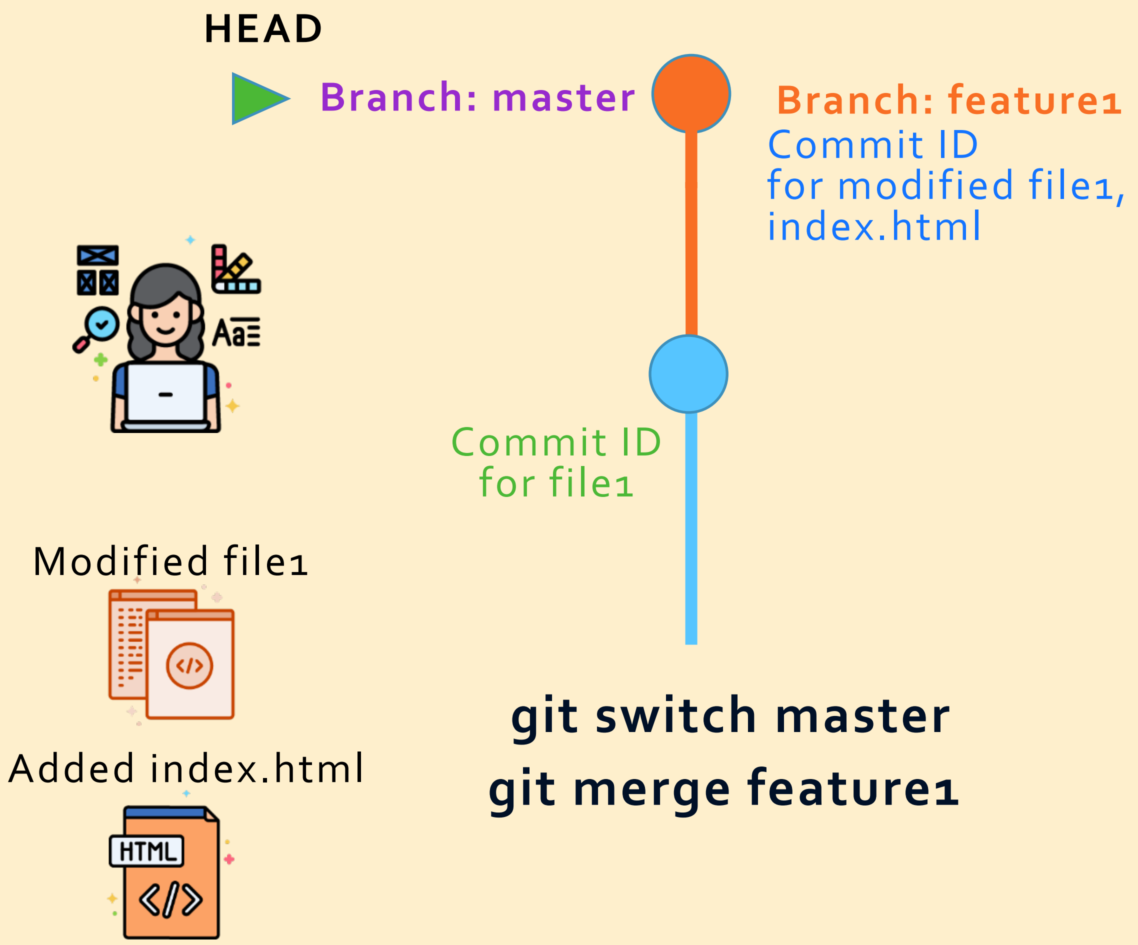
git merge <name of the branch to be merged into the branch you are on>



Modified file1

Added index.html

```
git switch master
git merge feature1
```



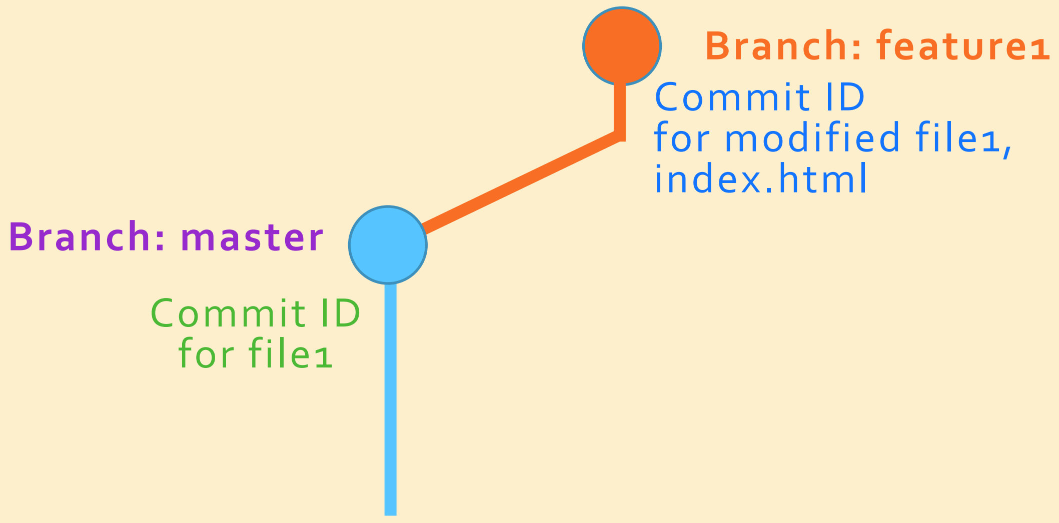
git switch master
git merge feature1

- Making master branch move to look at latest commit ID
- Fast forward merge
- NO CHANGES made on master branch

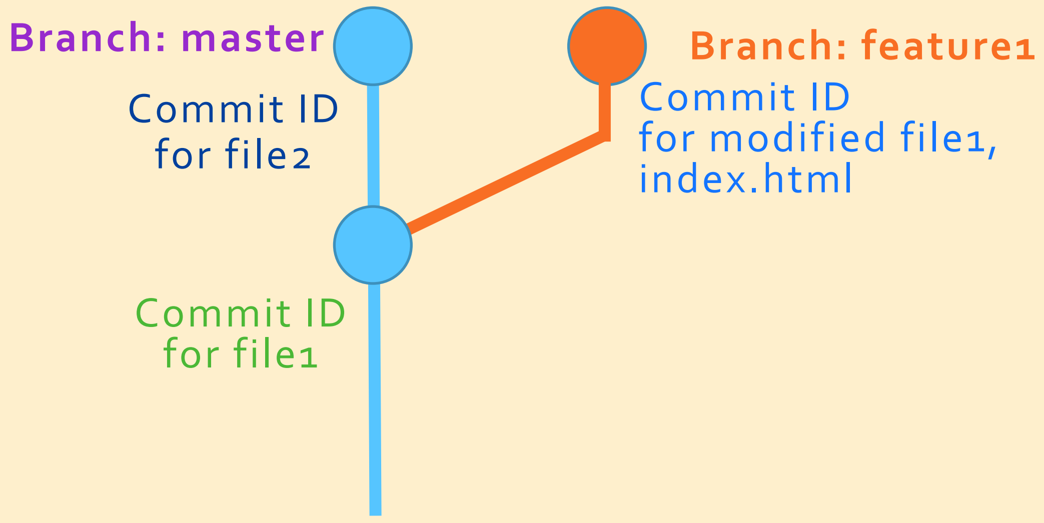
Changes in Multiple Branches

Modified file1

Added index.html

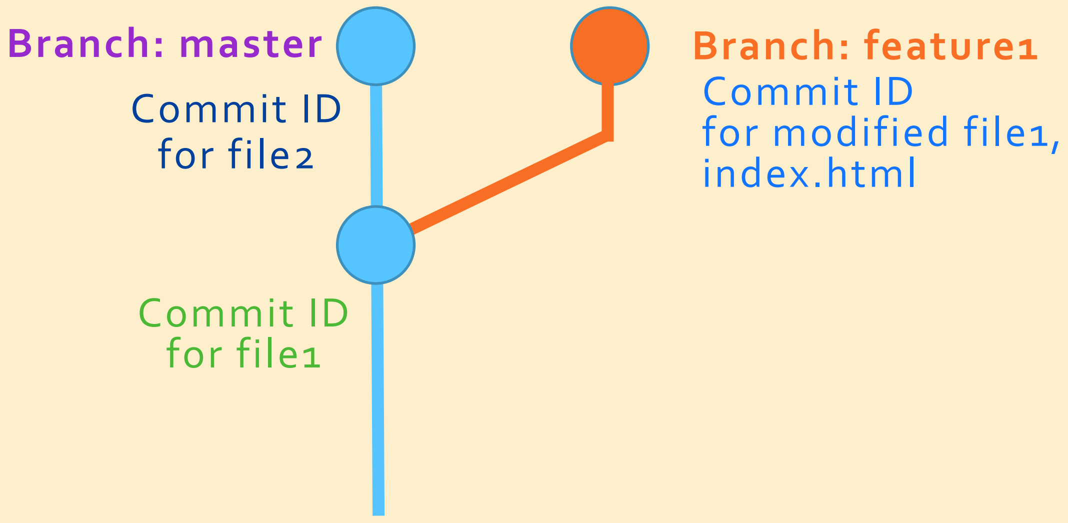


Changes in Multiple Branches



Committed file2
in master

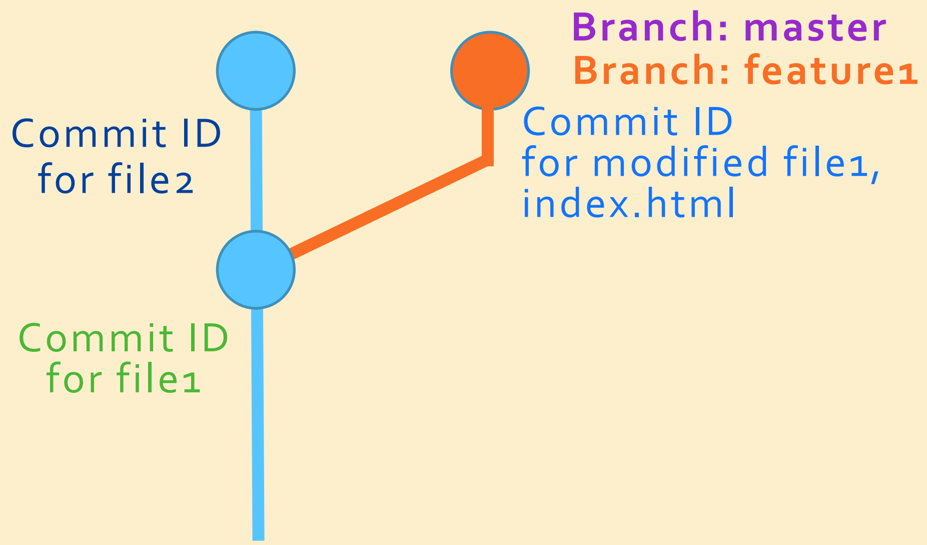
Changes in Multiple Branches



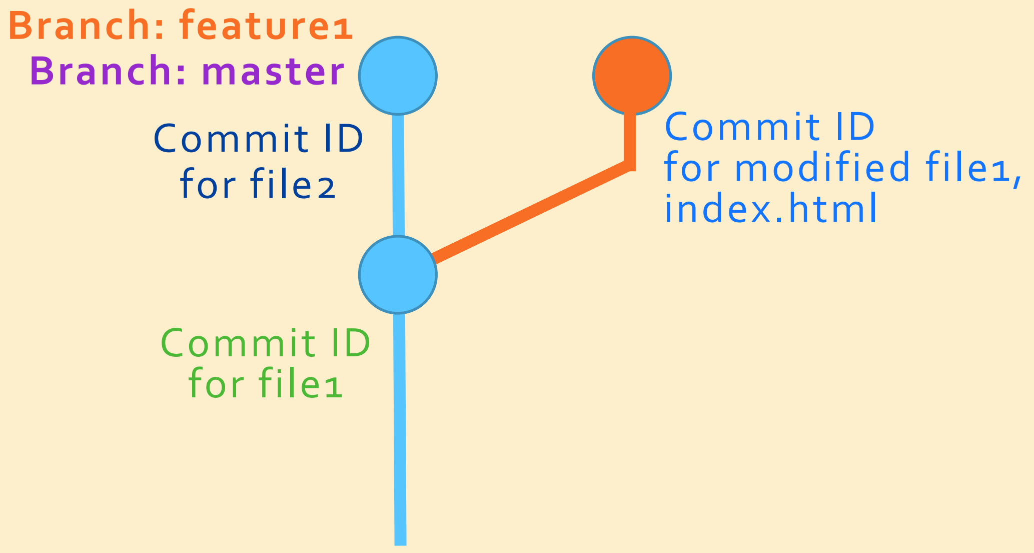
Committed file2 in master

- Changes made on both branches
- Just moving the branch to look at new commit ID won't do merge
- Need three-way merge

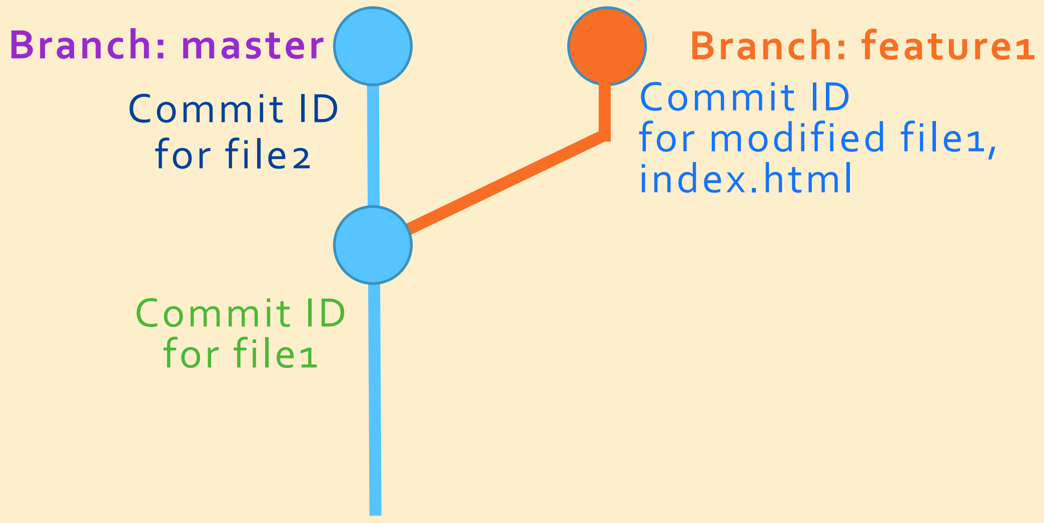
Fast Forward Merge Not Possible



Fast Forward Merge Not Possible

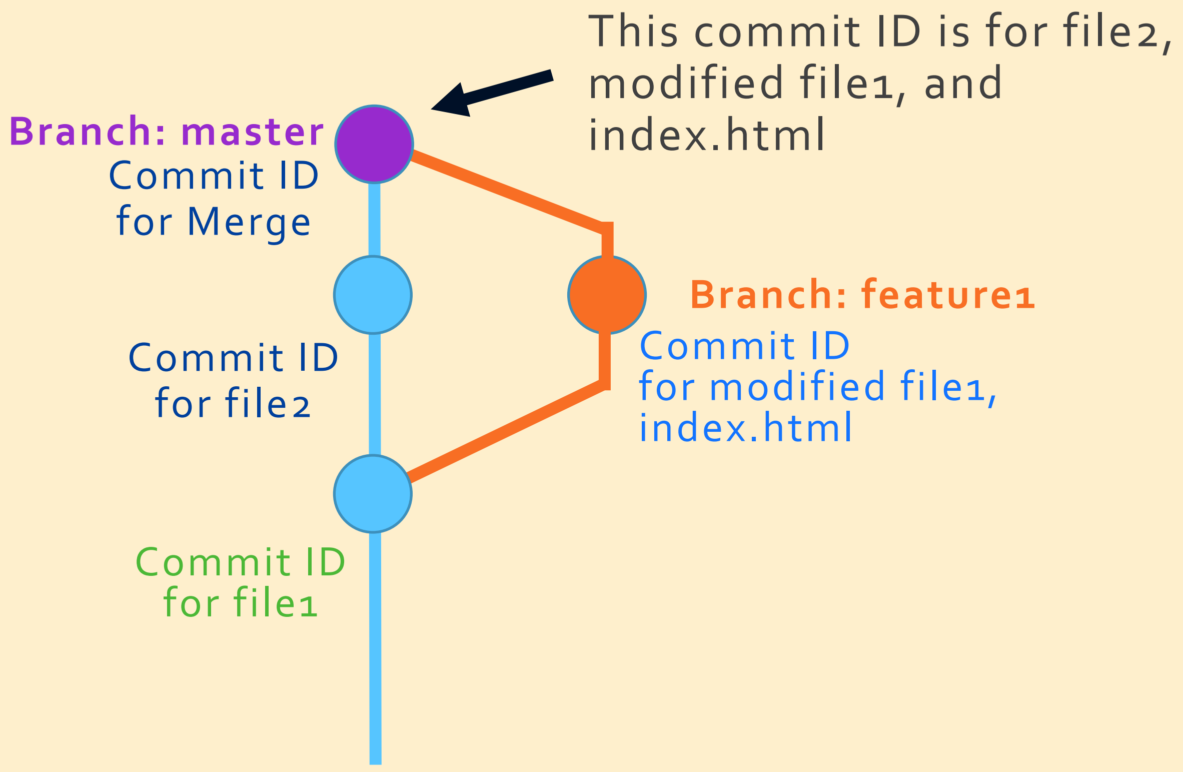


Recursive Three-Way Merge



```
git switch master
git merge feature1
```

Recursive Three-Way Merge

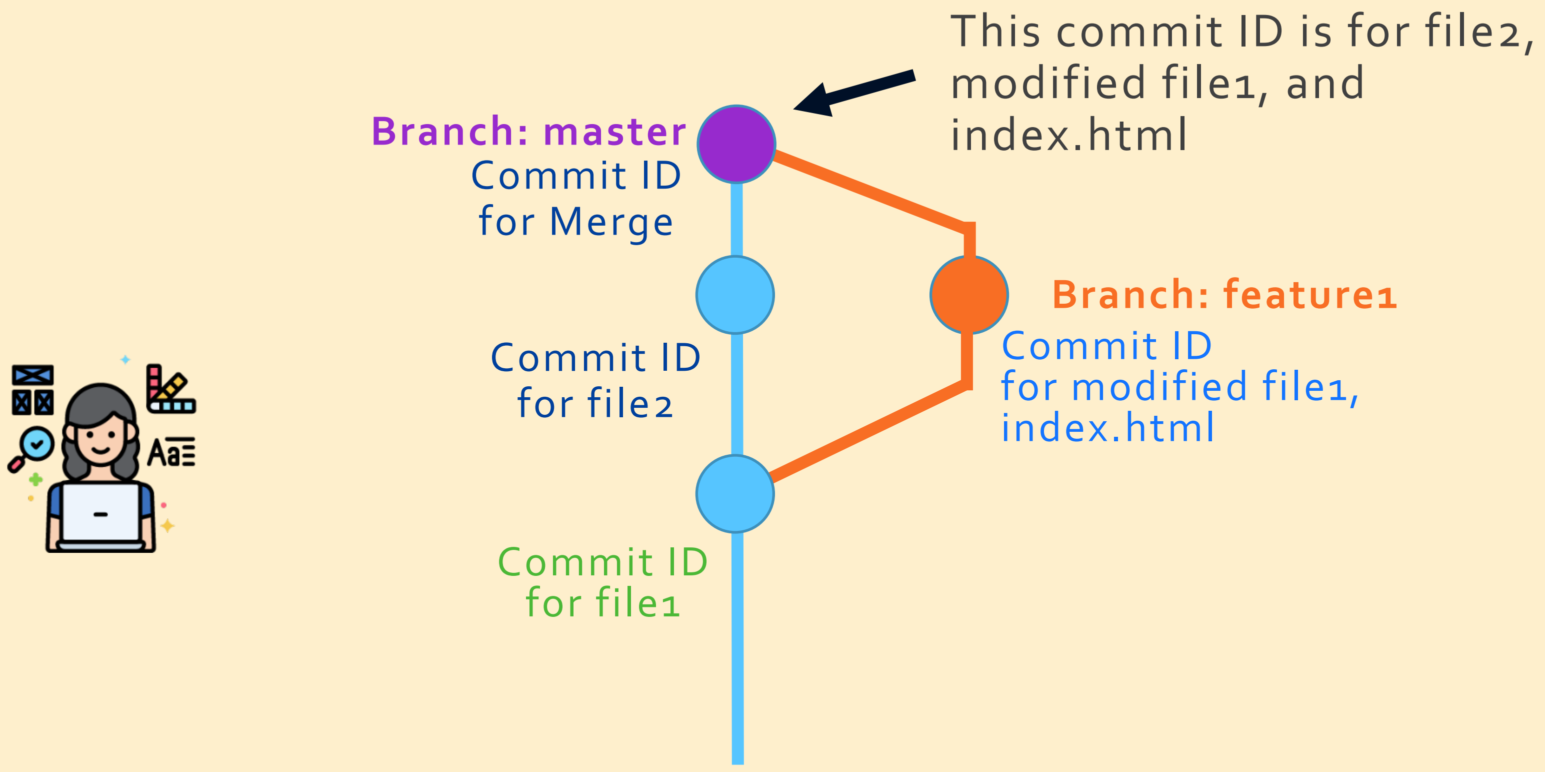

This commit ID is for file2, modified file1, and index.html
Branch: master
Commit ID for Merge
Commit ID for file2
Commit ID for file1
Branch: feature1
Commit ID for modified file1, index.html

git switch master

git merge feature1

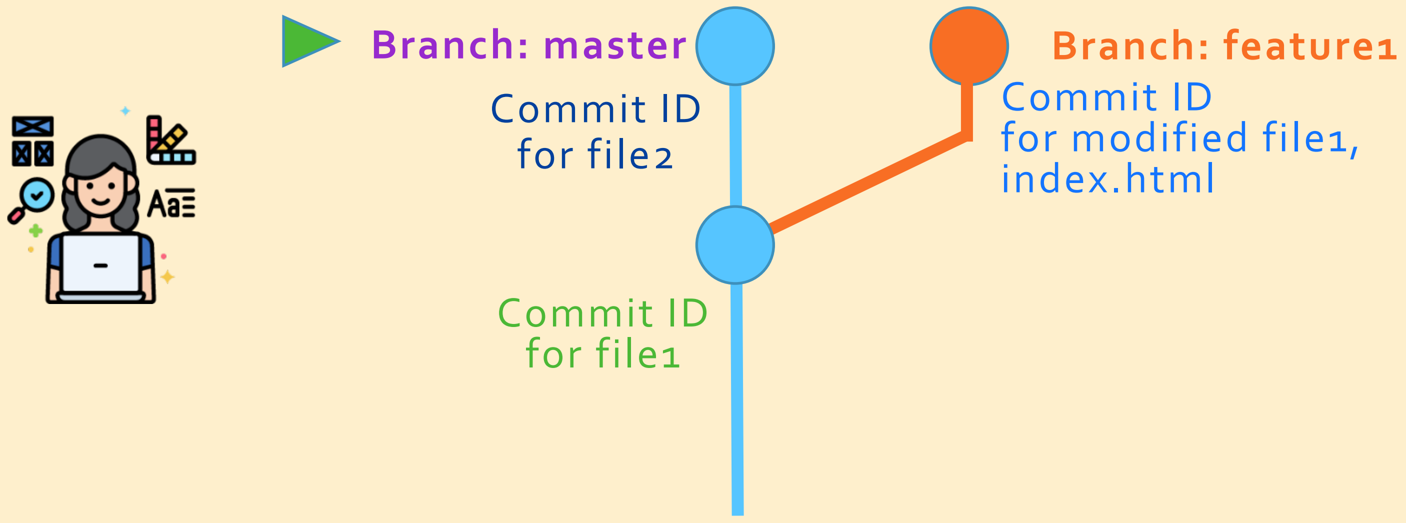
- ✓ feature1 merged into master
- ✓ Git will choose fast forward or three-way merge automatically

Delete Branch after Merge


This commit ID is for file2, modified file1, and index.html
Branch: master
Commit ID for Merge
Branch: feature1
Commit ID for modified file1, index.html
Commit ID for file2
Commit ID for file1

git branch –d feature1

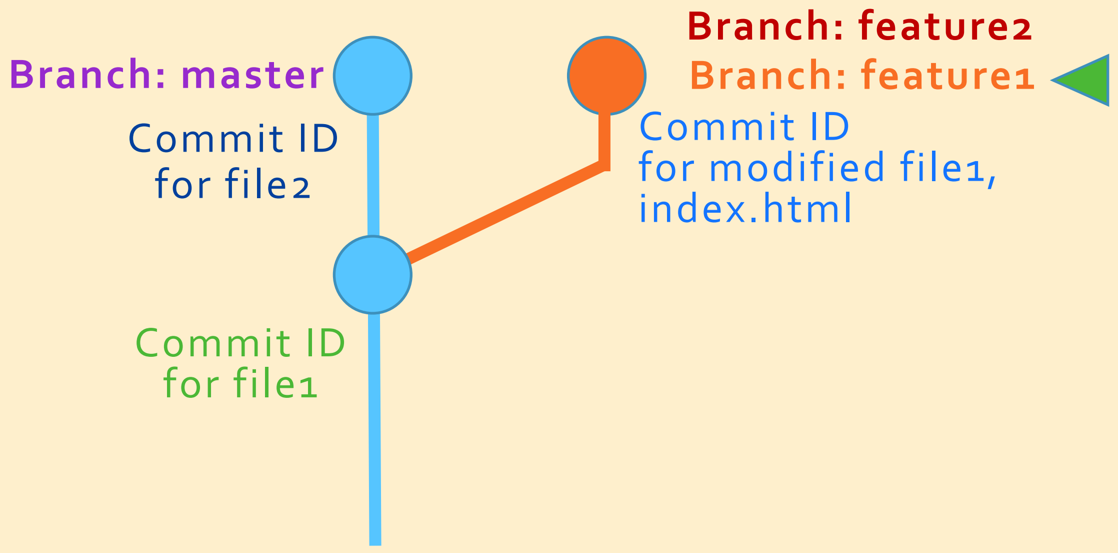
Quiz!!



```
git switch feature1
git branch feature2
```

- Which CommitID does feature2 branch look at?
- Where does the HEAD go?

Quiz Answer!!

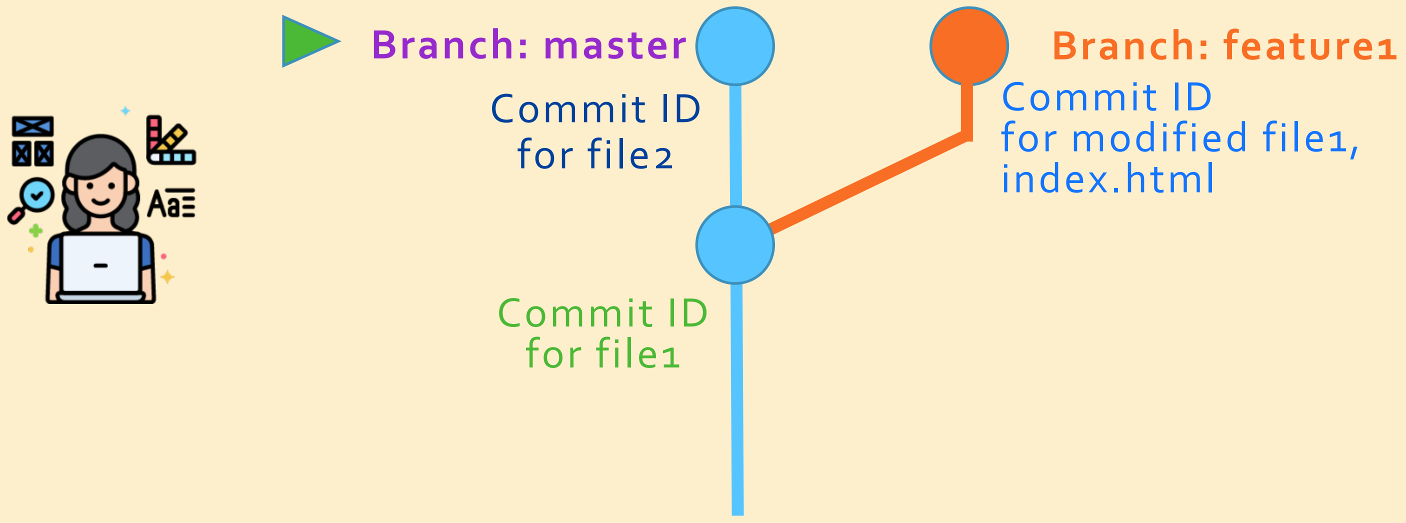


git switch feature1

git branch feature2

- Branches are just reference to CommitID
- New branch does NOT change HEAD unless switched

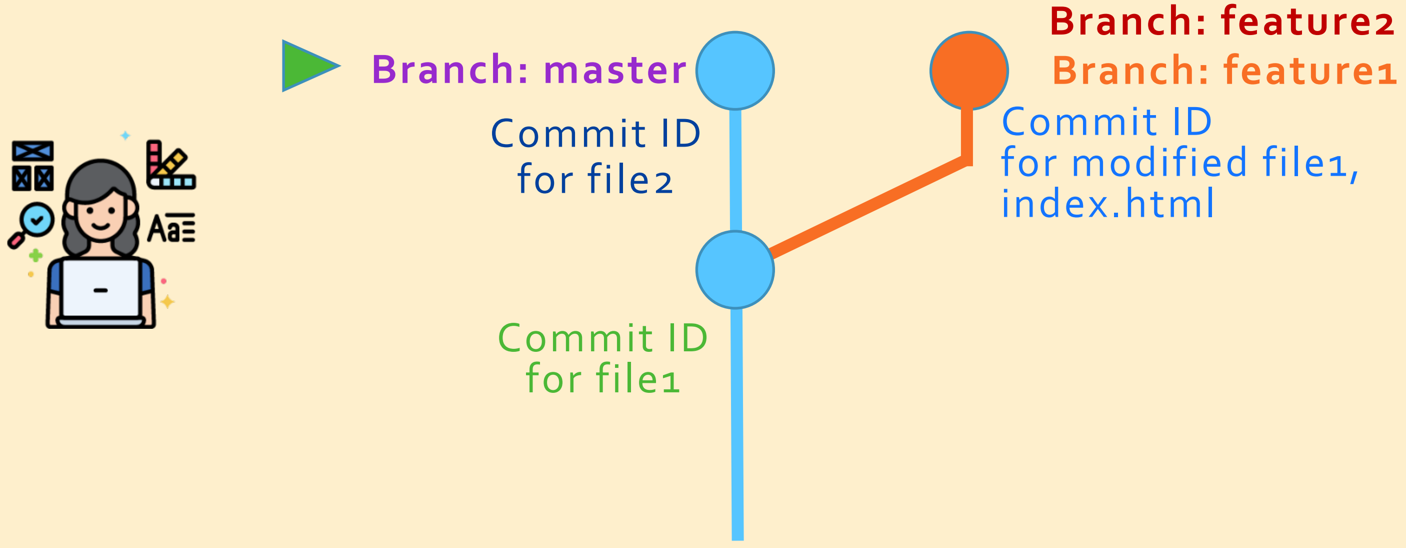
Alternate Method



git branch feature2 feature1

git branch <new branch> <old branch to reference>

Alternate Method



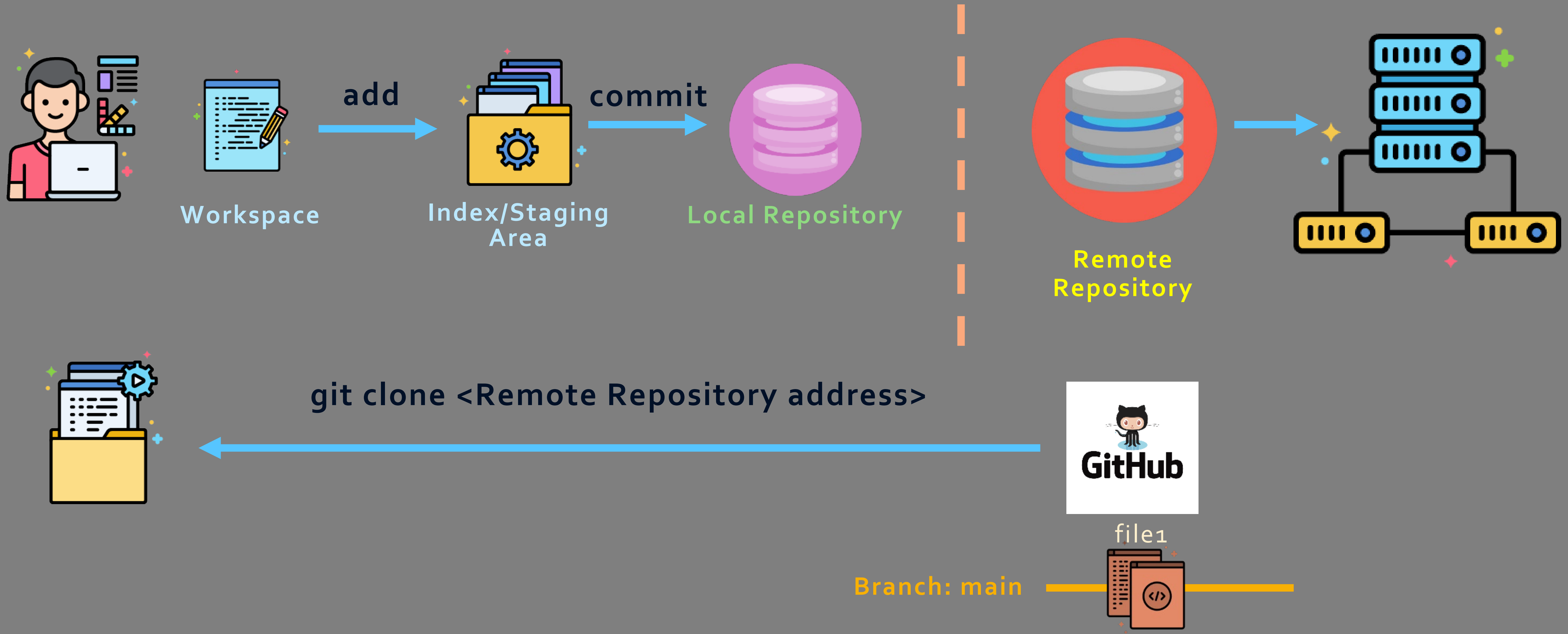
git branch feature2 feature1

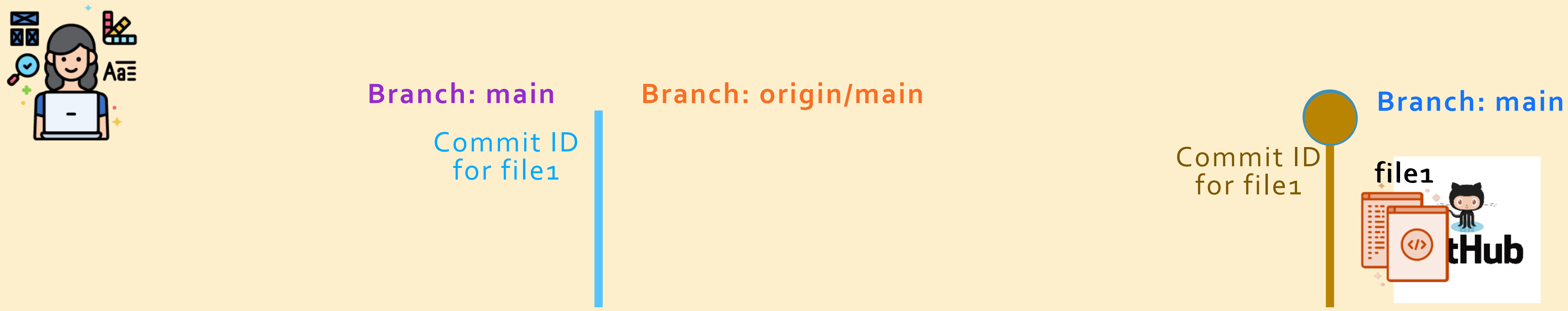
git branch <new branch> <old branch to reference>

Clone, Remote Branches, Fetch, Pull *(Yes, it is a packed lecture!!)*

Raj Saha
cloudwithraj.com
Cloud With Raj

GitHub to Local

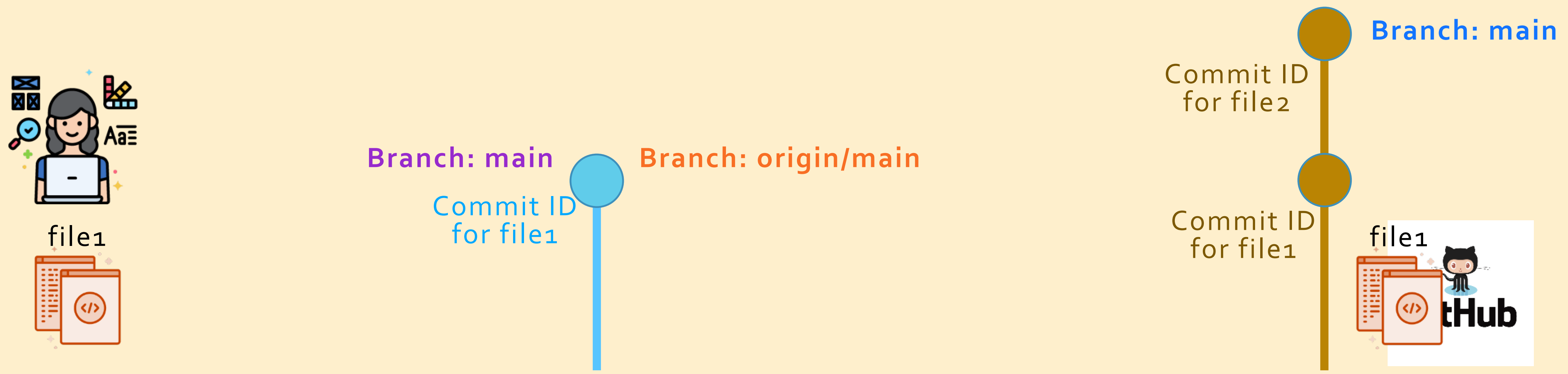




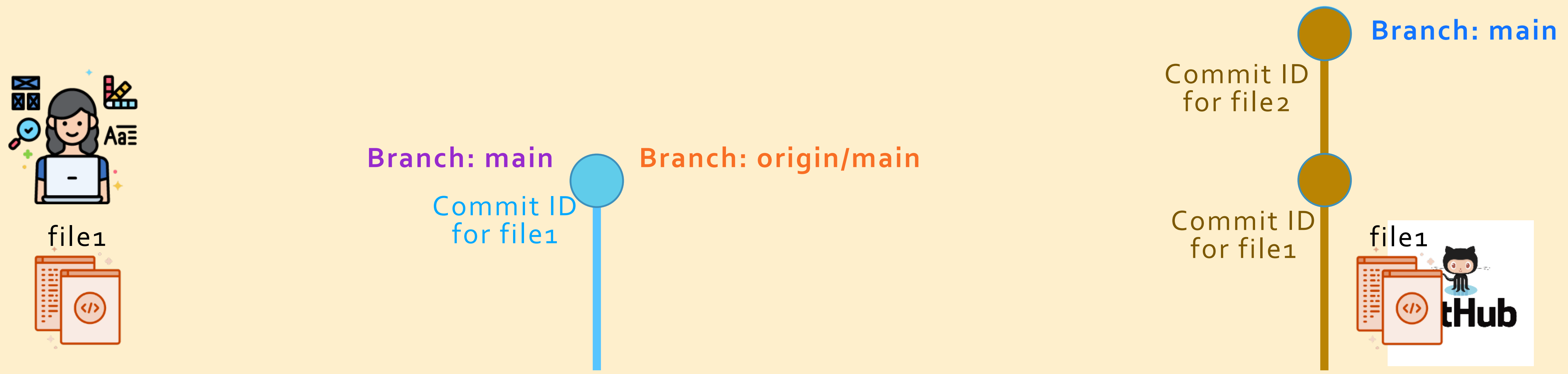
git clone
git branch -r

- ➢ origin/* branch is for remote tracking only, can NOT switch or commit in local

GitHub(Remote) Out of Sync with Local

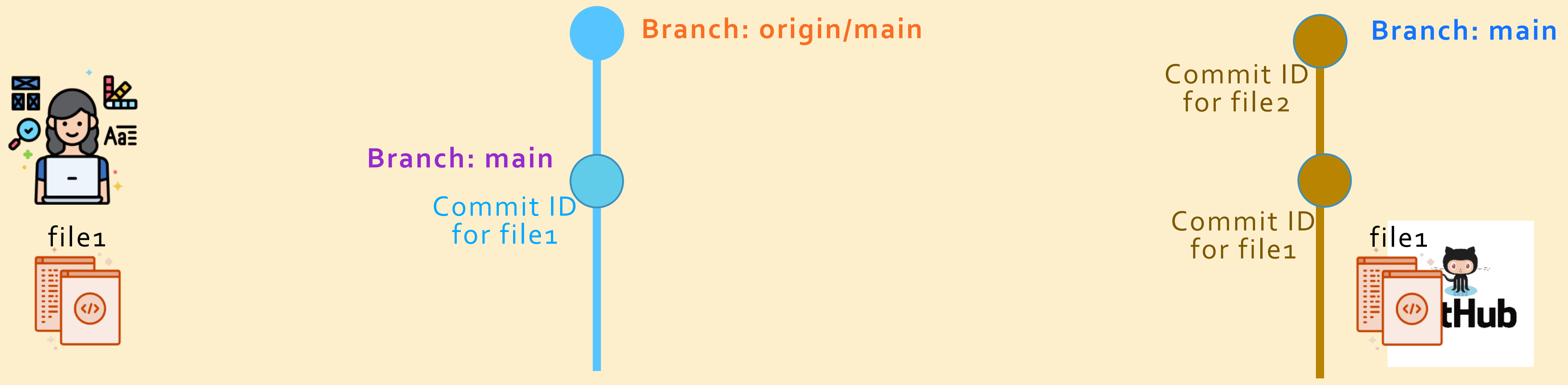


GitHub(Remote) Out of Sync with Local



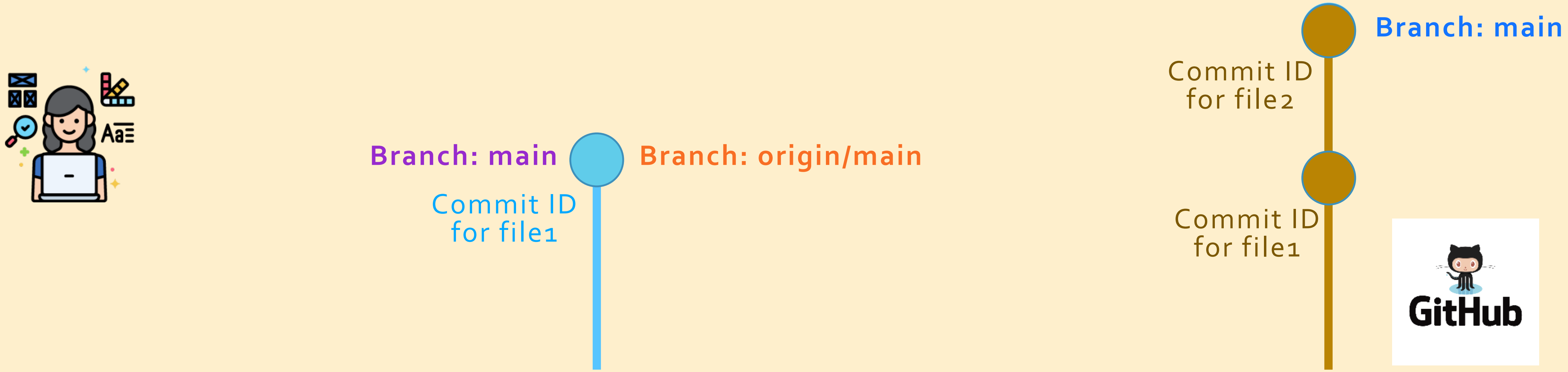
git fetch

Fast Forward Merge

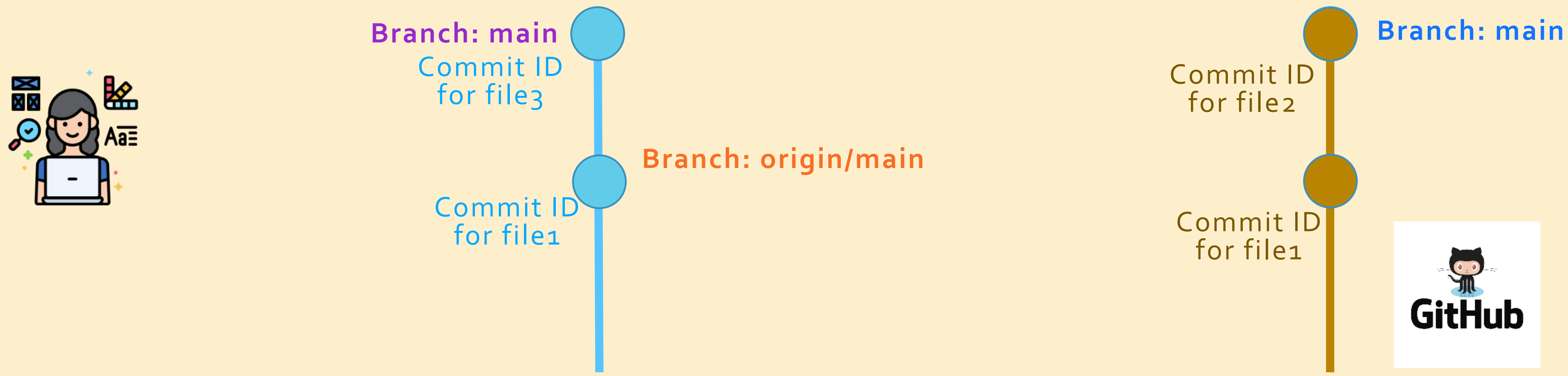


git merge origin/main

Changes in Local and Remote

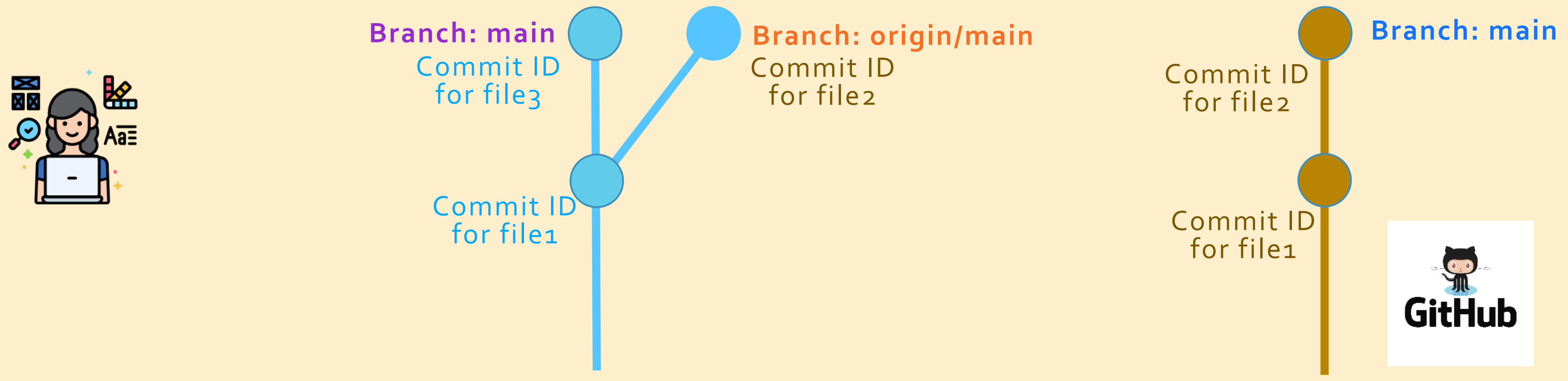


Changes in Local and Remote



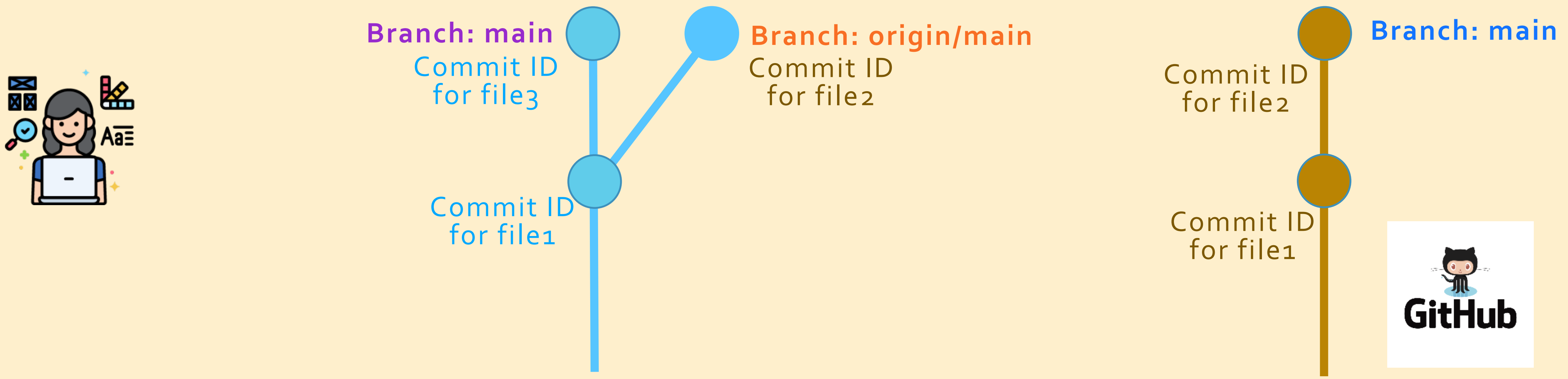
git fetch

Changes in Local and Remote



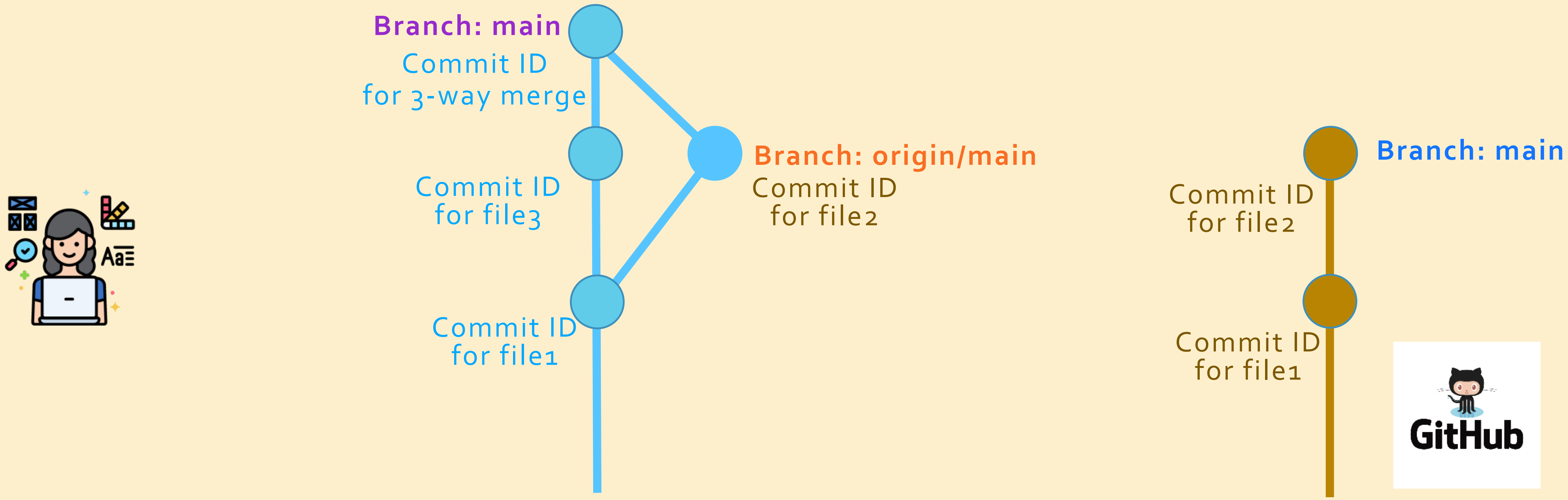
git fetch

Changes in Local and Remote



git fetch
git merge origin/main

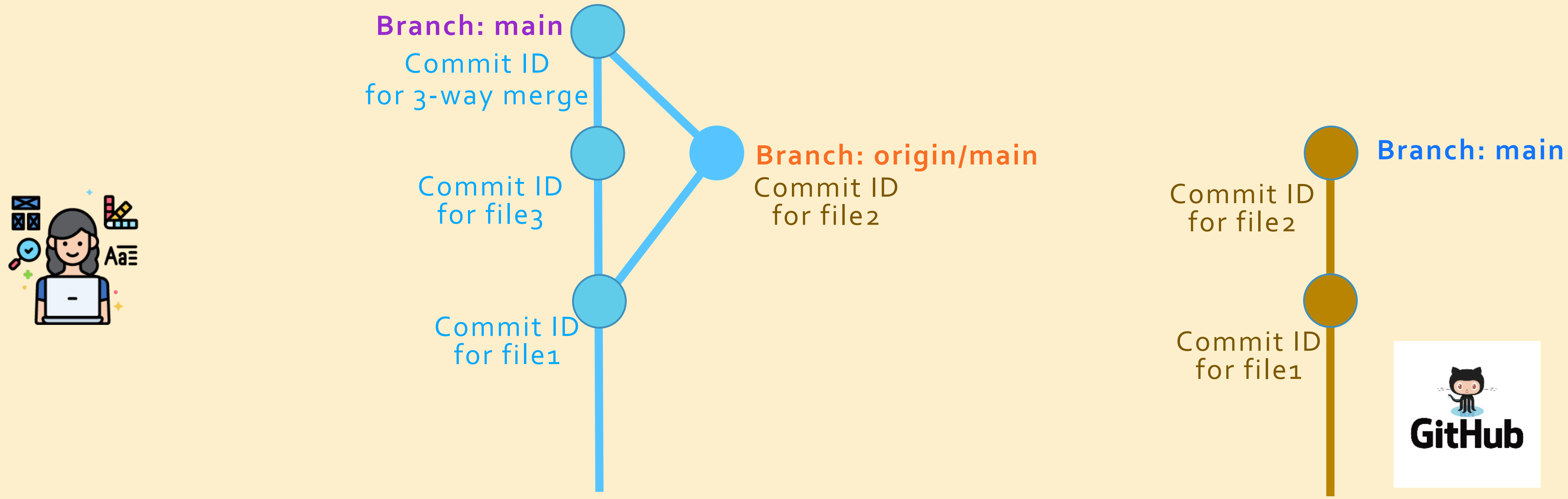
Three Way Merge



git fetch
git merge origin/main

git pull
(NOT = Pull Request)

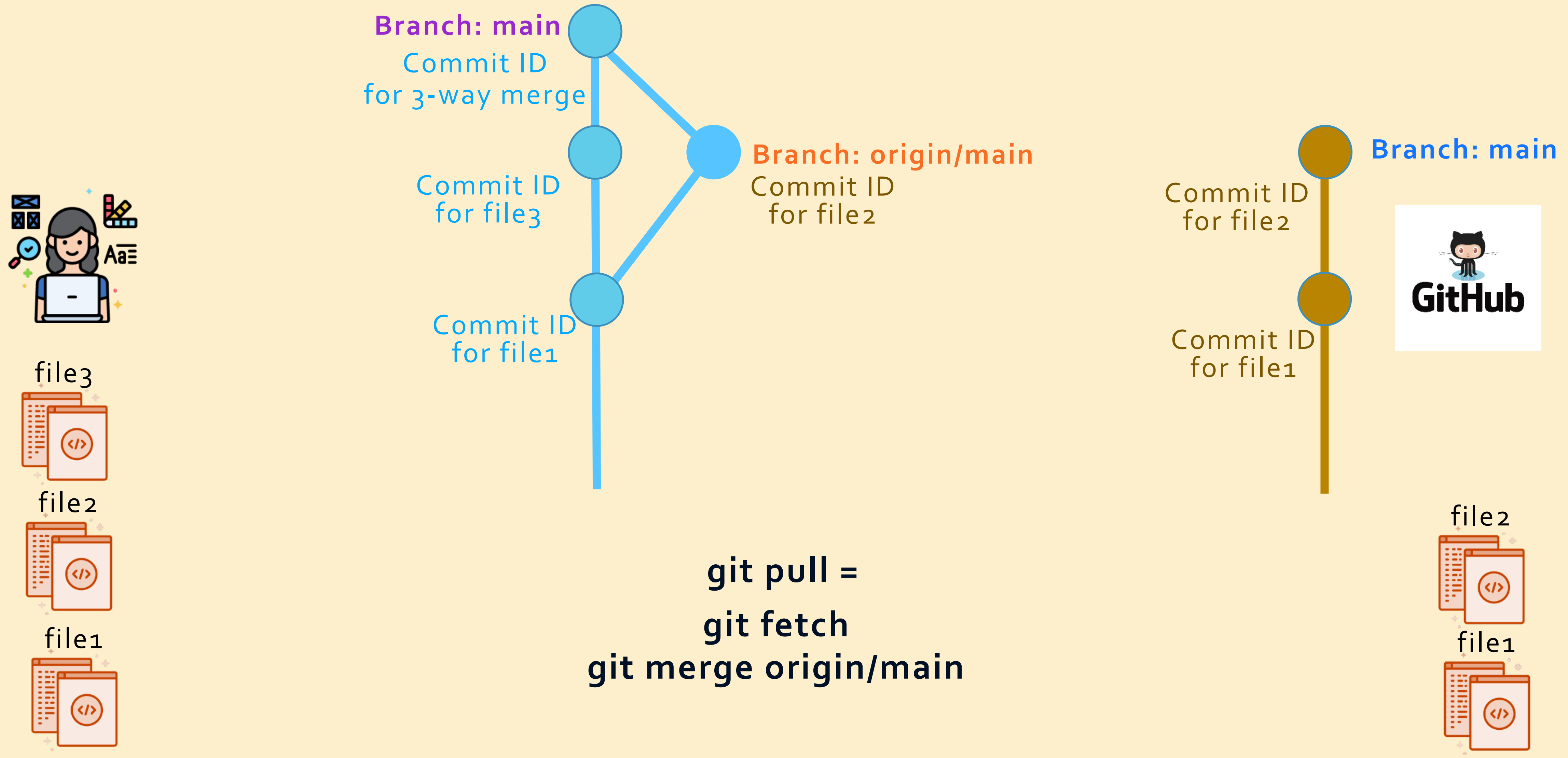
git pull = git fetch + git merge



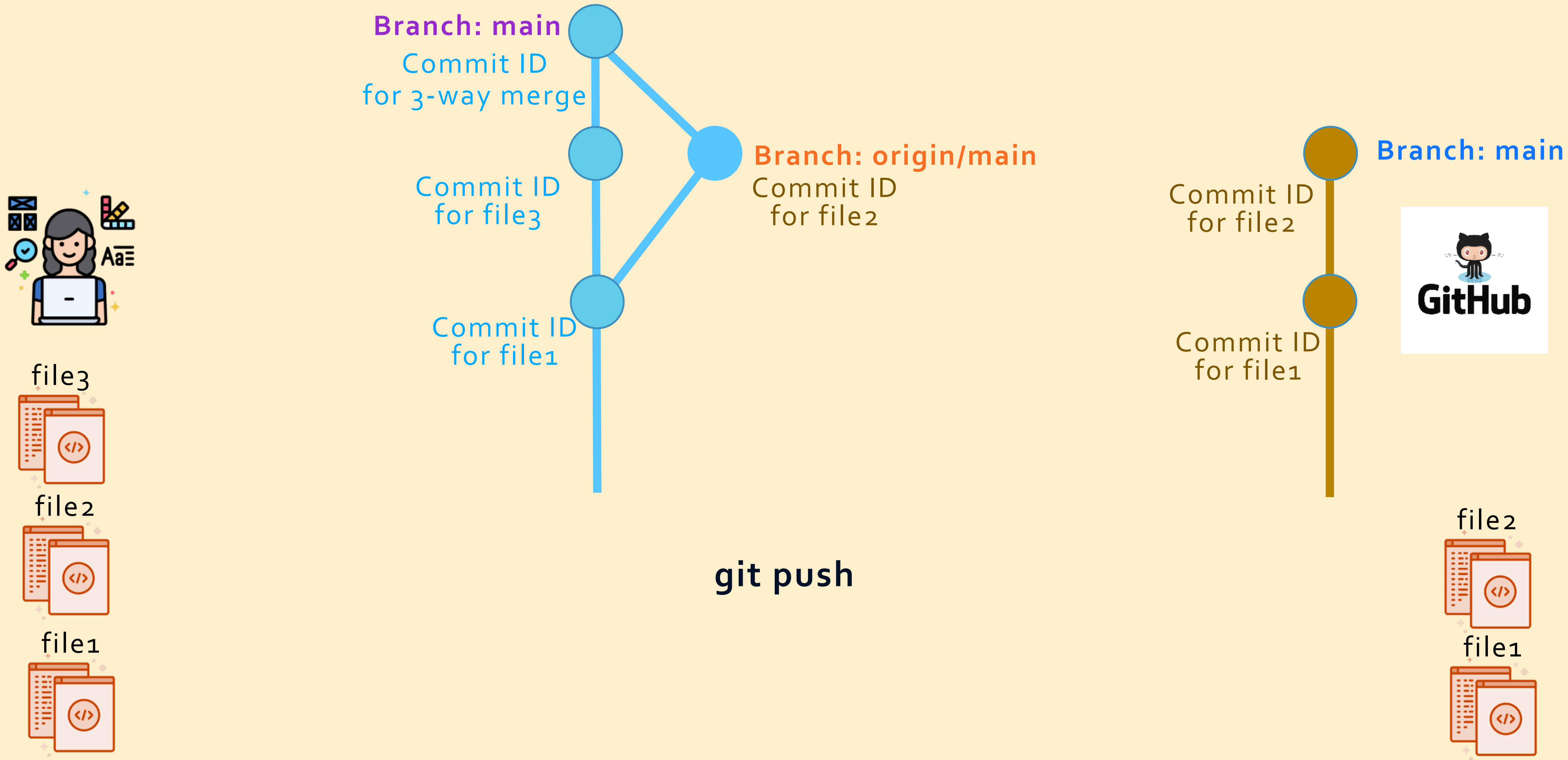
git pull =

git fetch
git merge origin/main

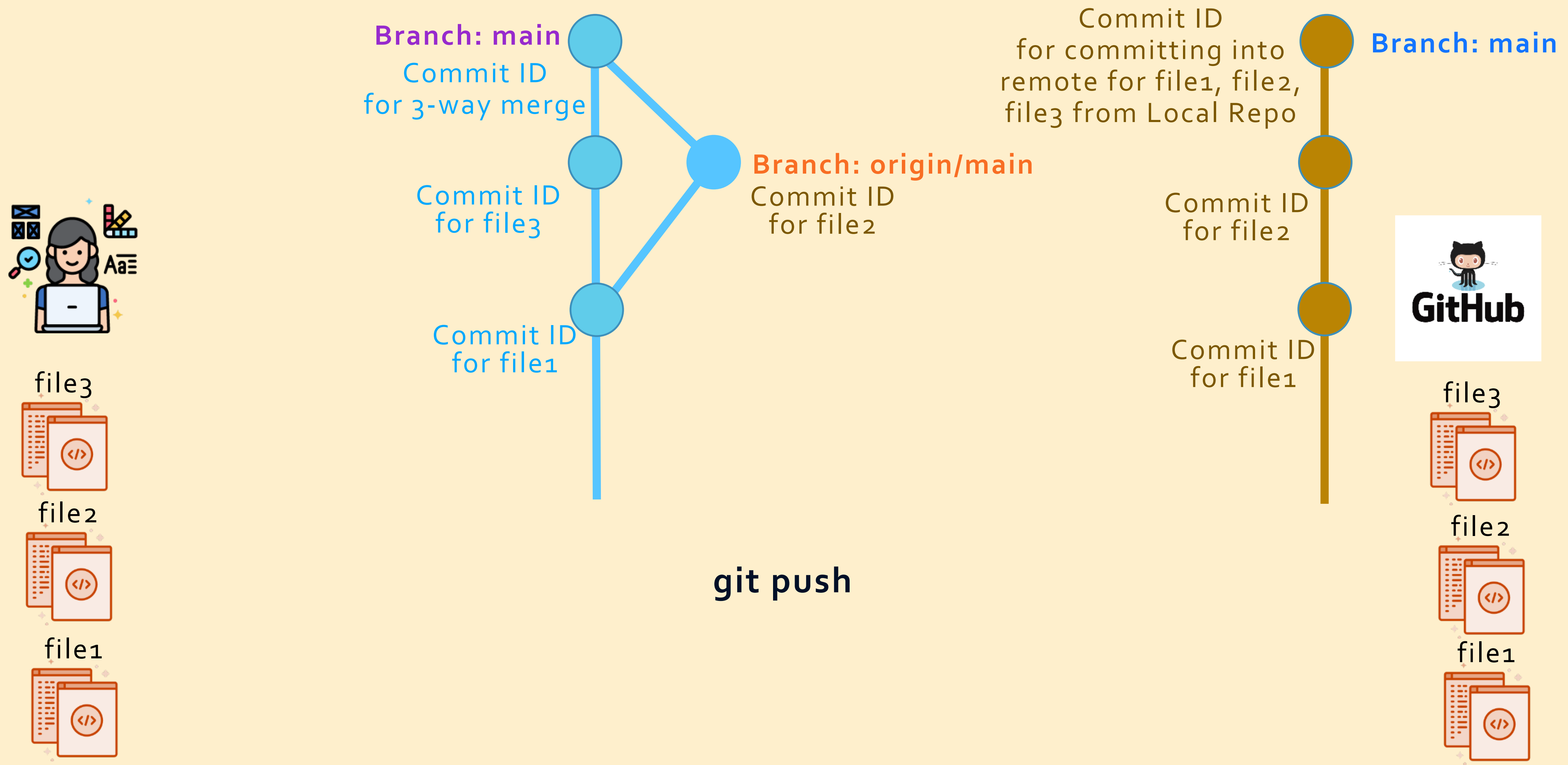
All in Local!
Remember git push?



```
git pull =

git fetch
git merge origin/main
```

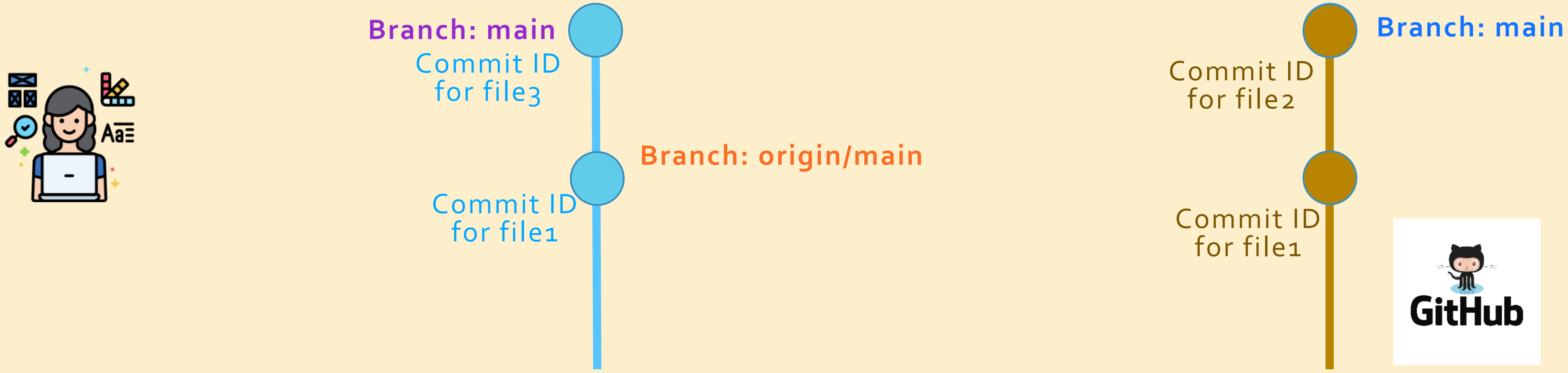

Branch: main
Commit ID
for 3-way merge
Commit ID
for file3
Commit ID
for file1
Branch: origin/main
Commit ID
for file2
file3
file2
file1
Branch: main
Commit ID
for file2
Commit ID
for file1
GitHub
file2
file1

git push


Branch: main
Commit ID
for 3-way merge
Commit ID
for file3
Commit ID
for file1
Branch: origin/main
Commit ID
for file2
Commit ID
for committing into
remote for file1, file2,
file3 from Local Repo
Branch: main
Commit ID
for file2
Commit ID
for file1
GitHub
file3
file2
file1
file3
file2
file1

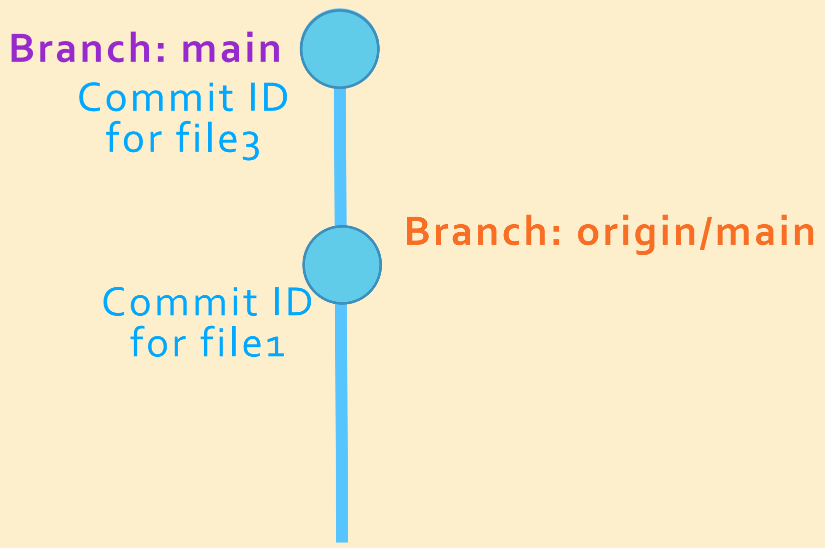
git push

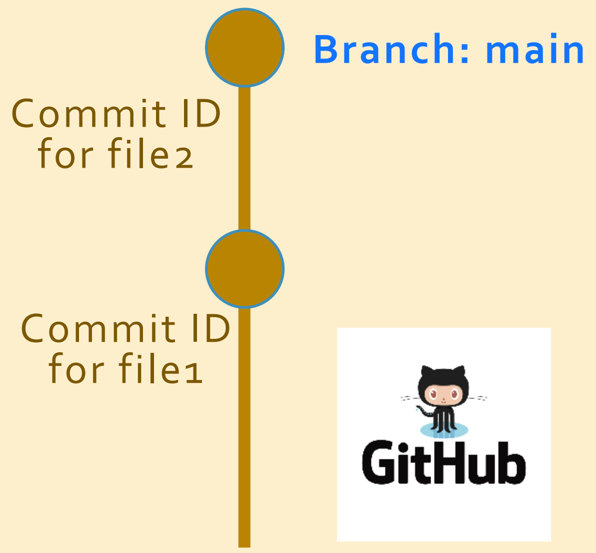
What if You Don't fetch/pull



git push –f

What if You Don't fetch/pull





git push –f

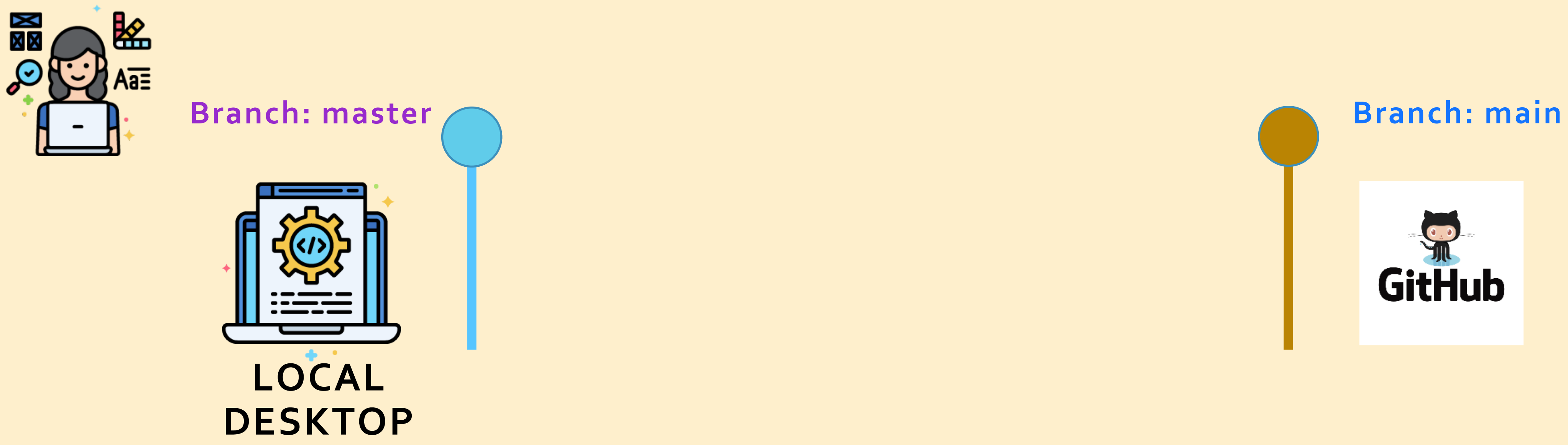
Master or Main??!!

Raj Saha
cloudwithraj.com
▶ Cloud With Raj

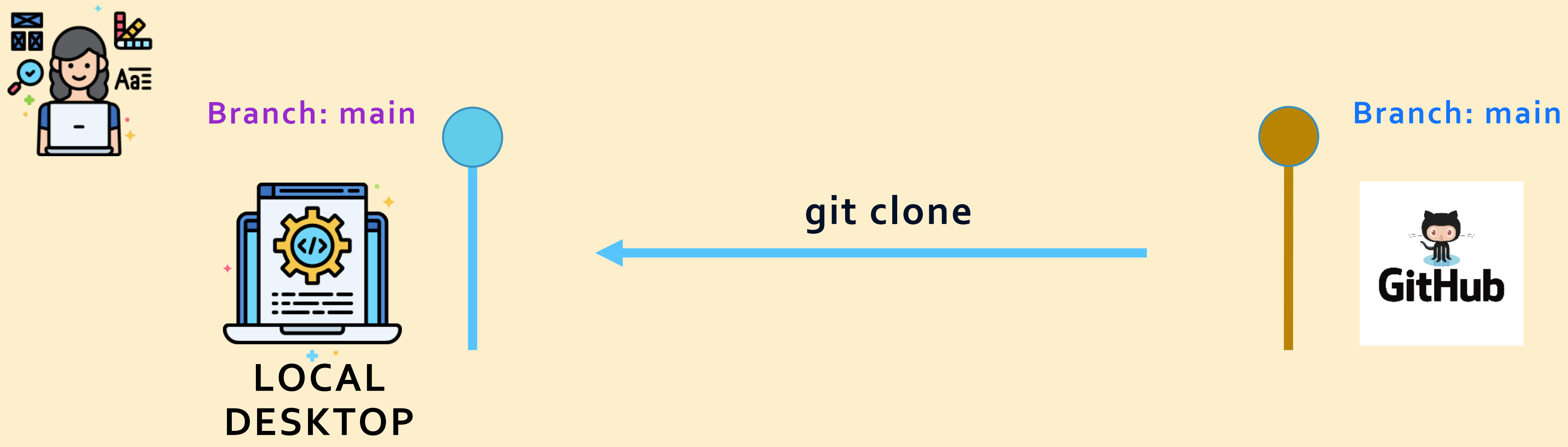
Couple Years Back..

Branch: master
LOCAL DESKTOP
Branch: master
GitHub

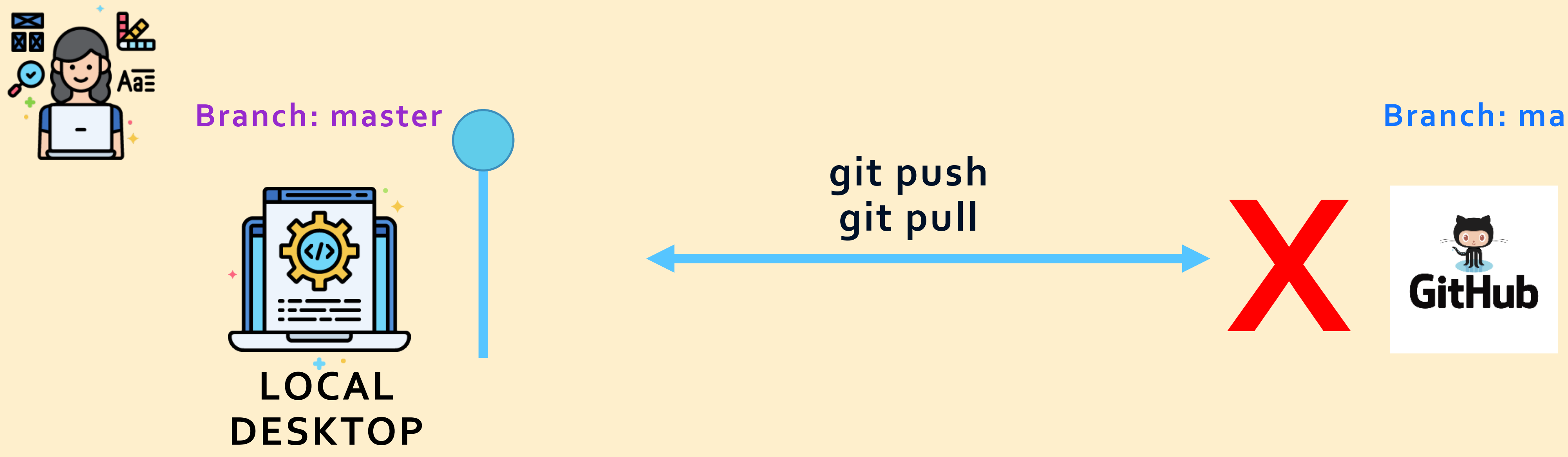
Improper Reference



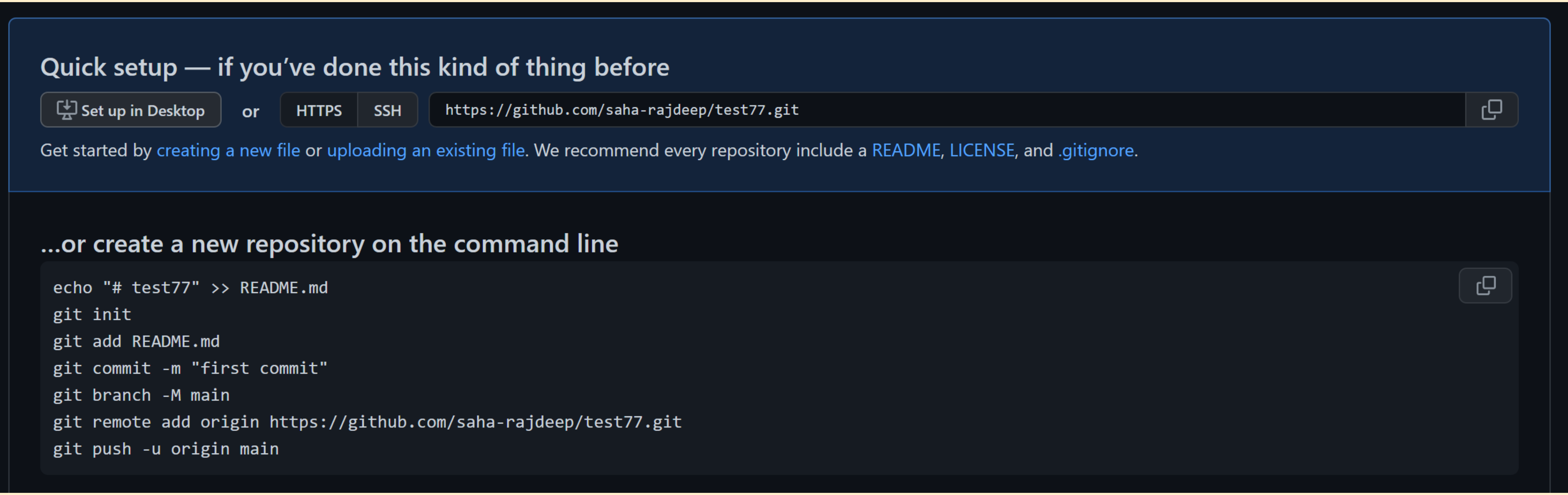
Clone



Local Folder to GitHub without Clone



Local Folder to GitHub without Clone



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

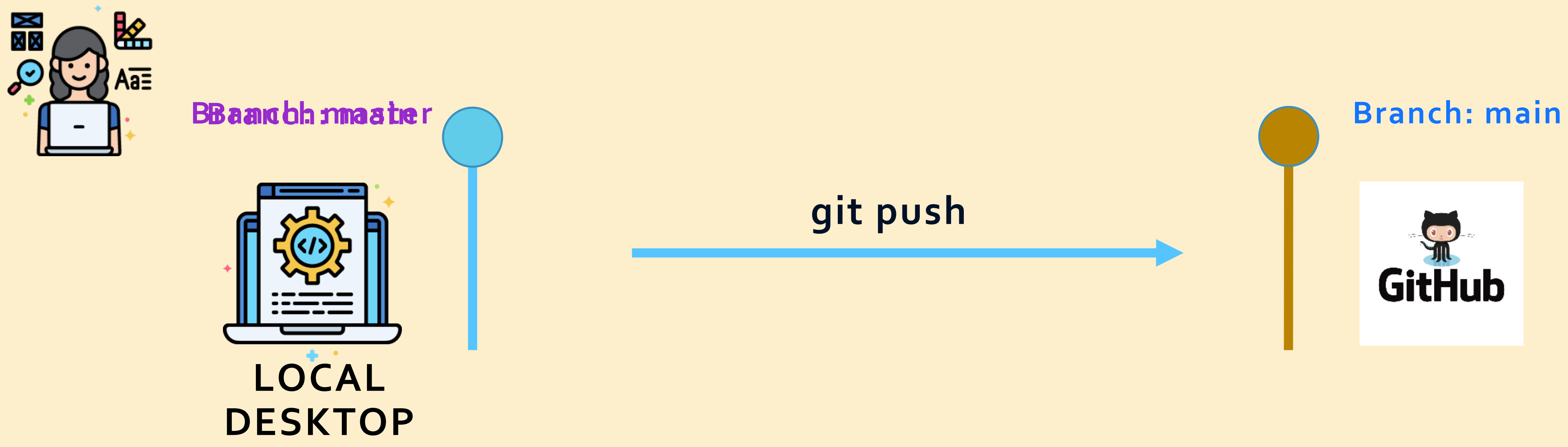
https://github.com/saha-rajdeep/test77.git

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

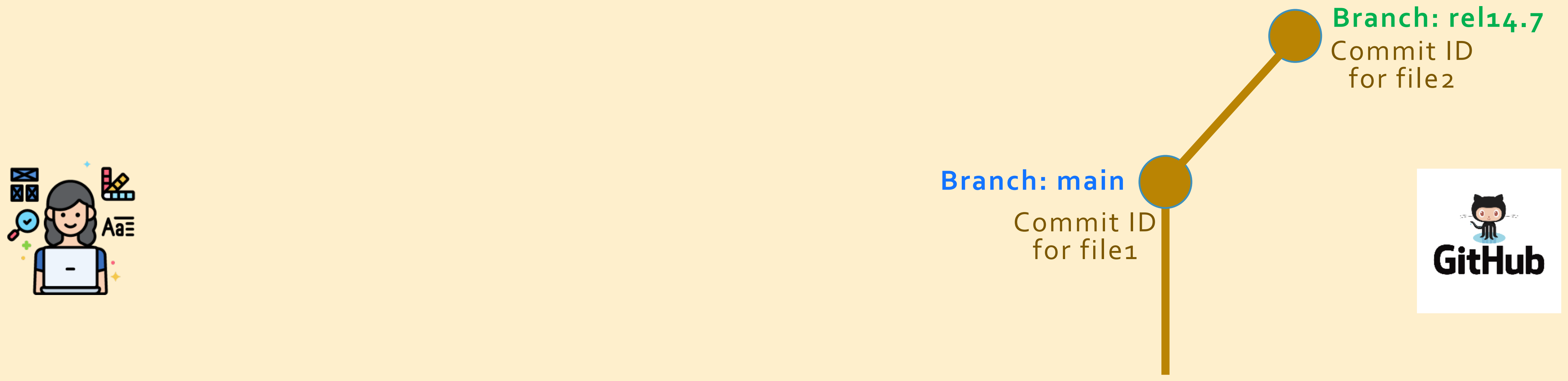
```
echo "# test77" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/saha-rajdeep/test77.git
git push -u origin main
```

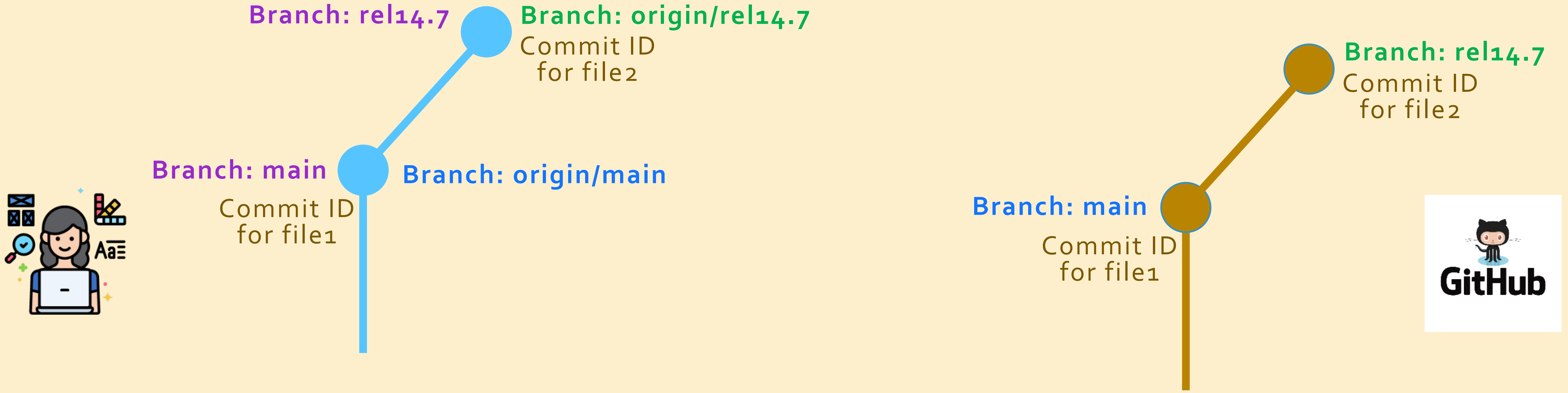
Local Folder to GitHub without Clone



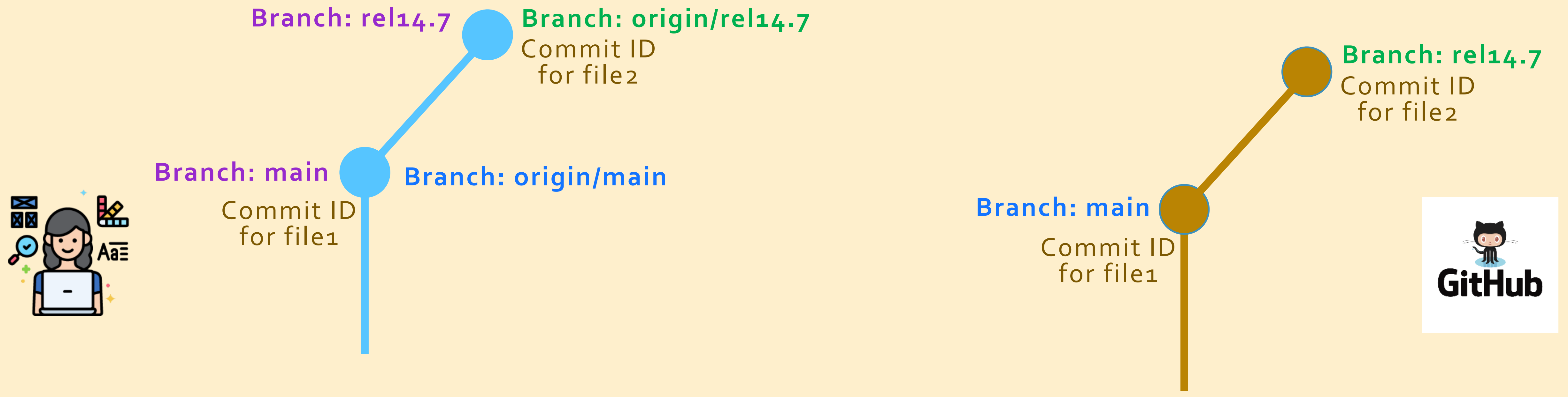
```
git branch –M main
```

Dealing with GitHub Branches

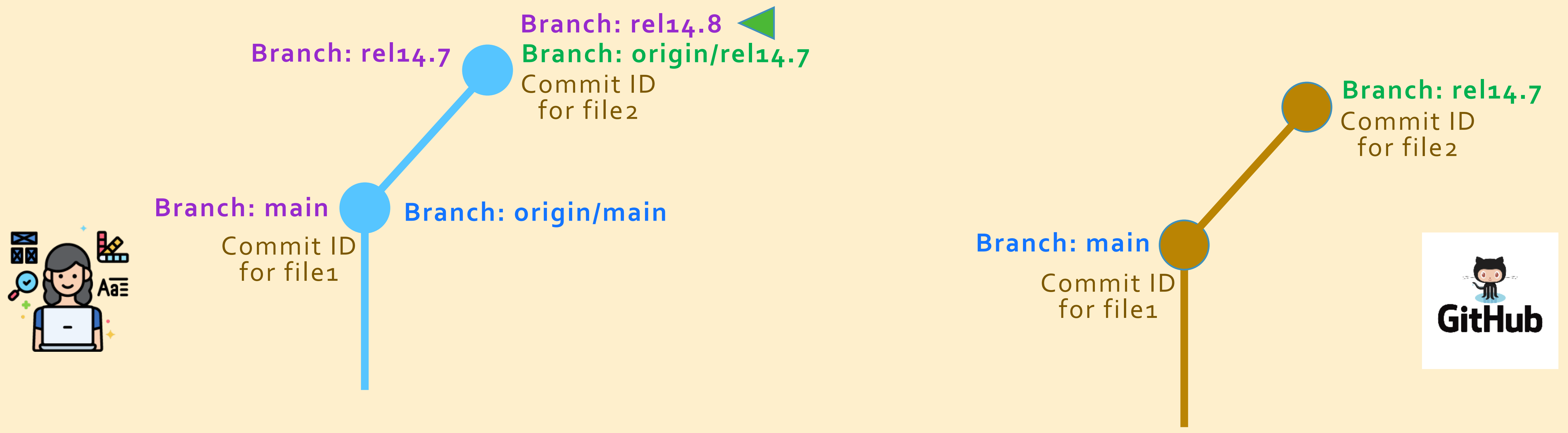

Branch: rel14.7
Commit ID for file2
Branch: main
Commit ID for file1
GitHub

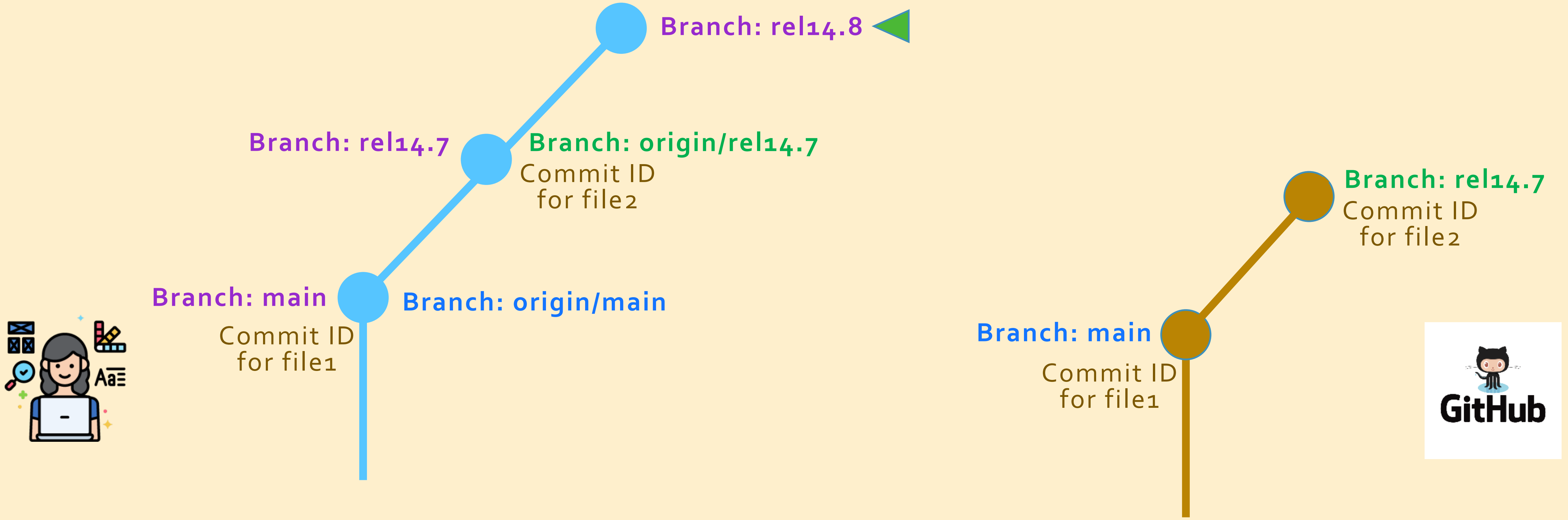

Branch: rel14.7
Branch: origin/rel14.7
Commit ID for file2
Branch: main
Branch: origin/main
Commit ID for file1
Branch: rel14.7
Commit ID for file2
Branch: main
Commit ID for file1
GitHub

git clone



```
git clone
git branch rel14.8 origin/rel14.7
git switch rel14.8
```

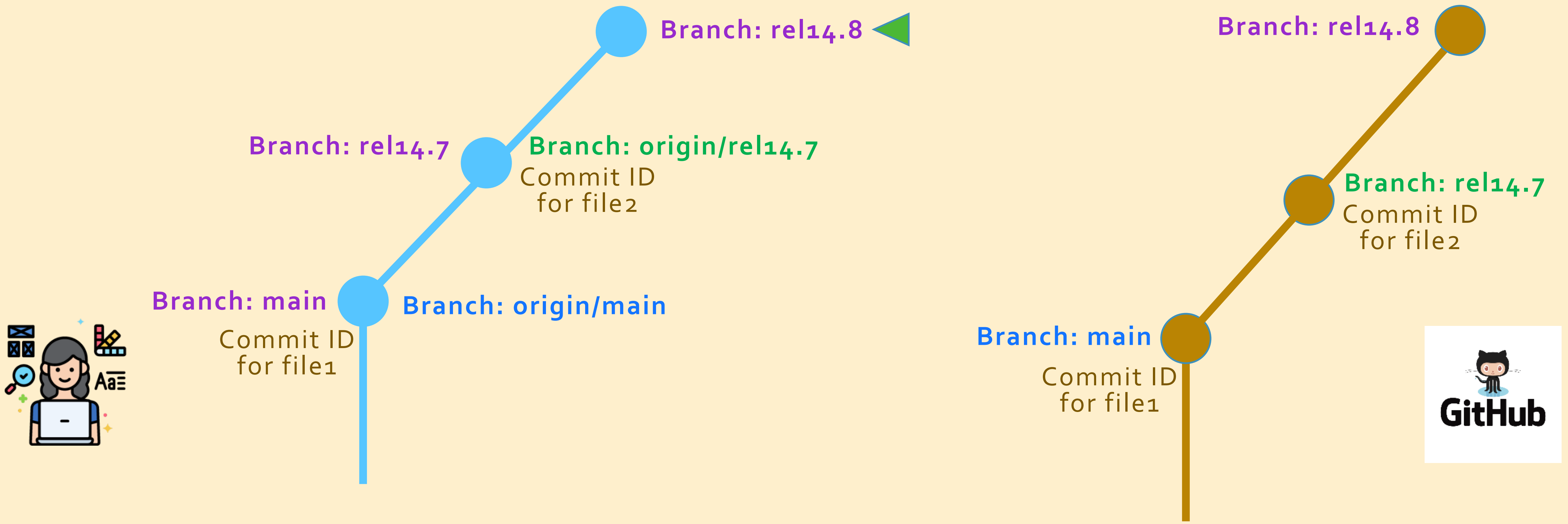

Branch: rel14.8
Branch: rel14.7
Branch: origin/rel14.7
Commit ID for file2
Branch: main
Branch: origin/main
Commit ID for file1
Branch: rel14.7
Commit ID for file2
Branch: main
Commit ID for file1
GitHub



git push

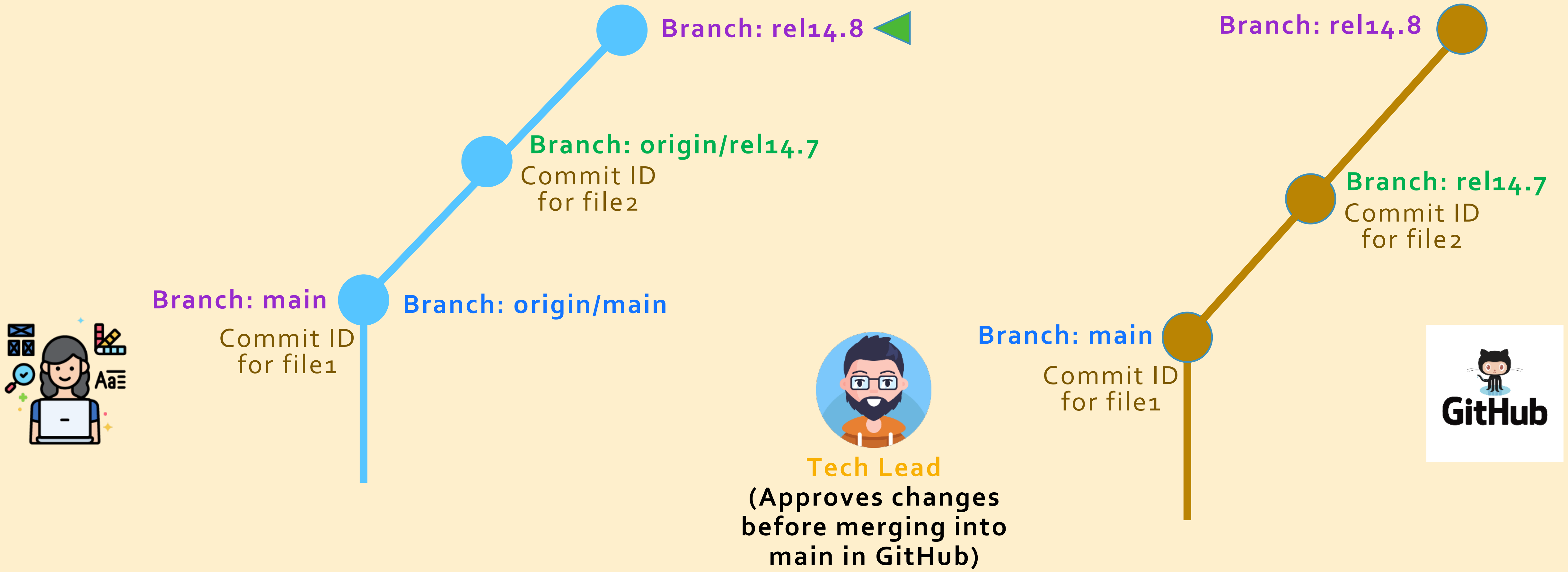
(Error coz branch rel14.8 does not exist in remote)

git push origin HEAD



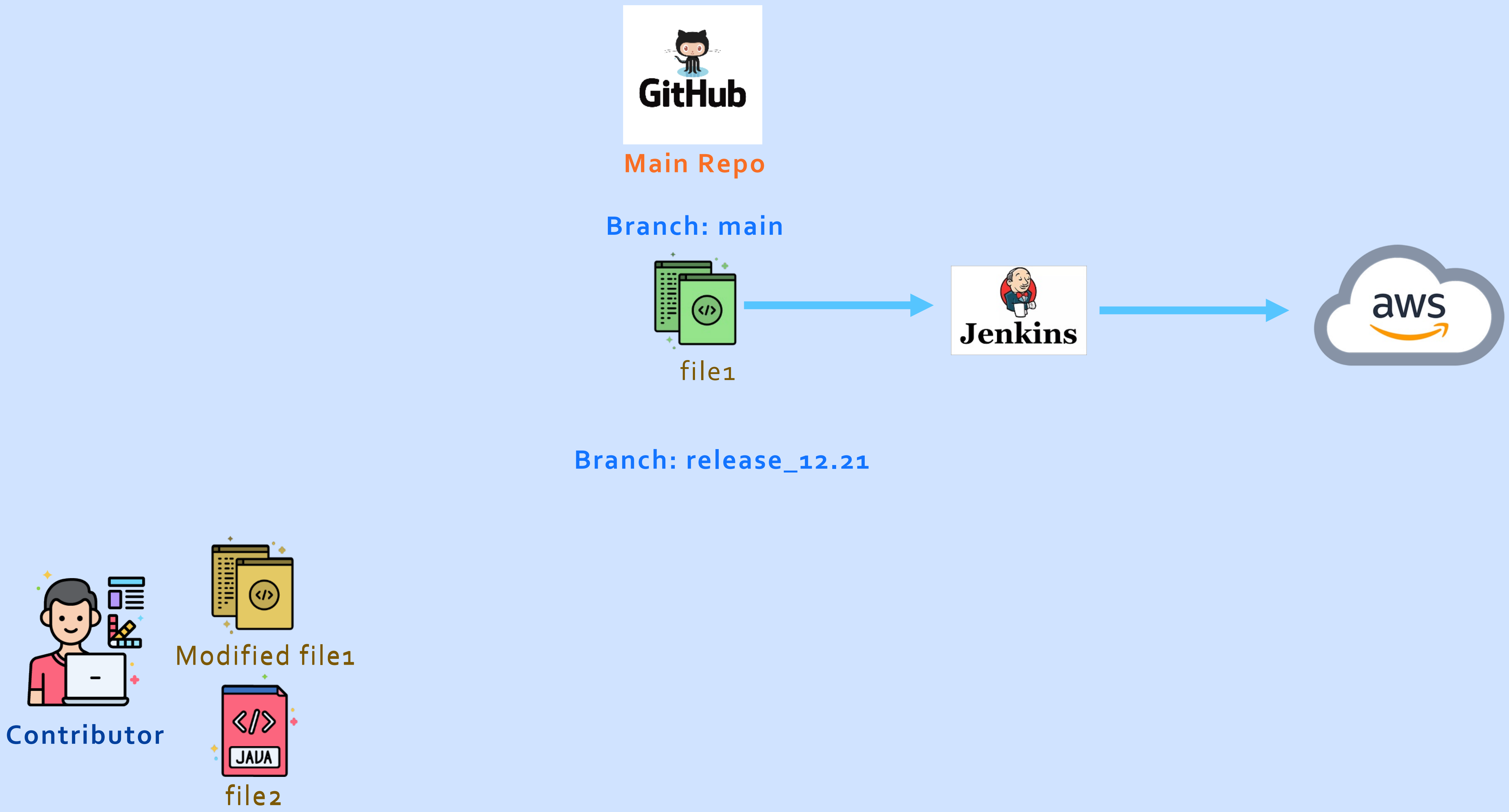
git push origin HEAD

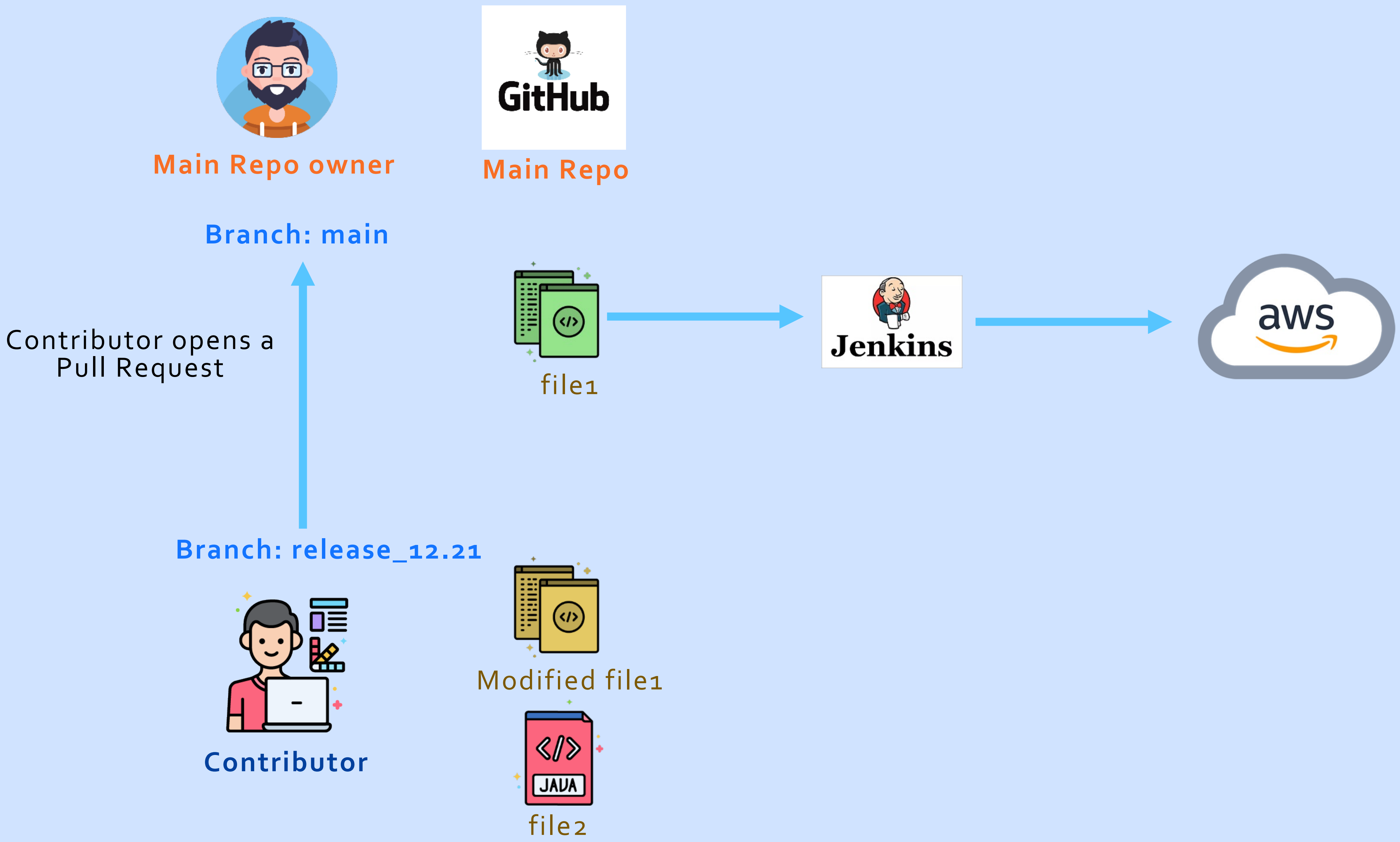
Pull Request (Merging into Main)


Branch: rel14.8
Branch: origin/rel14.7
Commit ID for file2
Branch: main
Branch: origin/main
Commit ID for file1
Tech Lead
(Approves changes before merging into main in GitHub)
Branch: rel14.8
Branch: rel14.7
Commit ID for file2
Branch: main
Commit ID for file1
GitHub

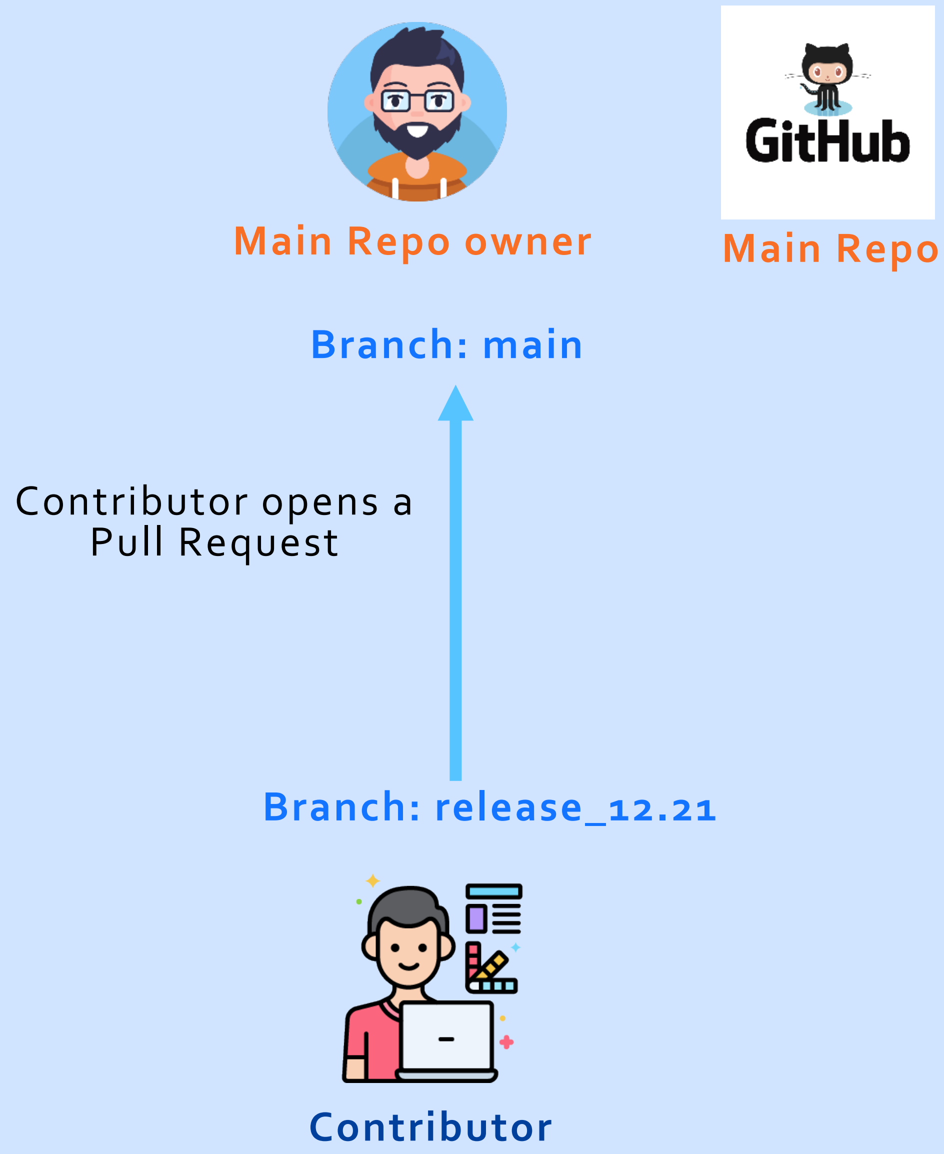
Pull Request

Raj Saha
cloudwithraj.com
Cloud With Raj

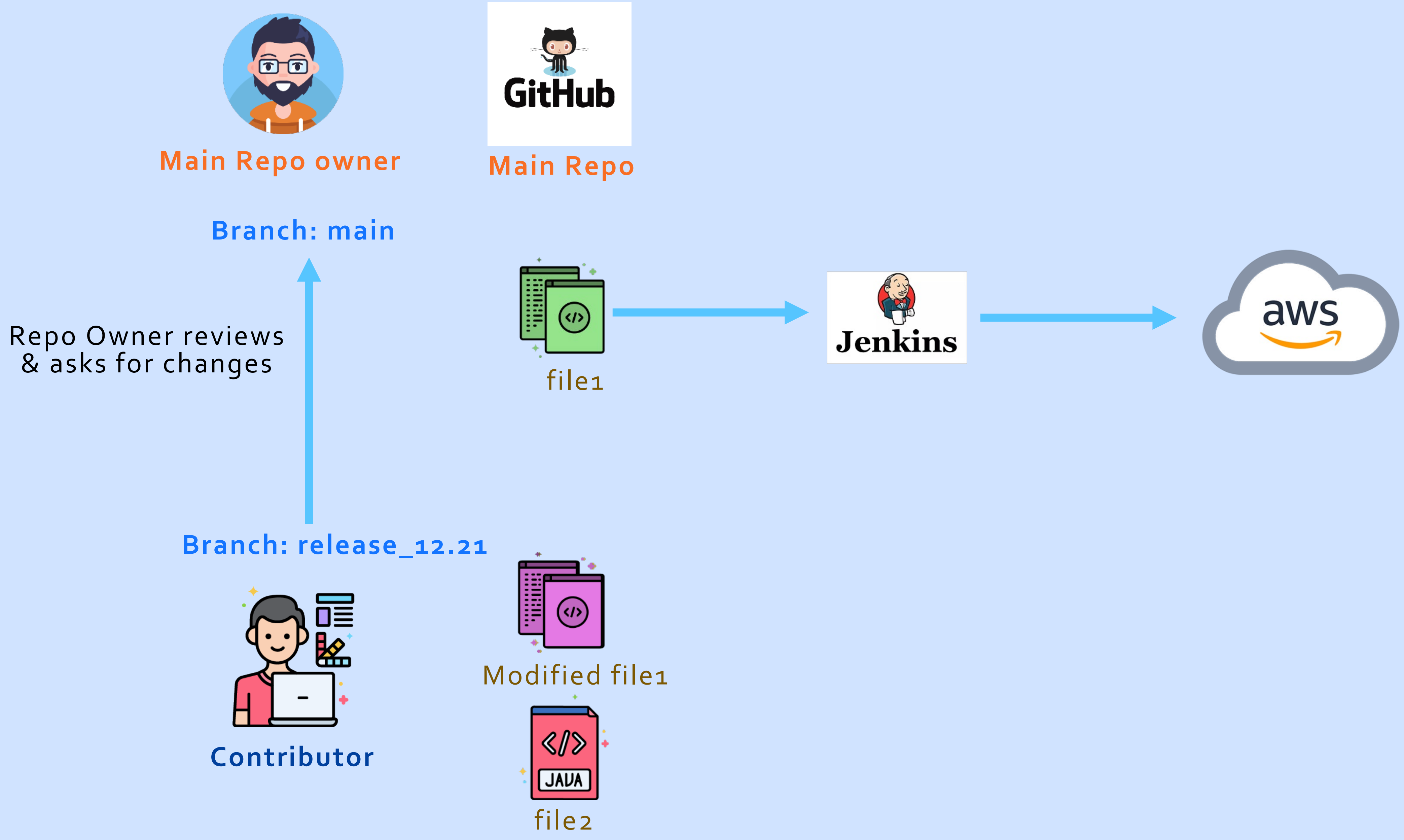

GitHub
Main Repo
Branch: main
file1
Jenkins
aws
Branch: release_12.21
Modified file1
Contributor
JAVA
file2

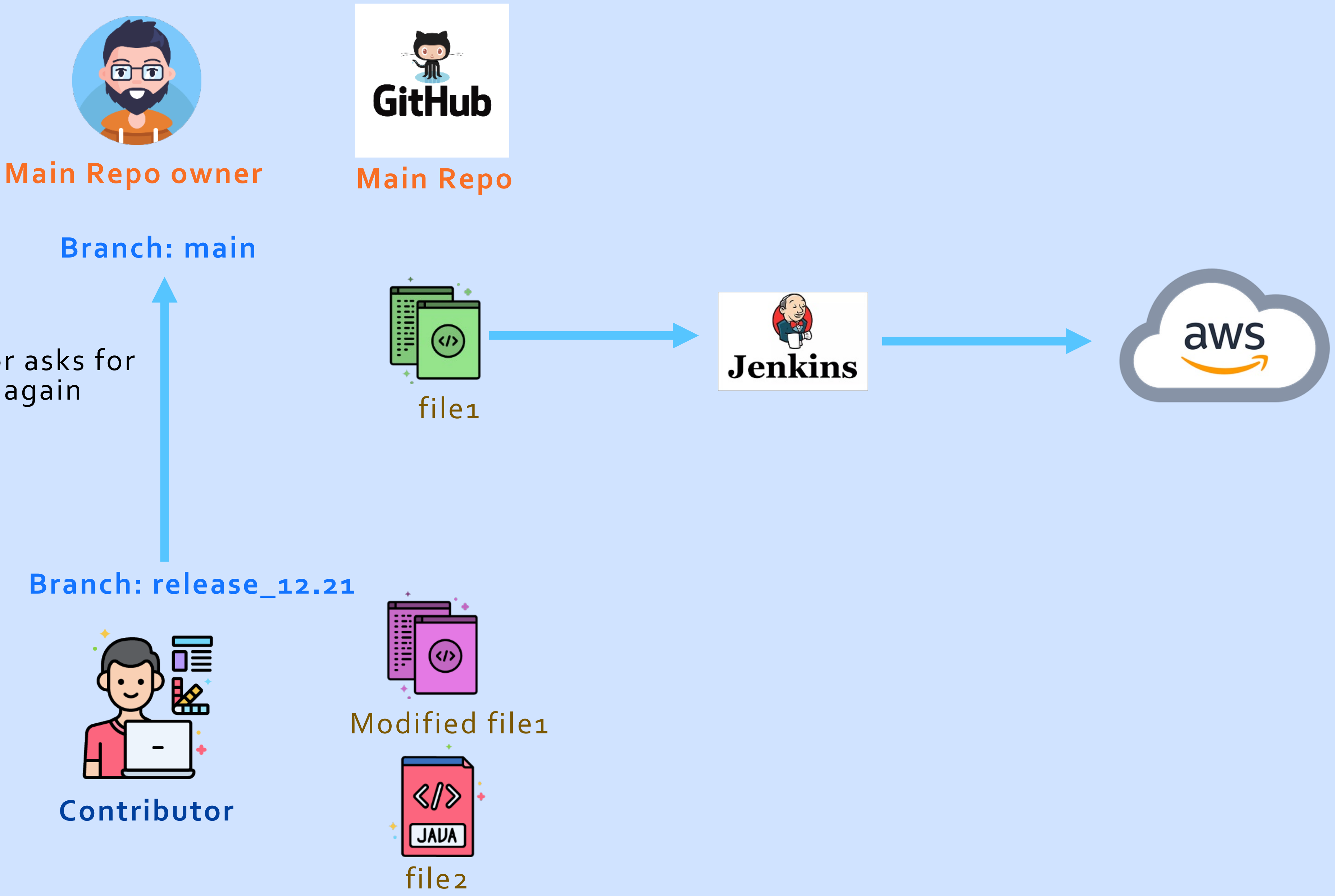

Main Repo owner
GitHub
Main Repo
Branch: main
Contributor opens a Pull Request
file1
Jenkins
aws
Branch: release_12.21
Modified file1
Contributor
JAVA
file2

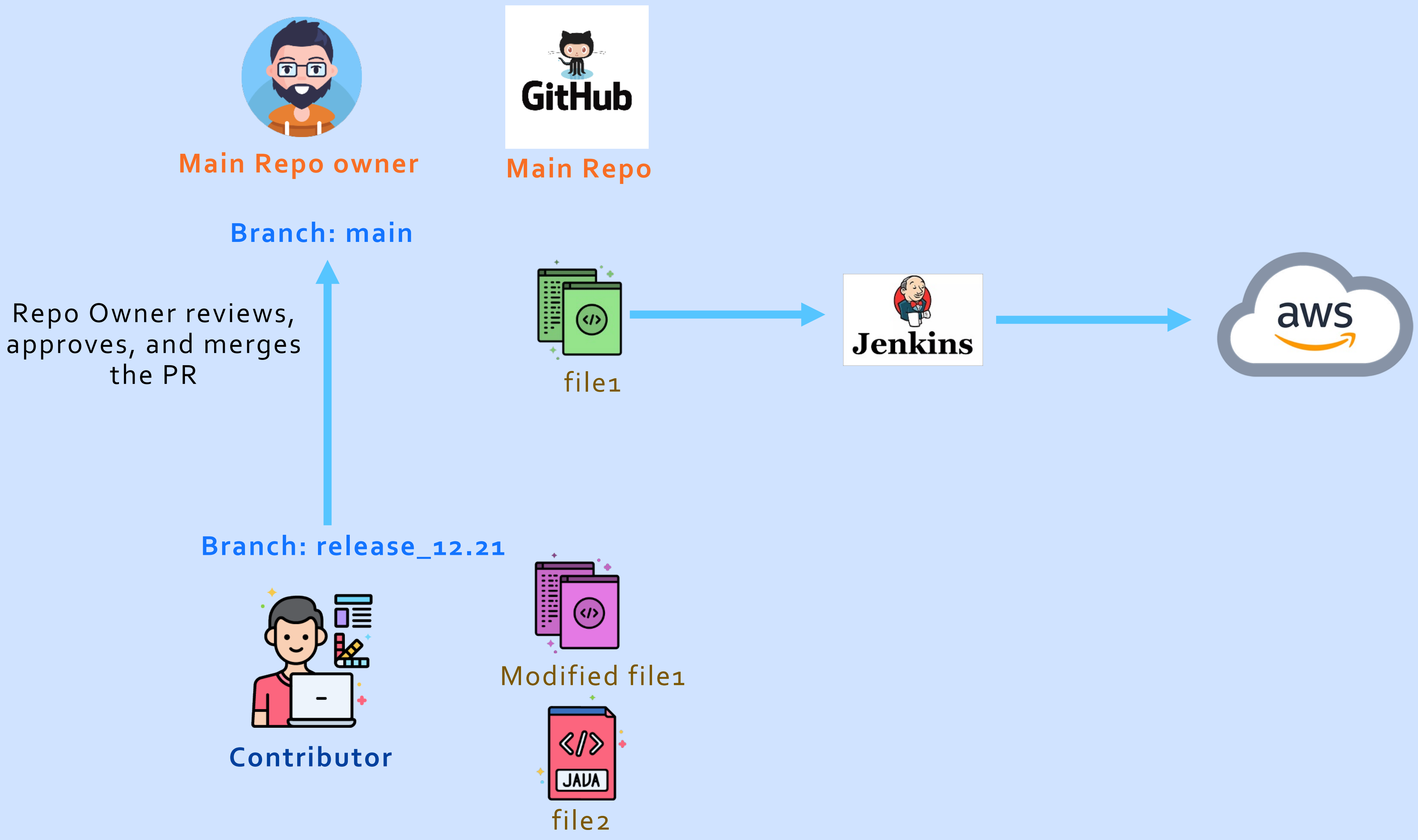
Pull Request (PR)

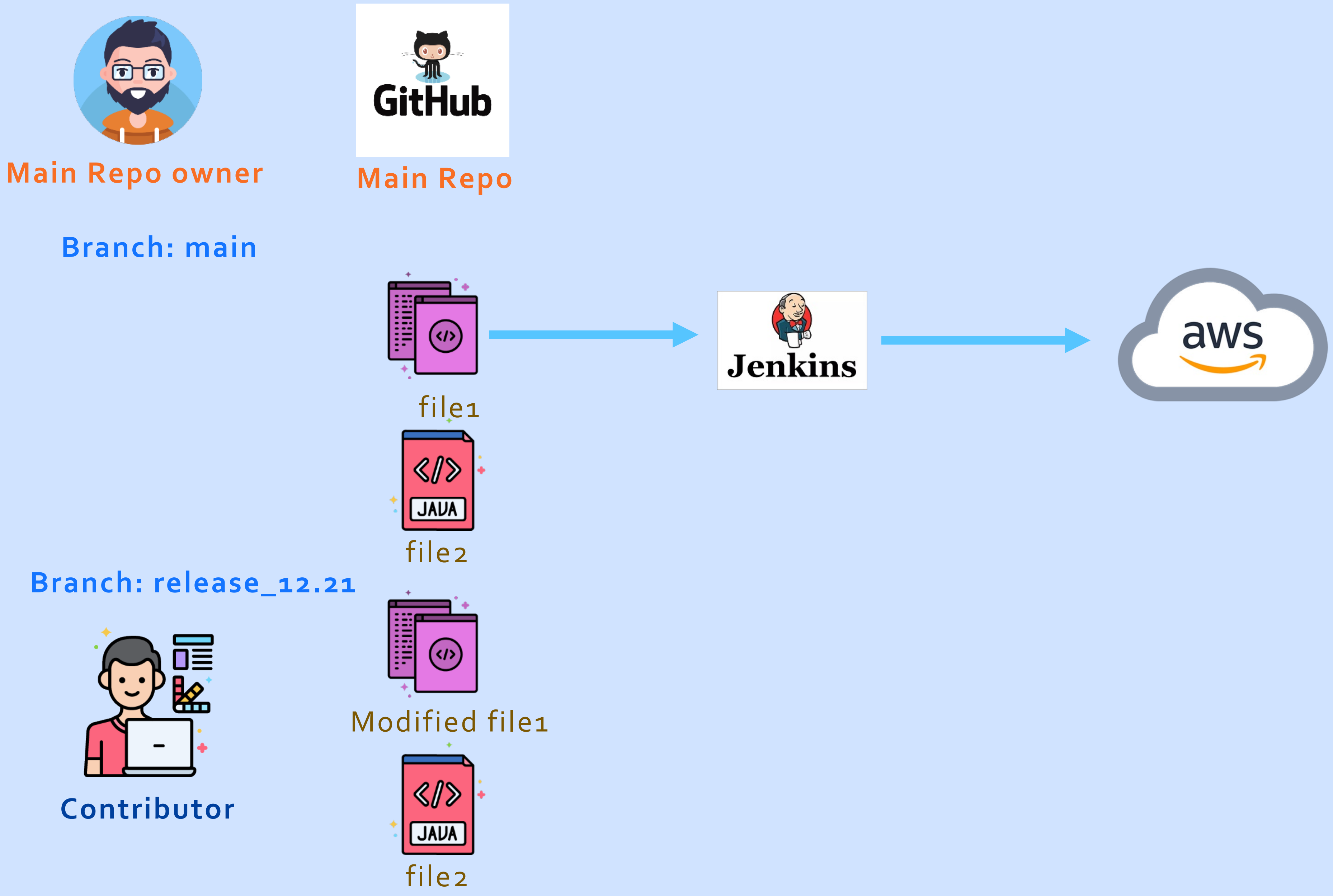


- Request to "Pull" the changes from one branch to another
  - From release_12.21 to main
  - Can be from branch in a different repo
- Pull Request allows collaborators of the project to review, comment, and update codes
- Pull Request is GitHub specific term
  - Same concept is called "Merge Request" in GitLab


Main Repo owner
GitHub
Main Repo
Branch: main
Repo Owner reviews & asks for changes
file1
Jenkins
aws
Branch: release_12.21
Contributor
Modified file1
JAVA
file2

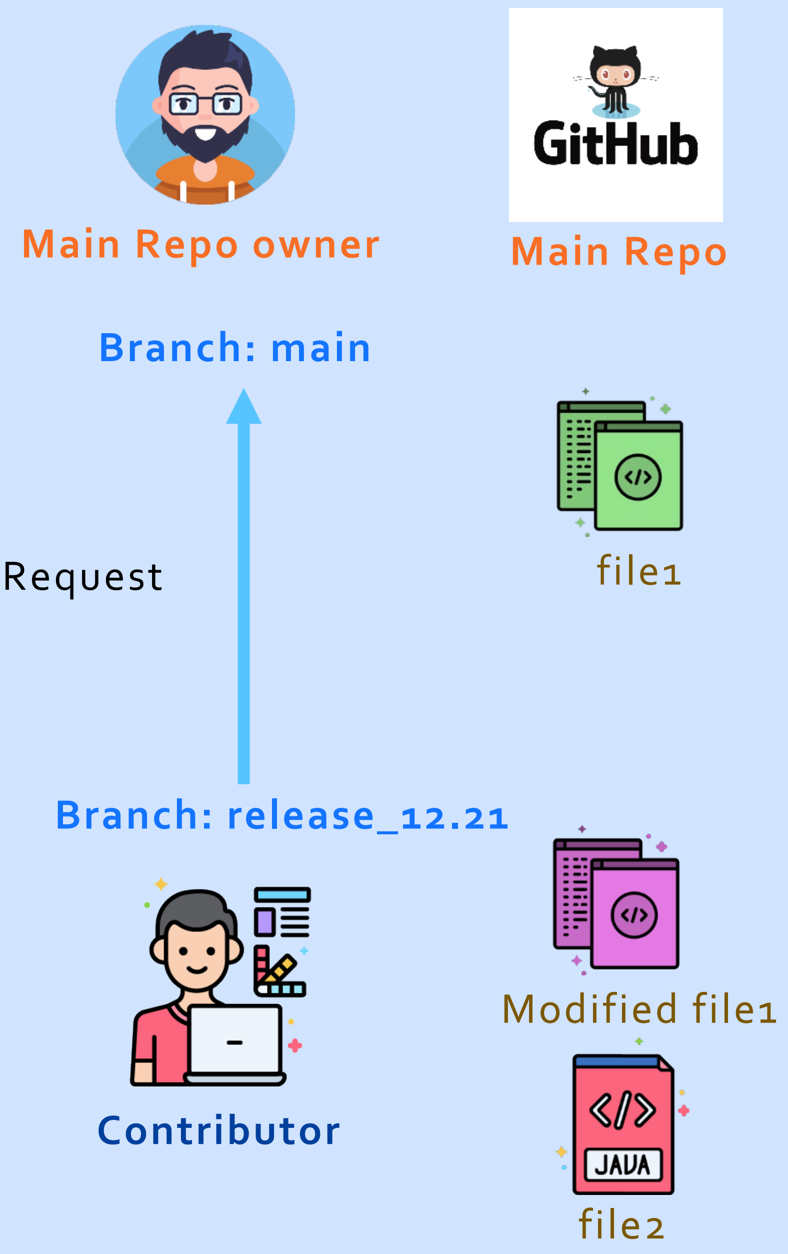

Main Repo owner
GitHub
Main Repo
Branch: main
Contributor asks for review again
file1
Jenkins
aws
Branch: release_12.21
Contributor
Modified file1
JAVA
file2

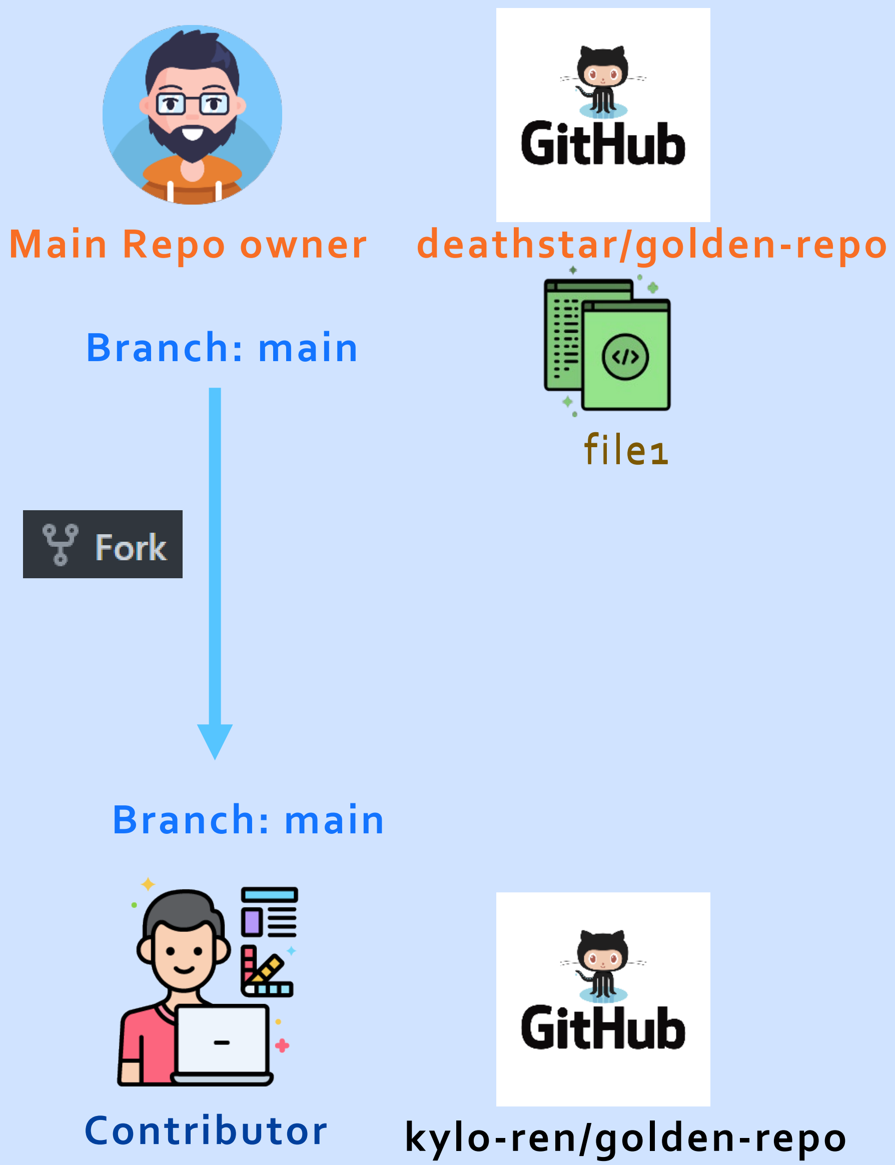

Main Repo owner
Main Repo
Branch: main
Repo Owner reviews, approves, and merges the PR
file1
Jenkins
aws
Branch: release_12.21
Modified file1
JAVA
file2
Contributor

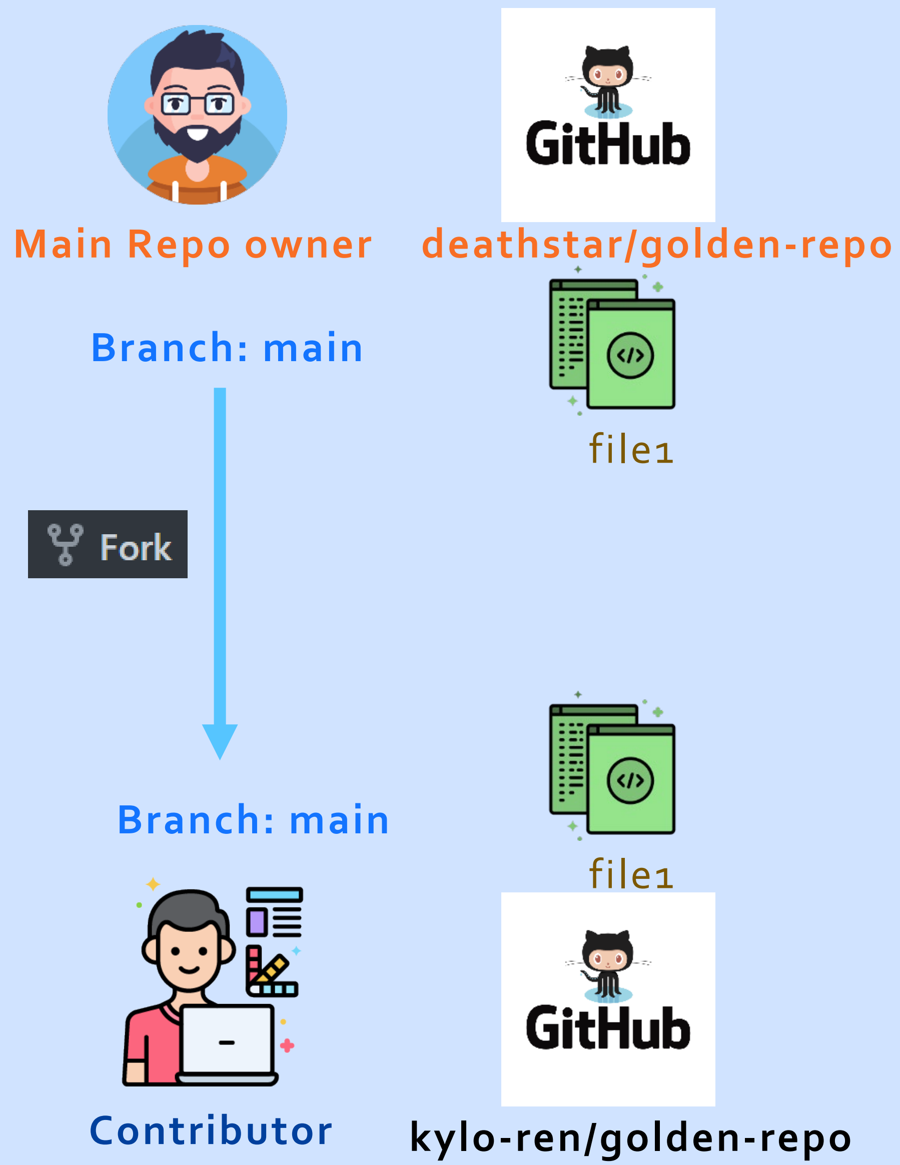

Main Repo owner
GitHub
Main Repo
Branch: main
file1
Jenkins
aws
file2
Branch: release_12.21
Modified file1
Contributor
file2

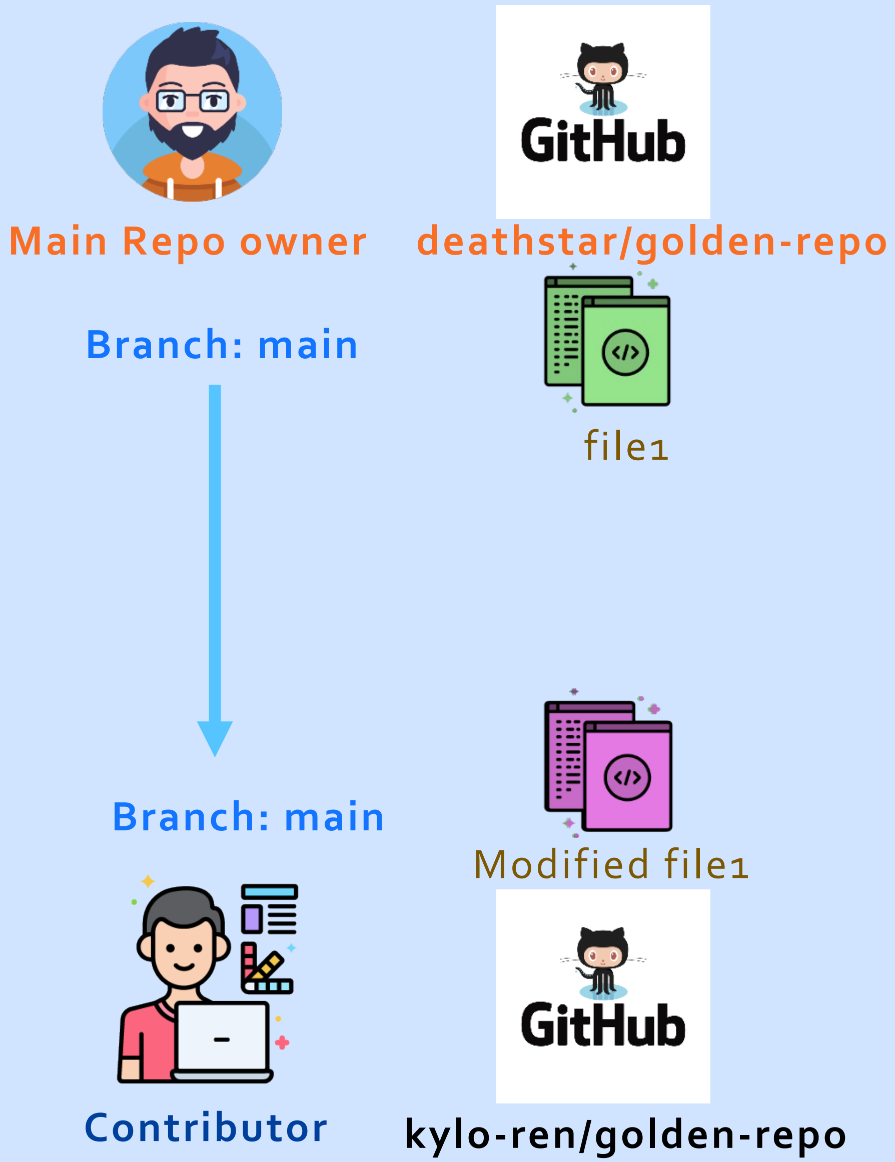
Fork

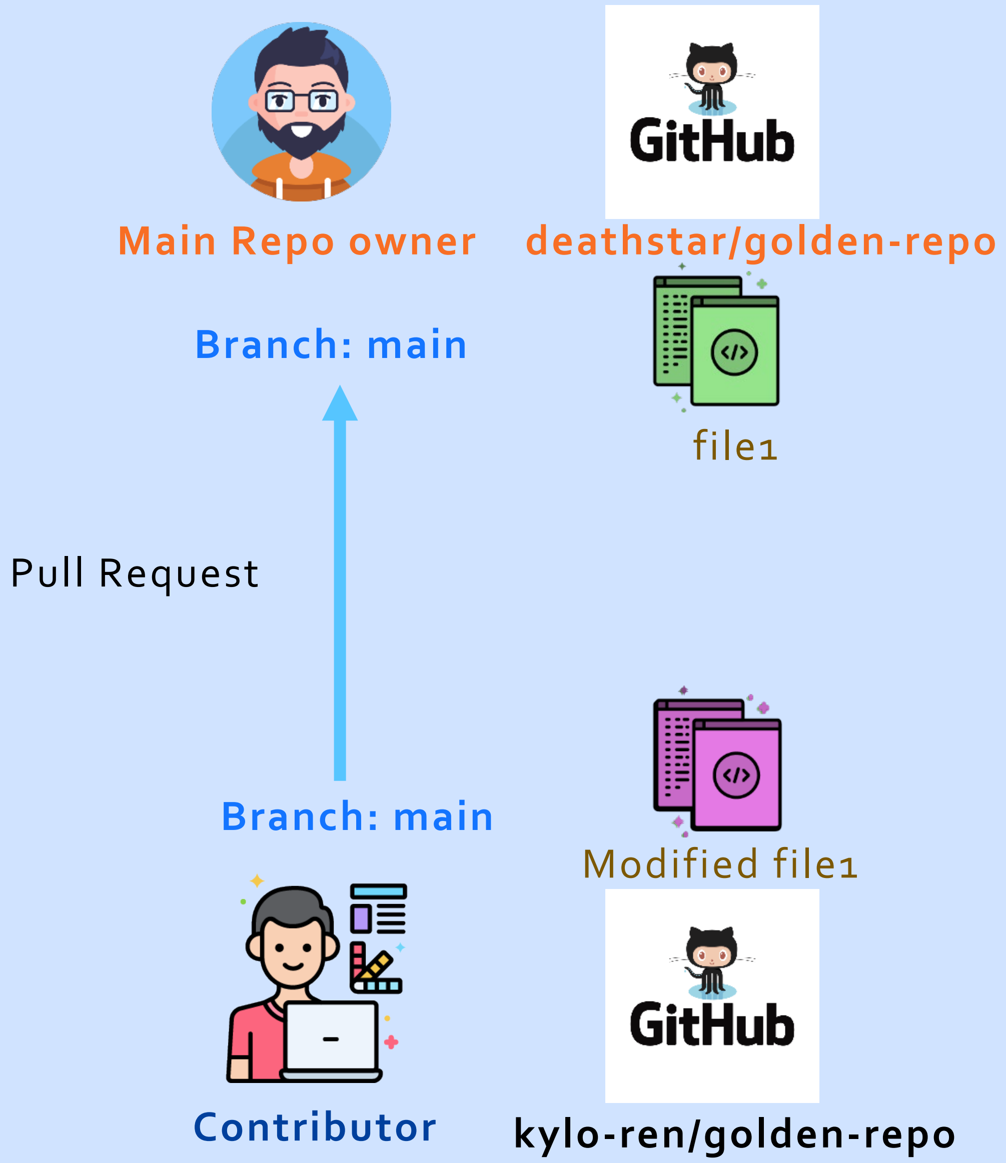
Raj Saha
cloudwithraj.com
Cloud With Raj

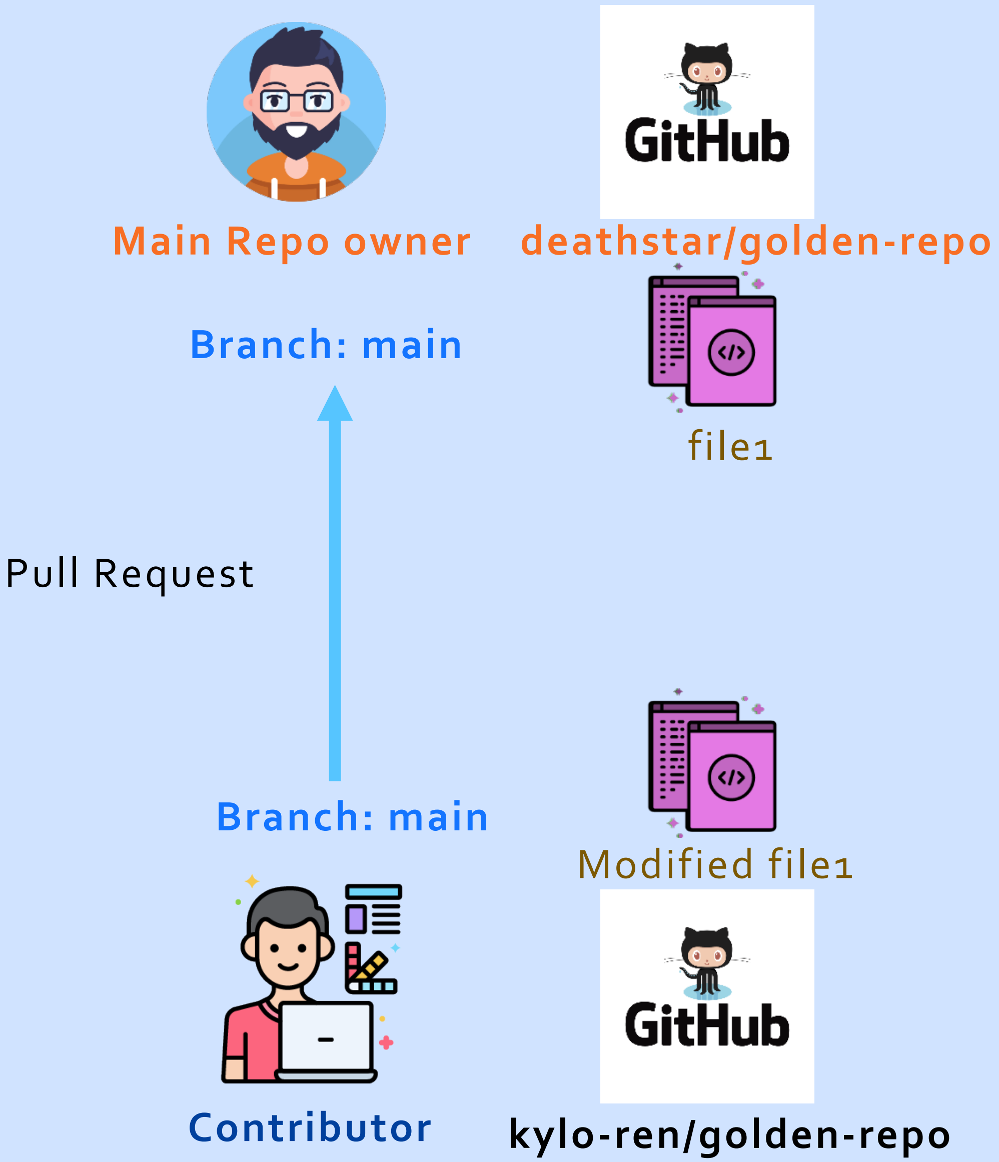

GitHub
Main Repo owner
Main Repo
Branch: main
file1
Pull Request
Branch: release_12.21
Modified file1
JAVA
file2
Contributor


Main Repo owner
deathstar/golden-repo
Branch: main
file1
Fork
Branch: main
Contributor
kylo-ren/golden-repo


Main Repo owner
deathstar/golden-repo
Branch: main
file1
Fork
Branch: main
file1
Contributor
kylo-ren/golden-repo

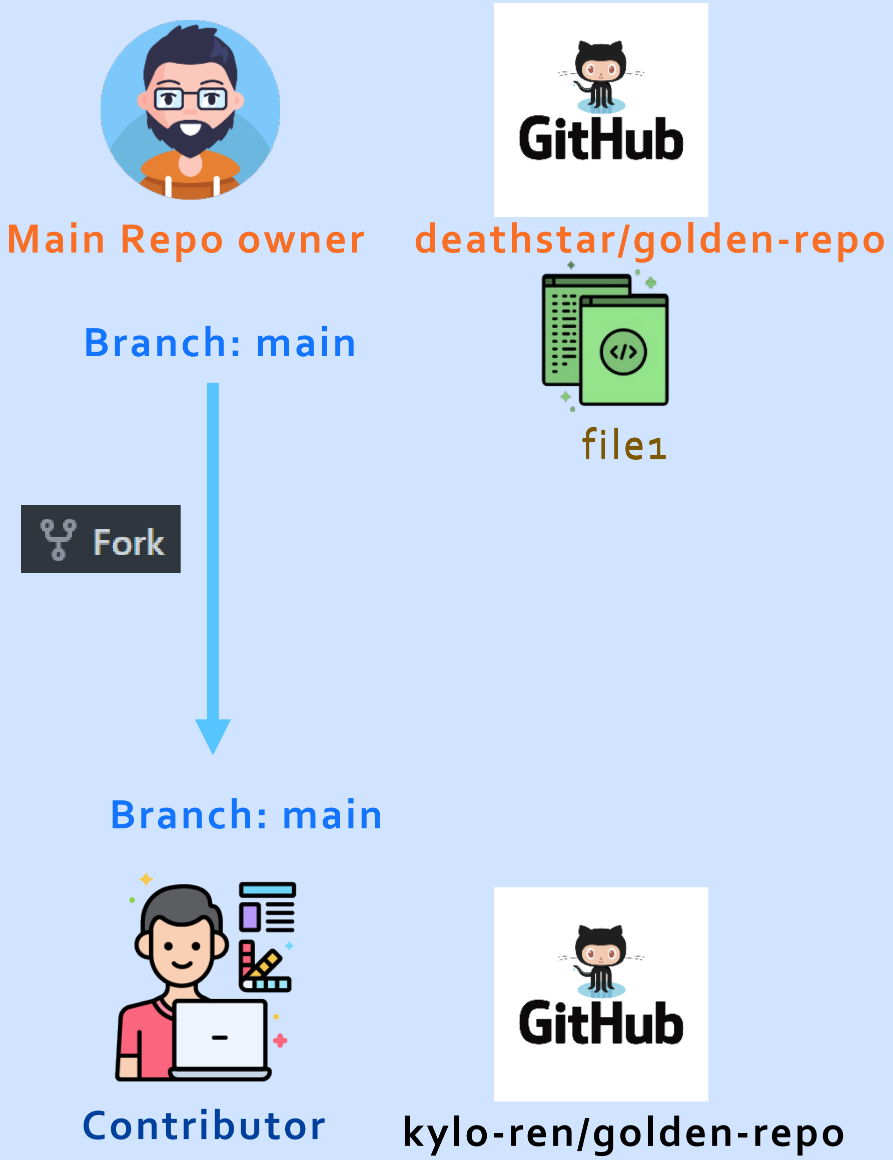

GitHub
Main Repo owner
deathstar/golden-repo
Branch: main
file1
Branch: main
Modified file1
GitHub
Contributor
kylo-ren/golden-repo

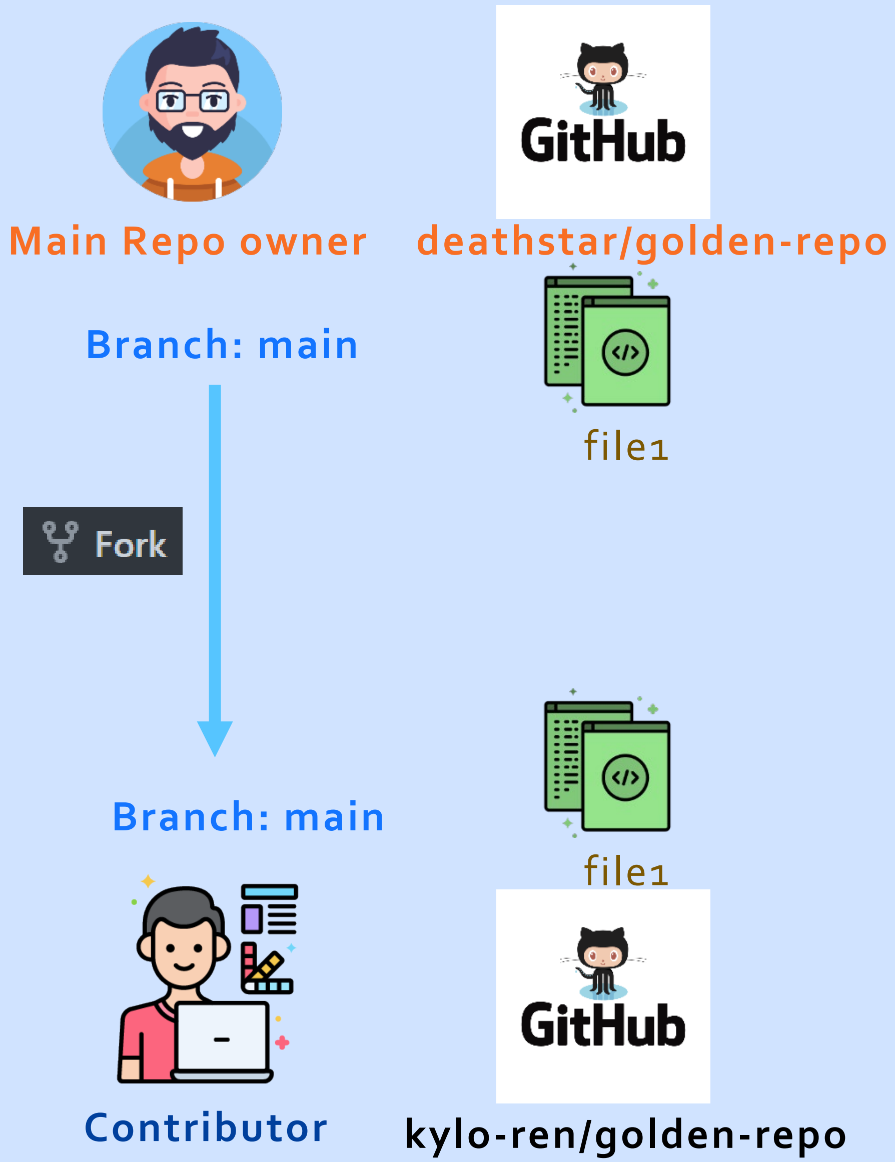

Main Repo owner
deathstar/golden-repo
Branch: main
file1
Pull Request
Branch: main
Modified file1
Contributor
kylo-ren/golden-repo

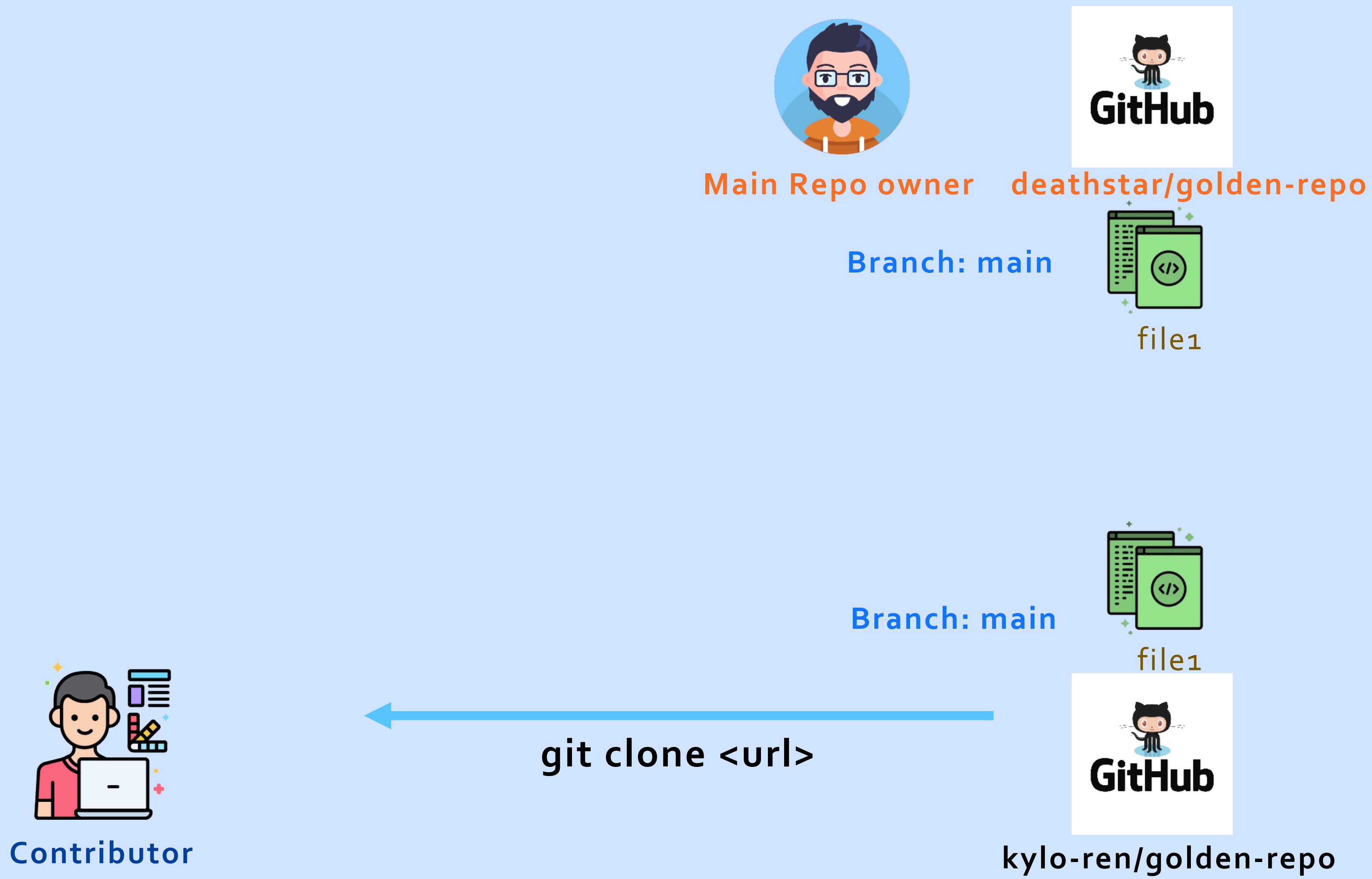

Main Repo owner
deathstar/golden-repo
Branch: main
file1
Pull Request
Branch: main
Modified file1
Contributor
kylo-ren/golden-repo

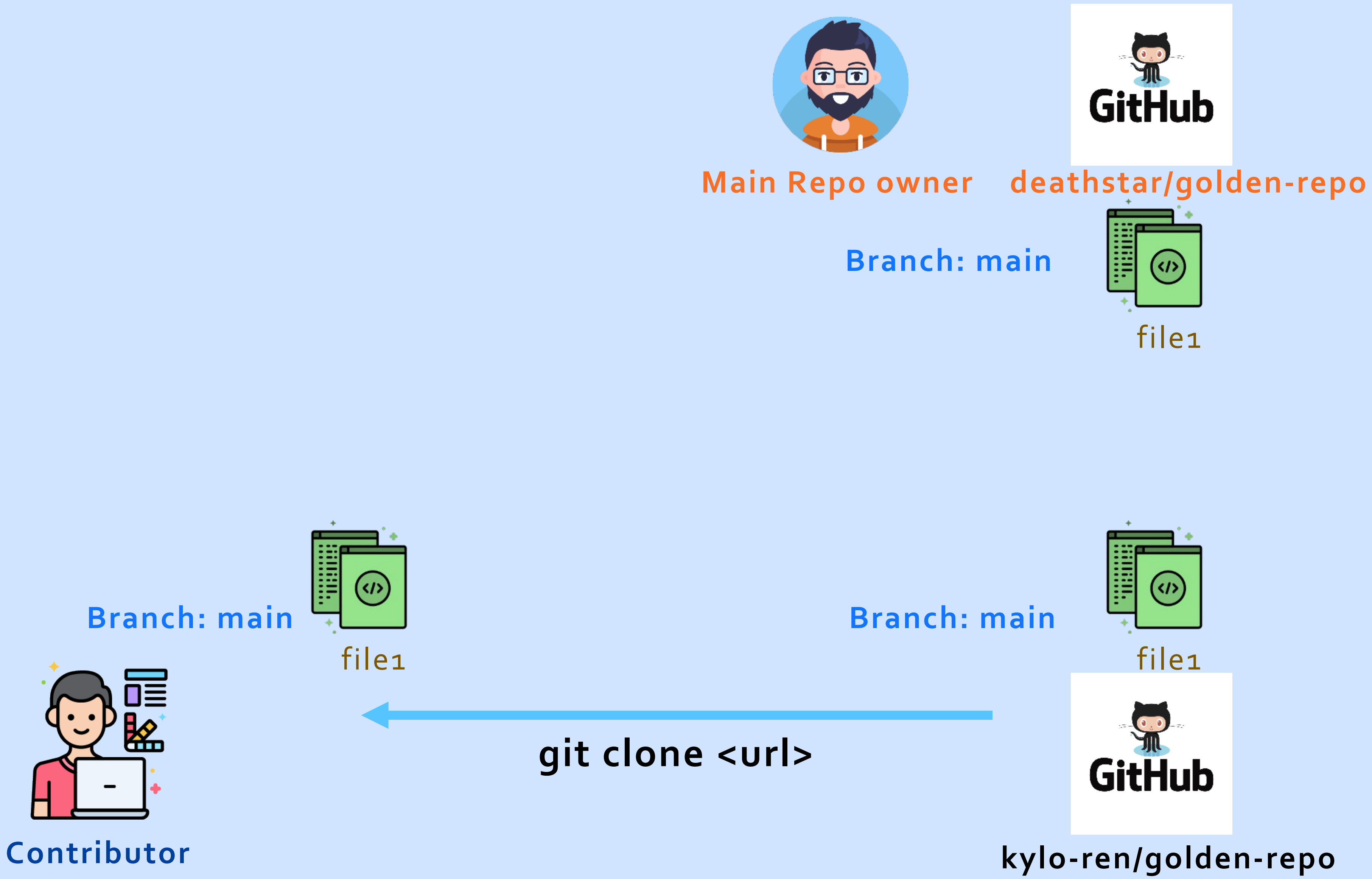
Real World GitHub Workflow

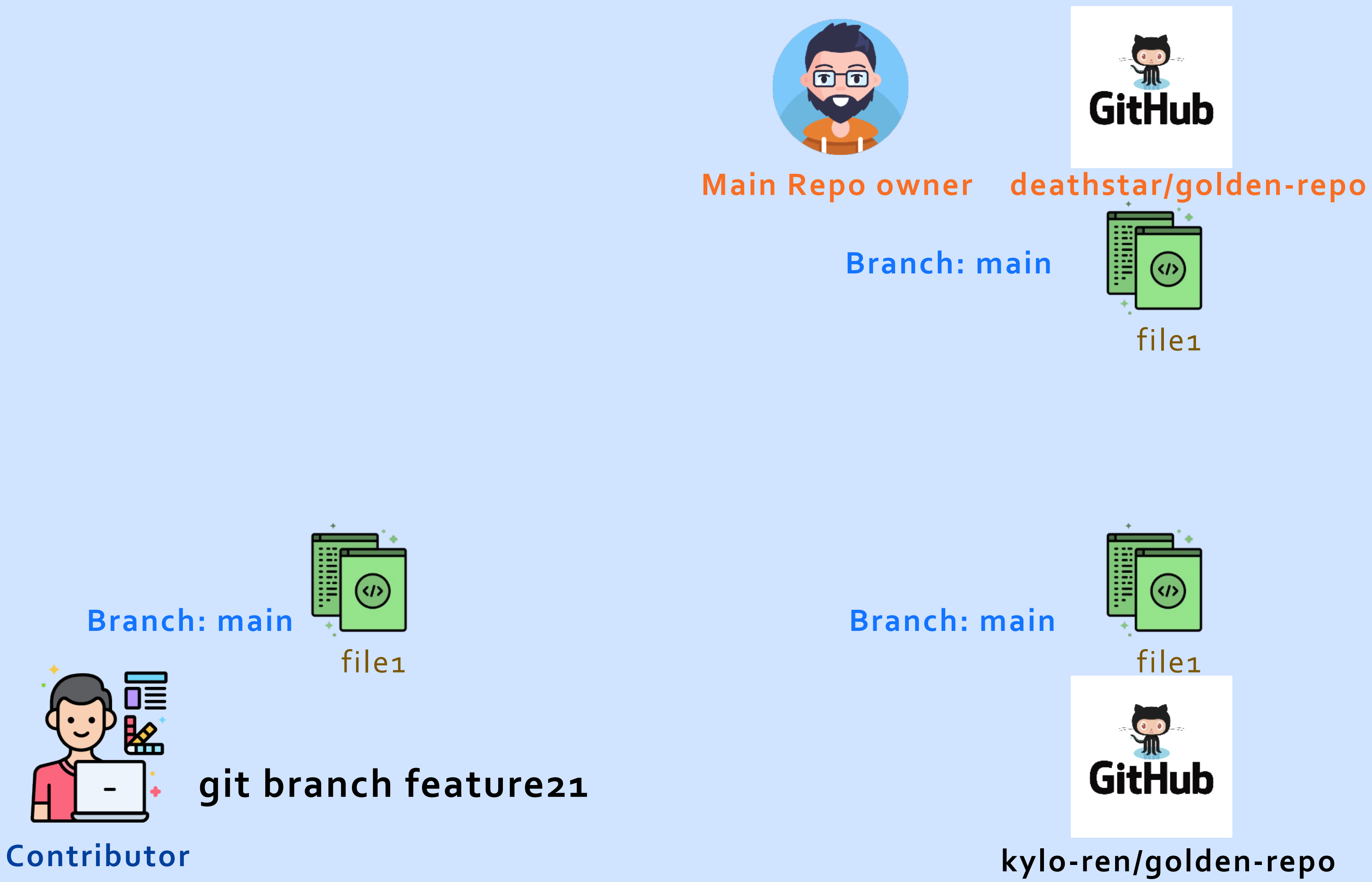
Raj Saha
cloudwithraj.com
Cloud With Raj


Main Repo owner
deathstar/golden-repo
Branch: main
file1
Fork
Branch: main
Contributor
kylo-ren/golden-repo

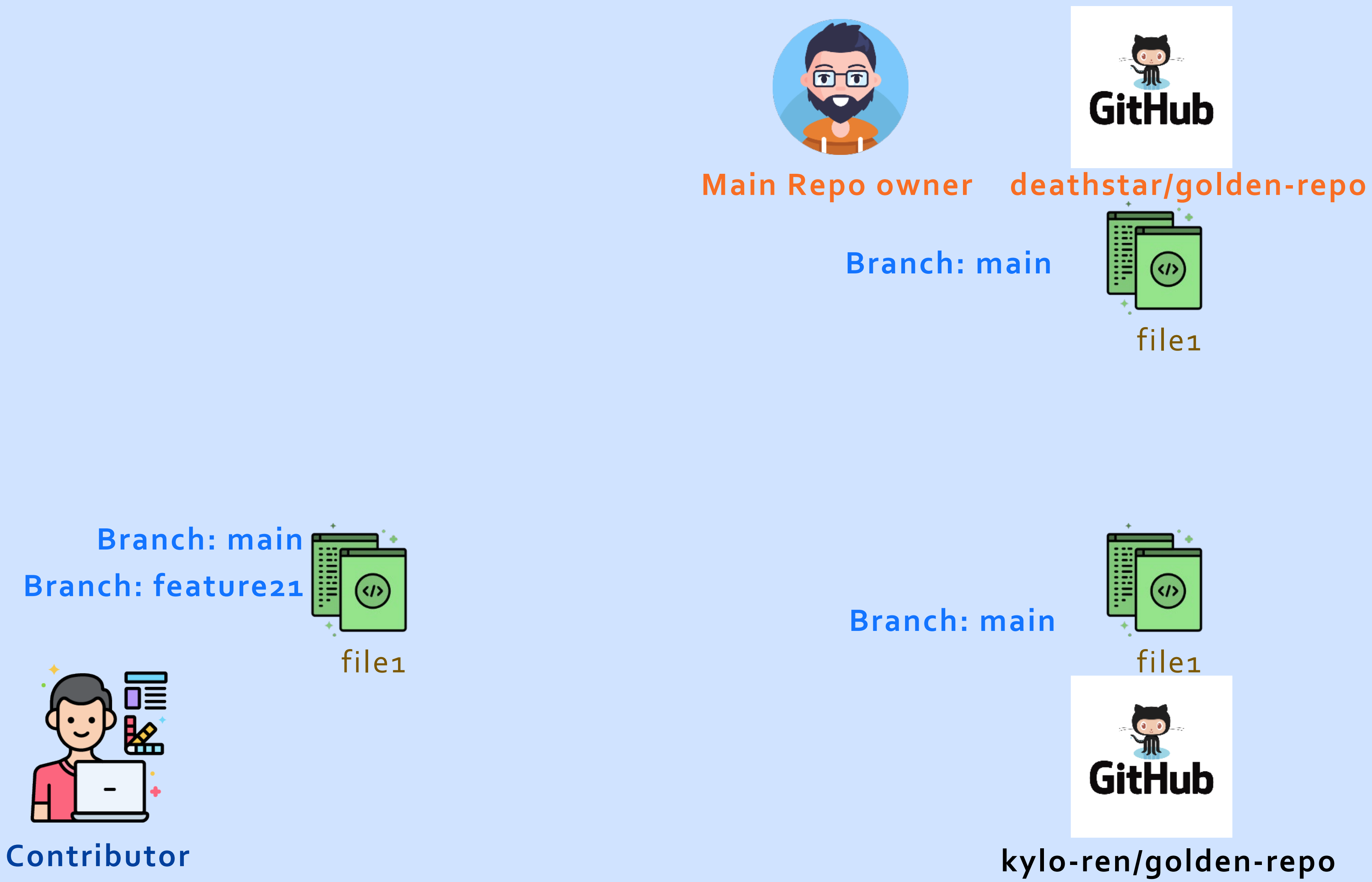

Main Repo owner
deathstar/golden-repo
Branch: main
file1
Fork
Branch: main
file1
Contributor
kylo-ren/golden-repo


Main Repo owner
deathstar/golden-repo
Branch: main
file1
Branch: main
file1
git clone <url>
Contributor
kylo-ren/golden-repo

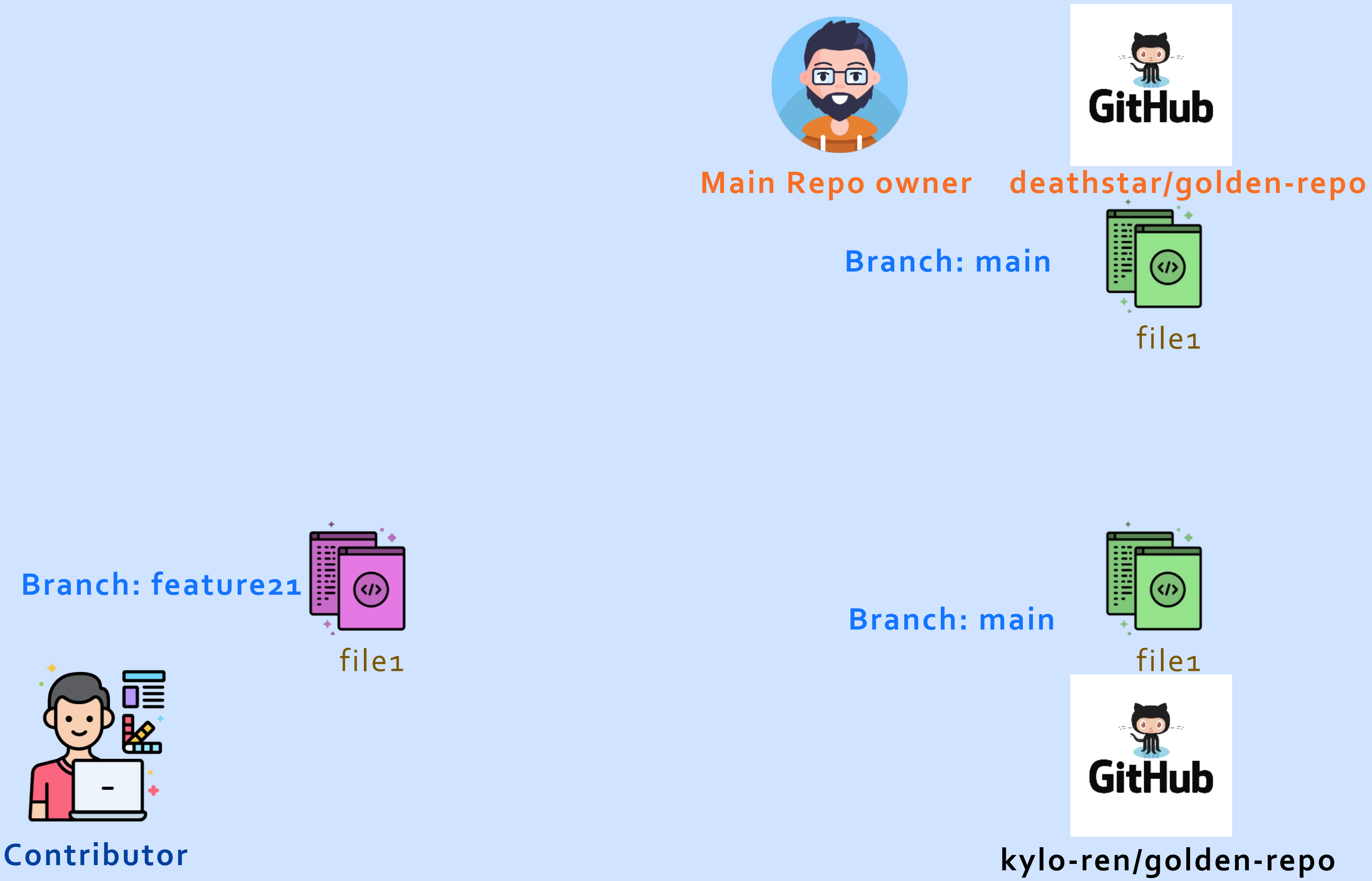

Main Repo owner
deathstar/golden-repo
Branch: main
file1
Branch: main
file1
Branch: main
file1
git clone <url>
Contributor
kylo-ren/golden-repo

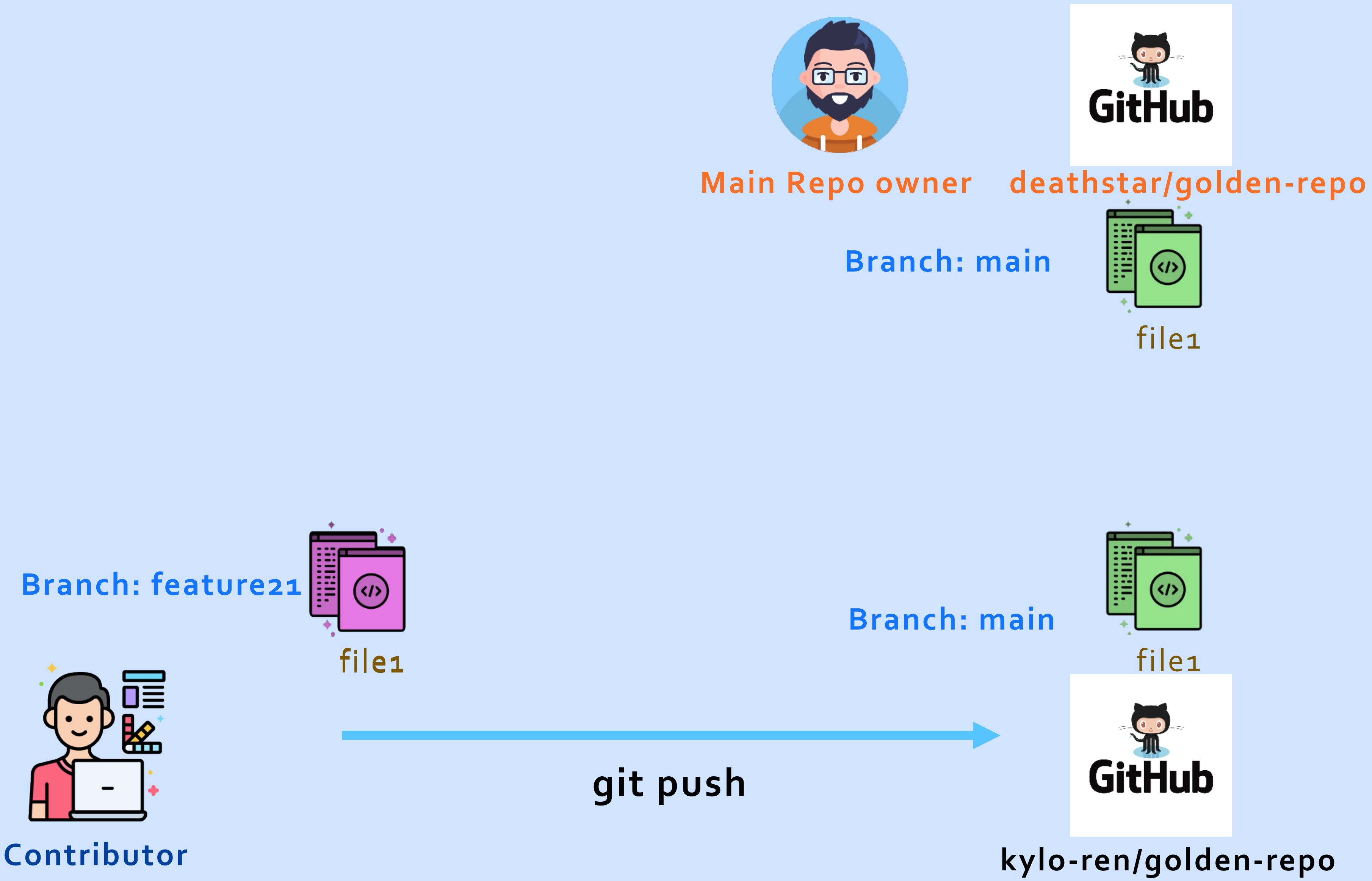


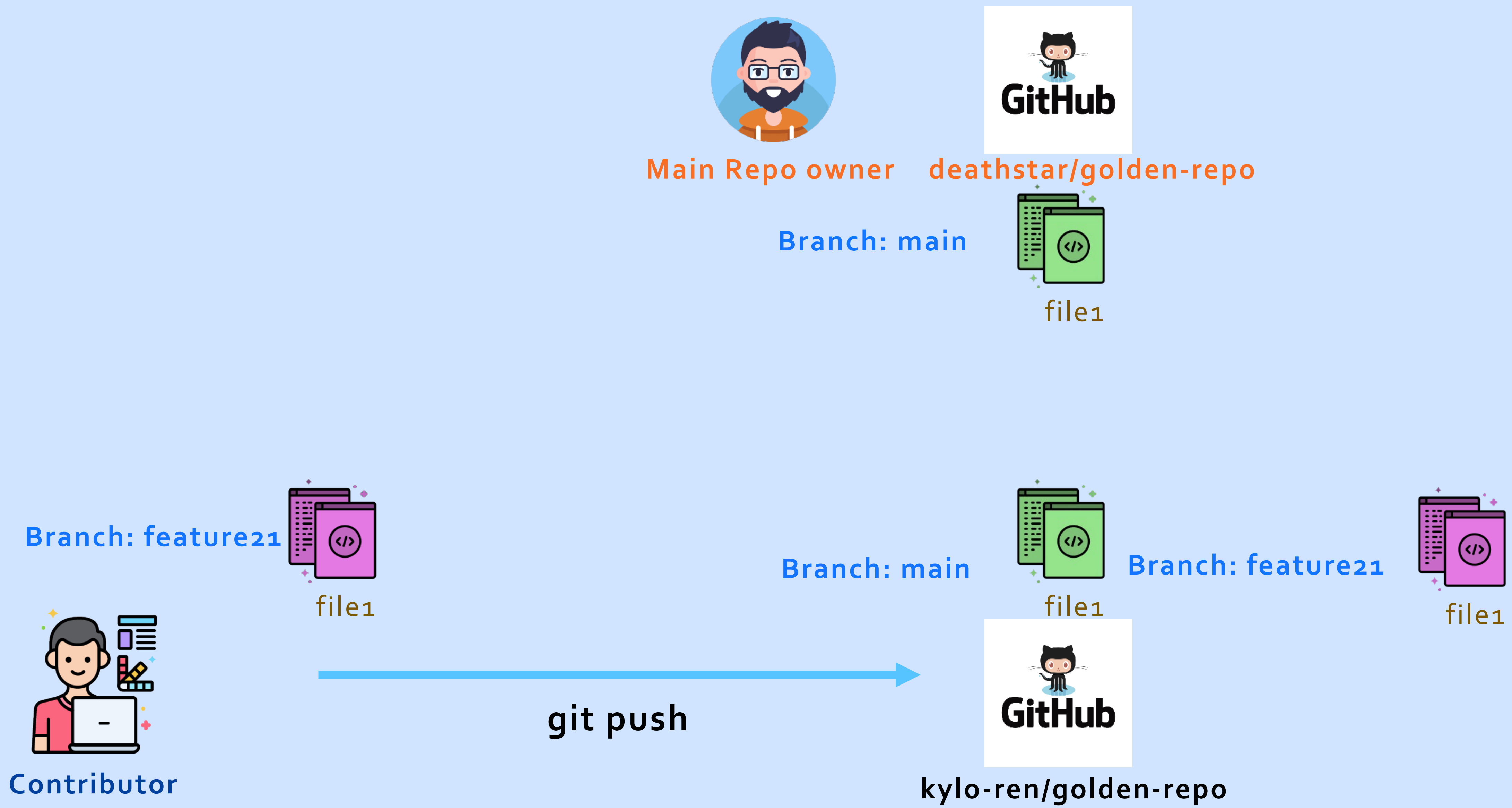
```
git branch feature21
```

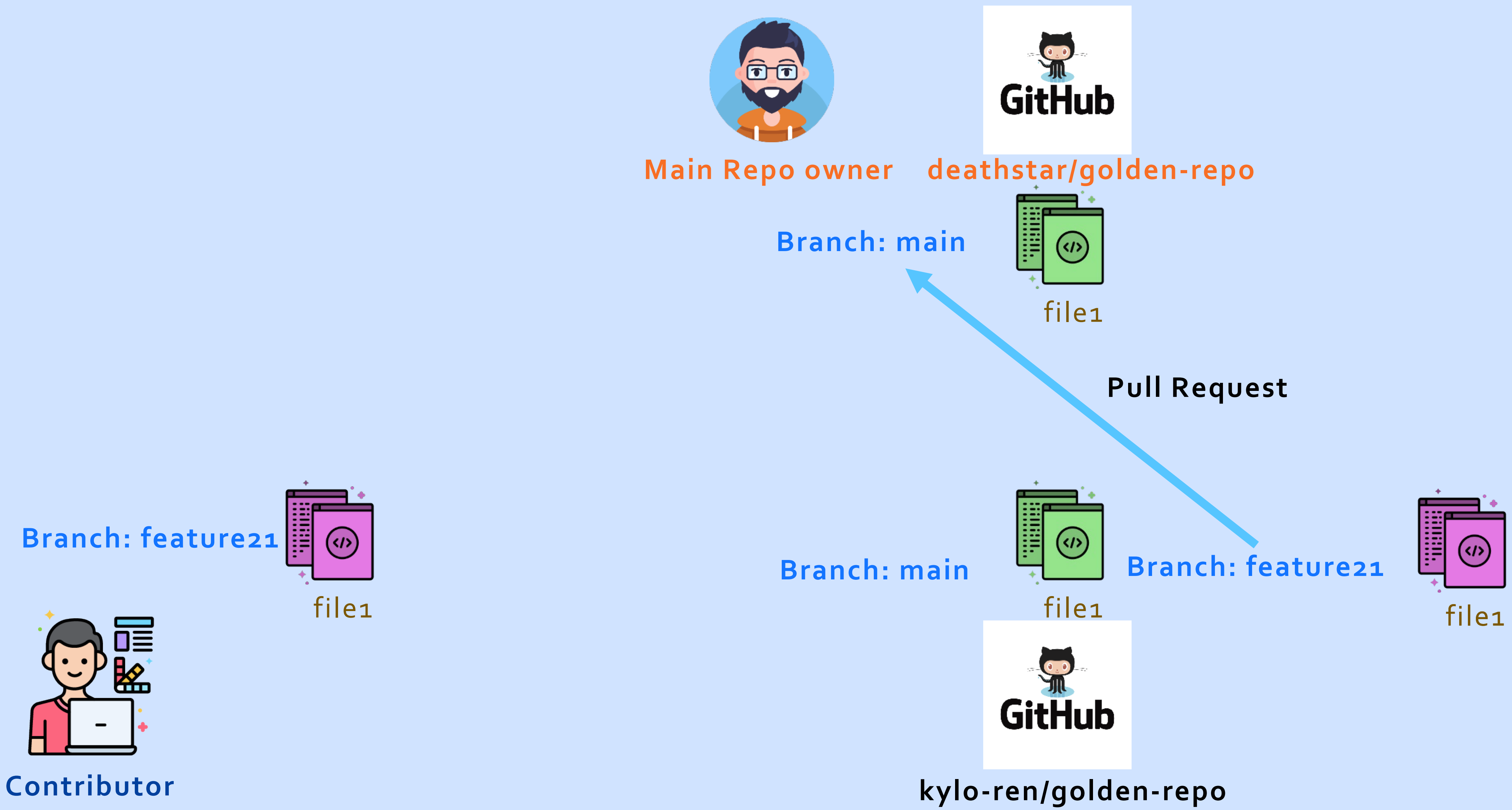


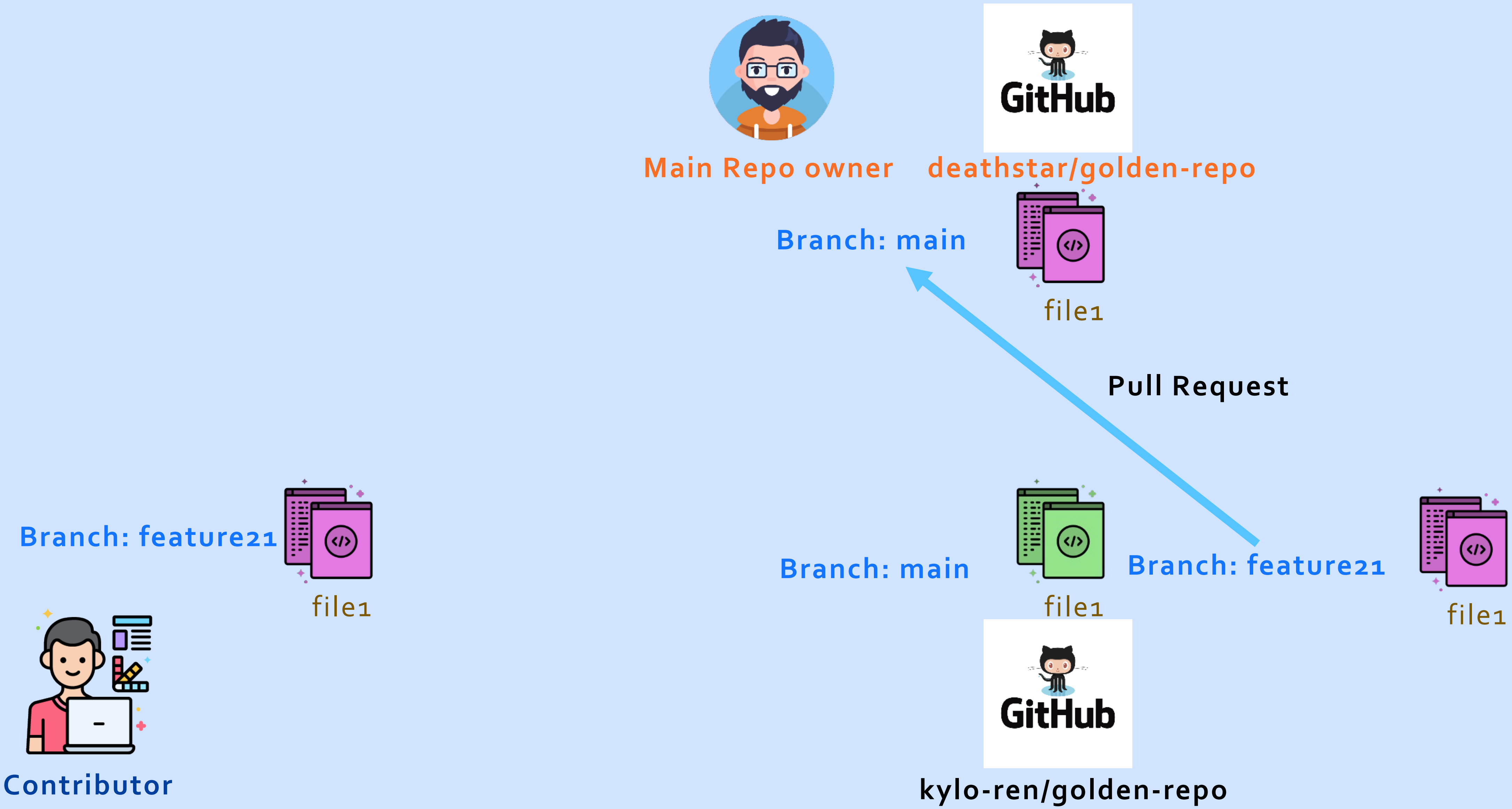
Main Repo owner
deathstar/golden-repo
Branch: main
file1
Branch: main
Branch: feature21
file1
Contributor
Branch: main
file1
kylo-ren/golden-repo


Main Repo owner
GitHub
deathstar/golden-repo
Branch: main
file1
Branch: feature21
file1
Branch: main
file1
GitHub
Contributor
kylo-ren/golden-repo


Main Repo owner
deathstar/golden-repo
Branch: main
file1
Branch: feature21
file1
Branch: main
file1
git push
Contributor
kylo-ren/golden-repo


Main Repo owner
GitHub
deathstar/golden-repo
Branch: main
file1
Branch: feature21
file1
Contributor
git push
Branch: main
file1
Branch: feature21
file1
GitHub
kylo-ren/golden-repo

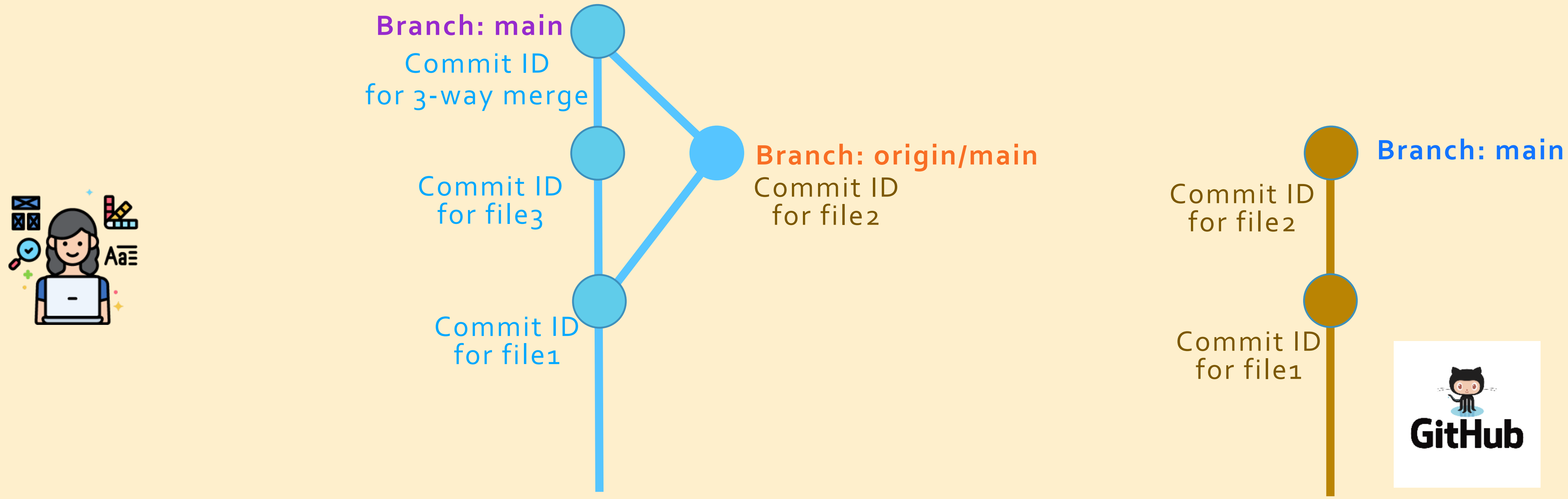

Main Repo owner
deathstar/golden-repo
Branch: main
file1
Pull Request
Branch: feature21
file1
Branch: main
file1
Branch: feature21
file1
Contributor
kylo-ren/golden-repo


Main Repo owner
deathstar/golden-repo
Branch: main
file1
Pull Request
Branch: feature21
file1
Branch: main
file1
Branch: feature21
file1
Contributor
kylo-ren/golden-repo

Git Pull vs Pull Request (PR)

Raj Saha
cloudwithraj.com
Cloud With Raj

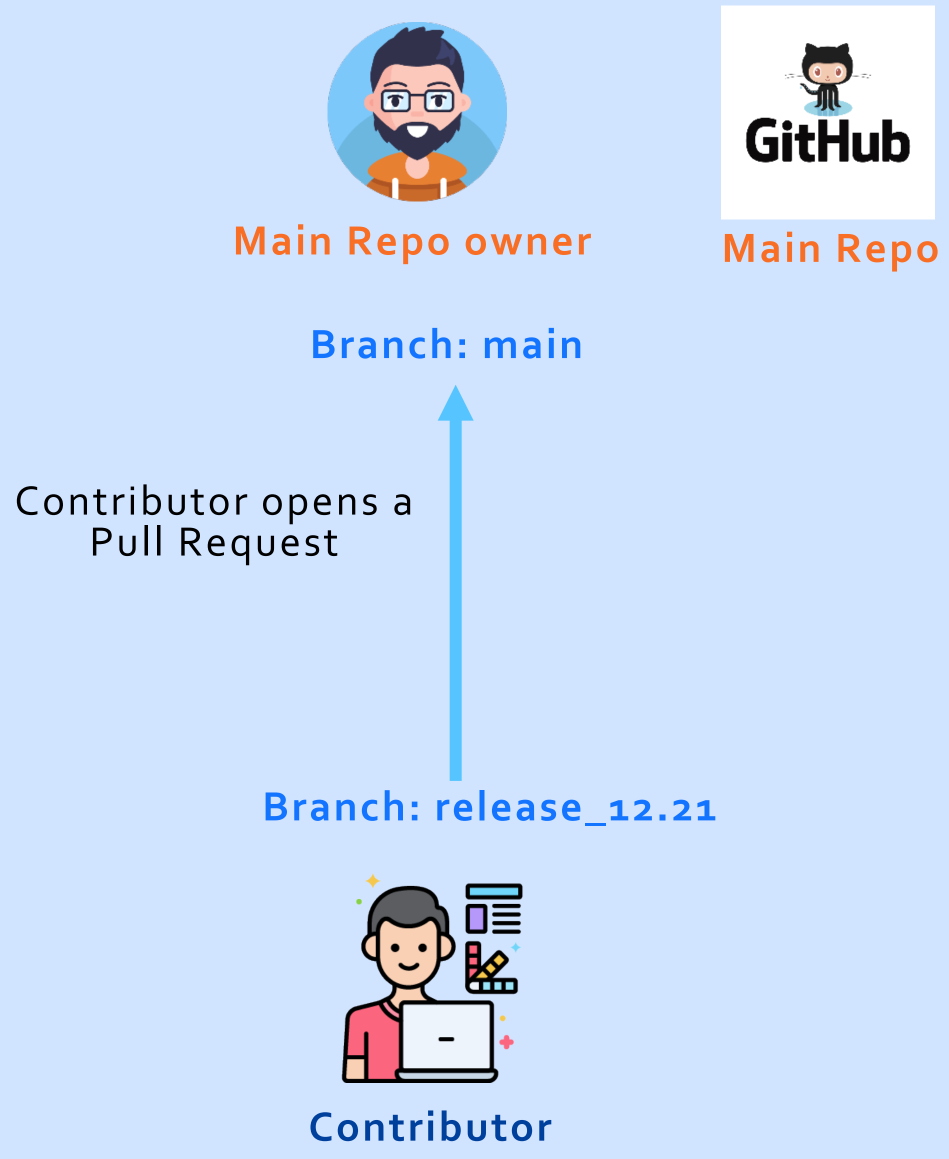
git pull = git fetch + git merge



git pull =

git fetch
git merge origin/main

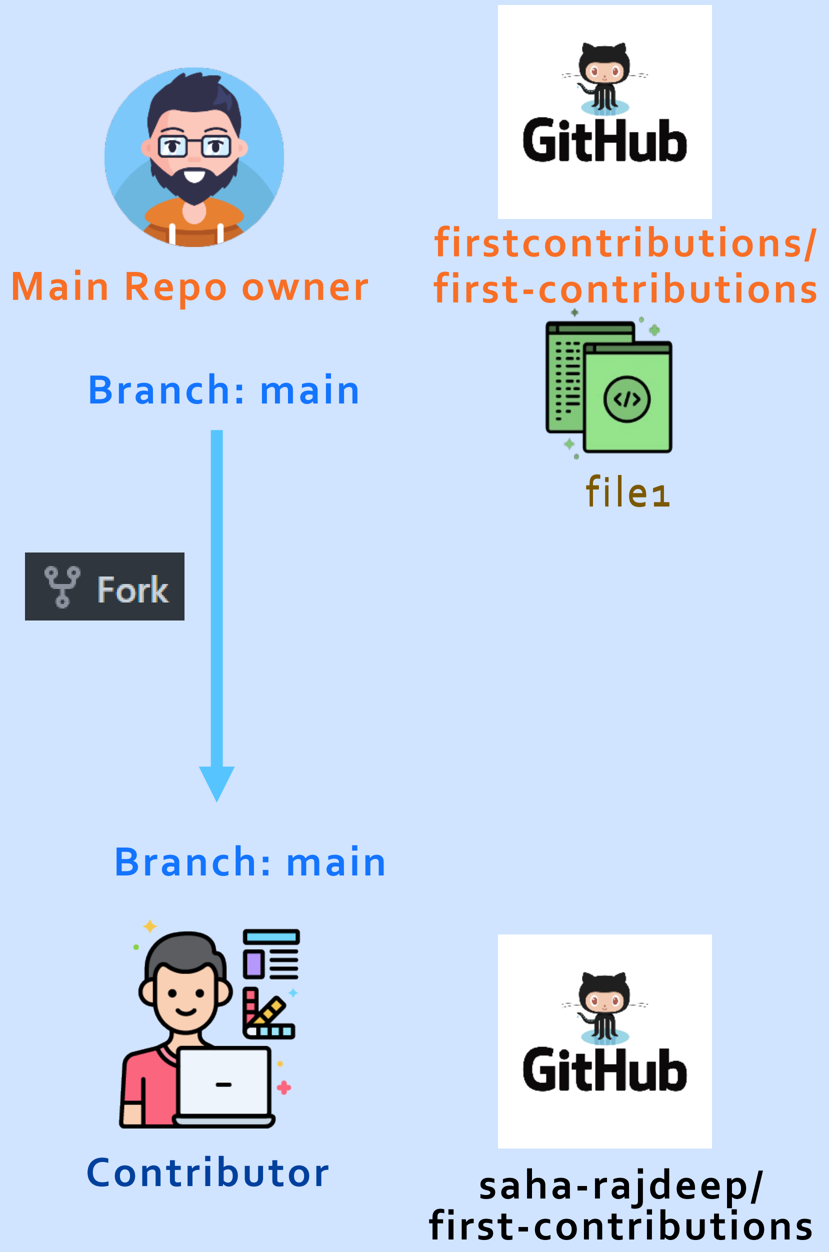
Pull Request (PR)

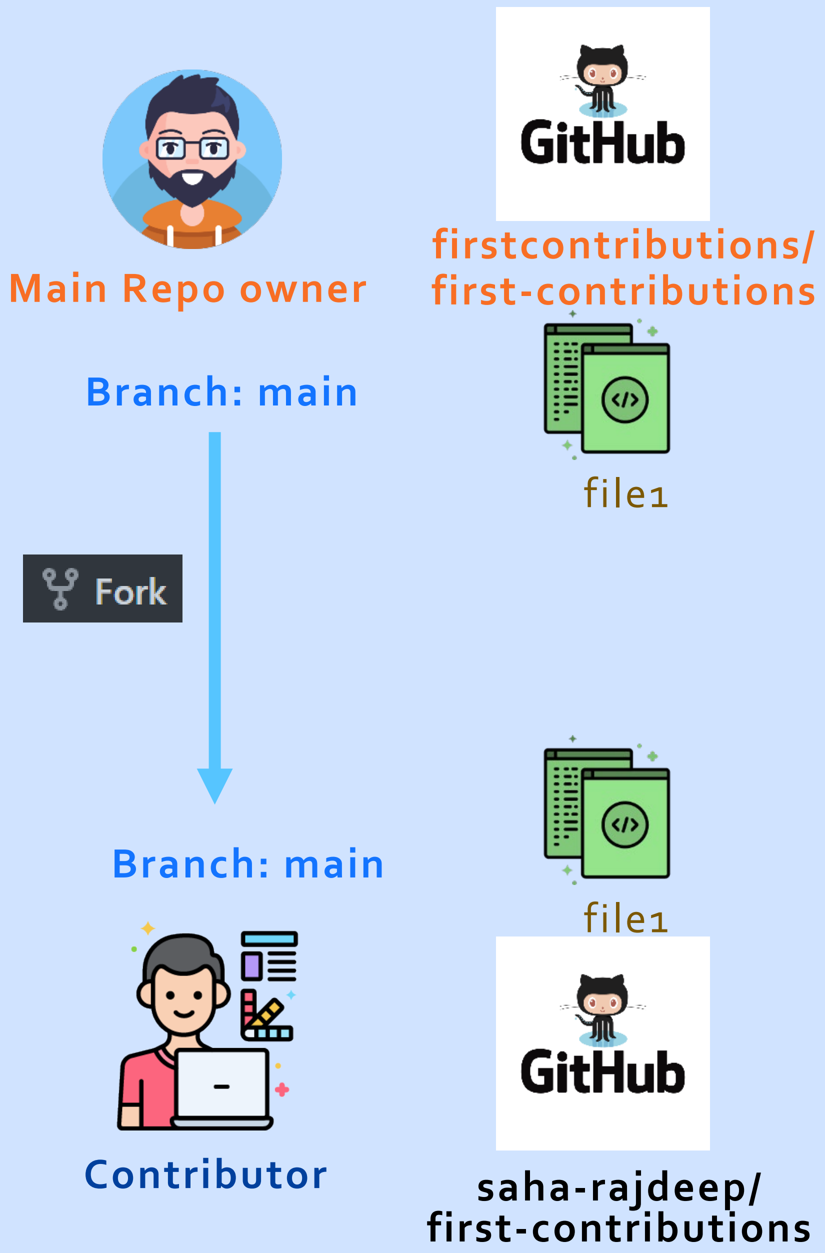


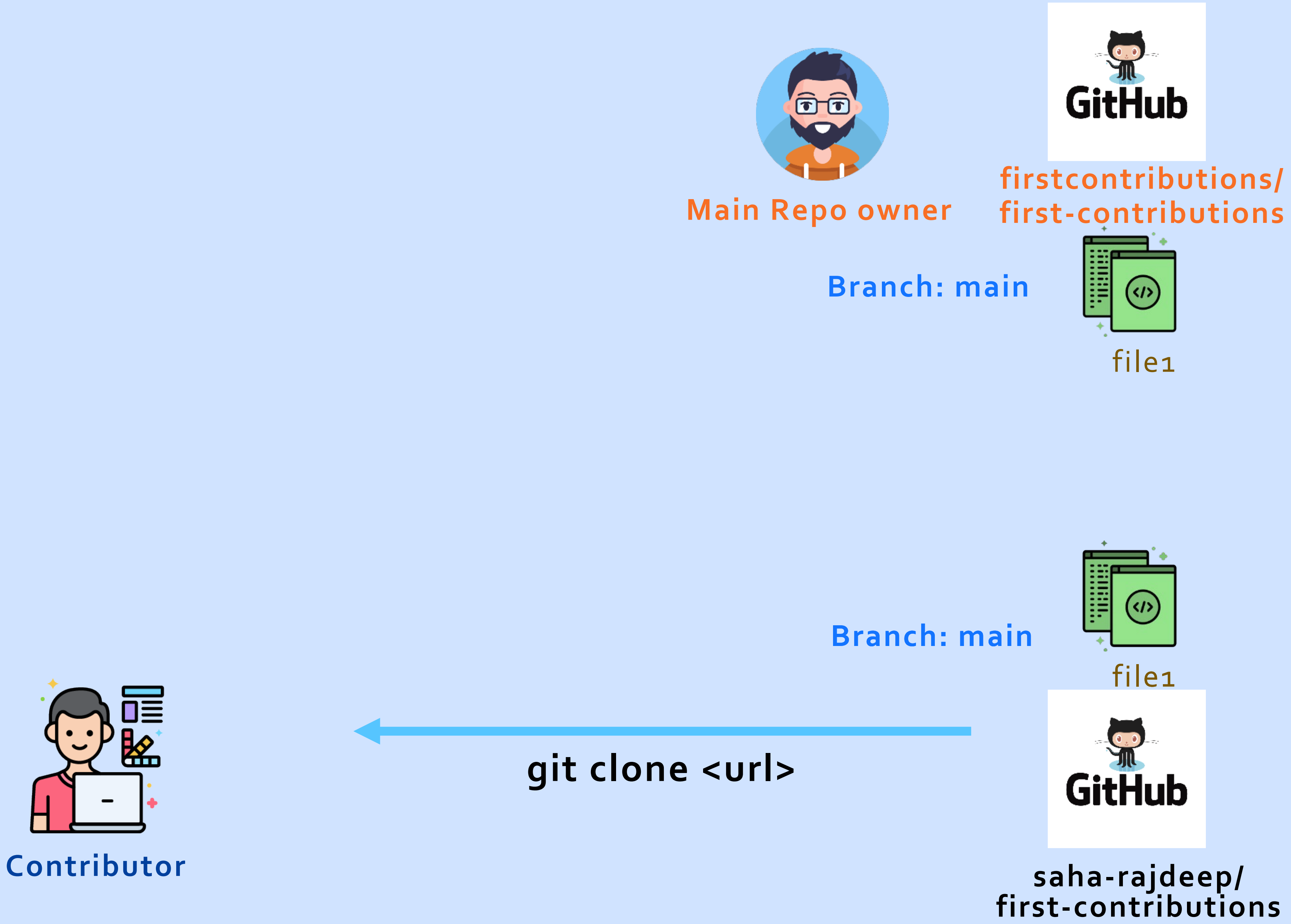
- Request to "Pull" the changes from one branch to another
  - Can be from branch in a different repo
- Pull Request allows collaborators of the project to review, comment, and update codes
- Pull Request is GitHub specific term
- Git Pull is Git based action

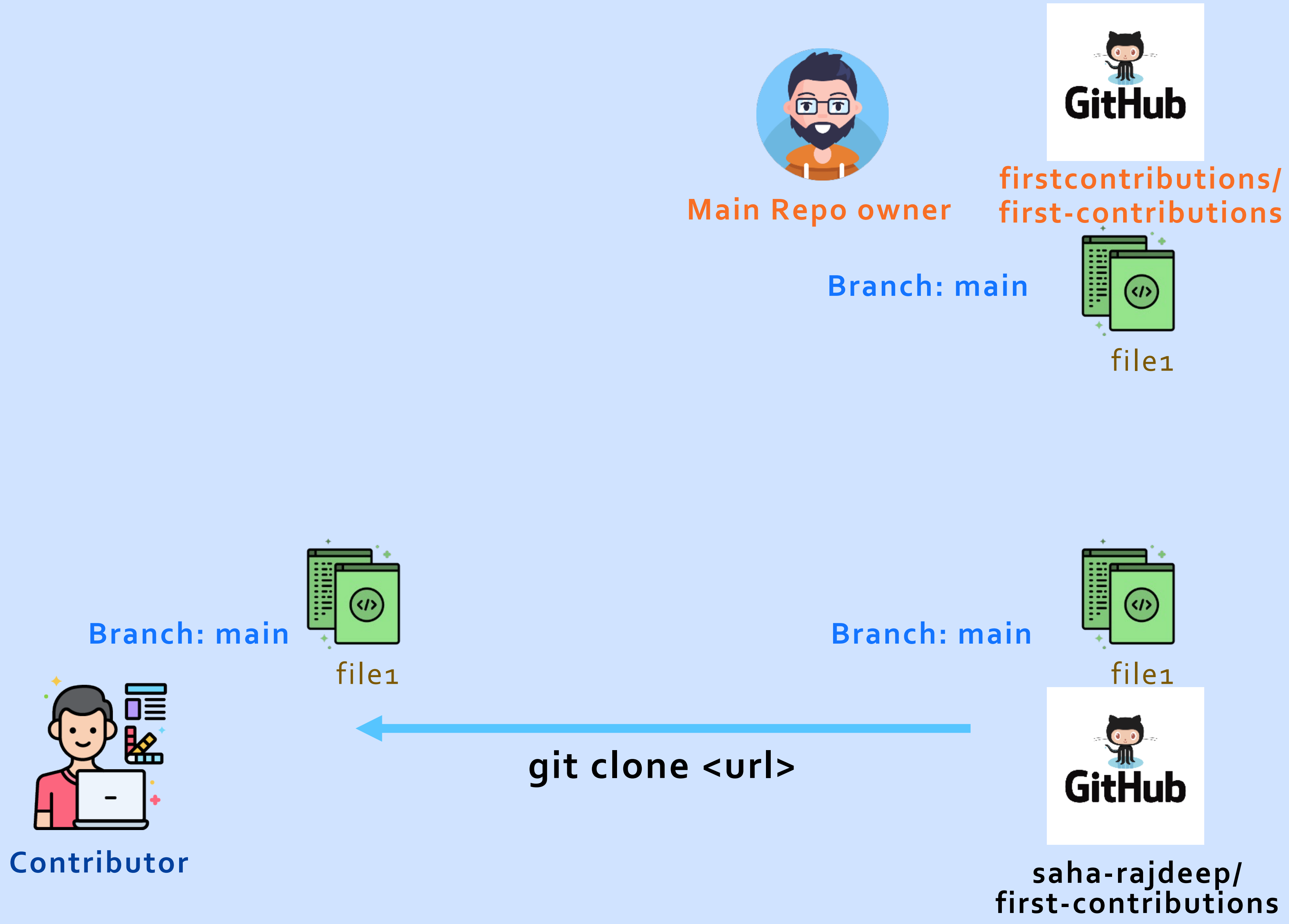
Keeping Forked Repo Up To Date

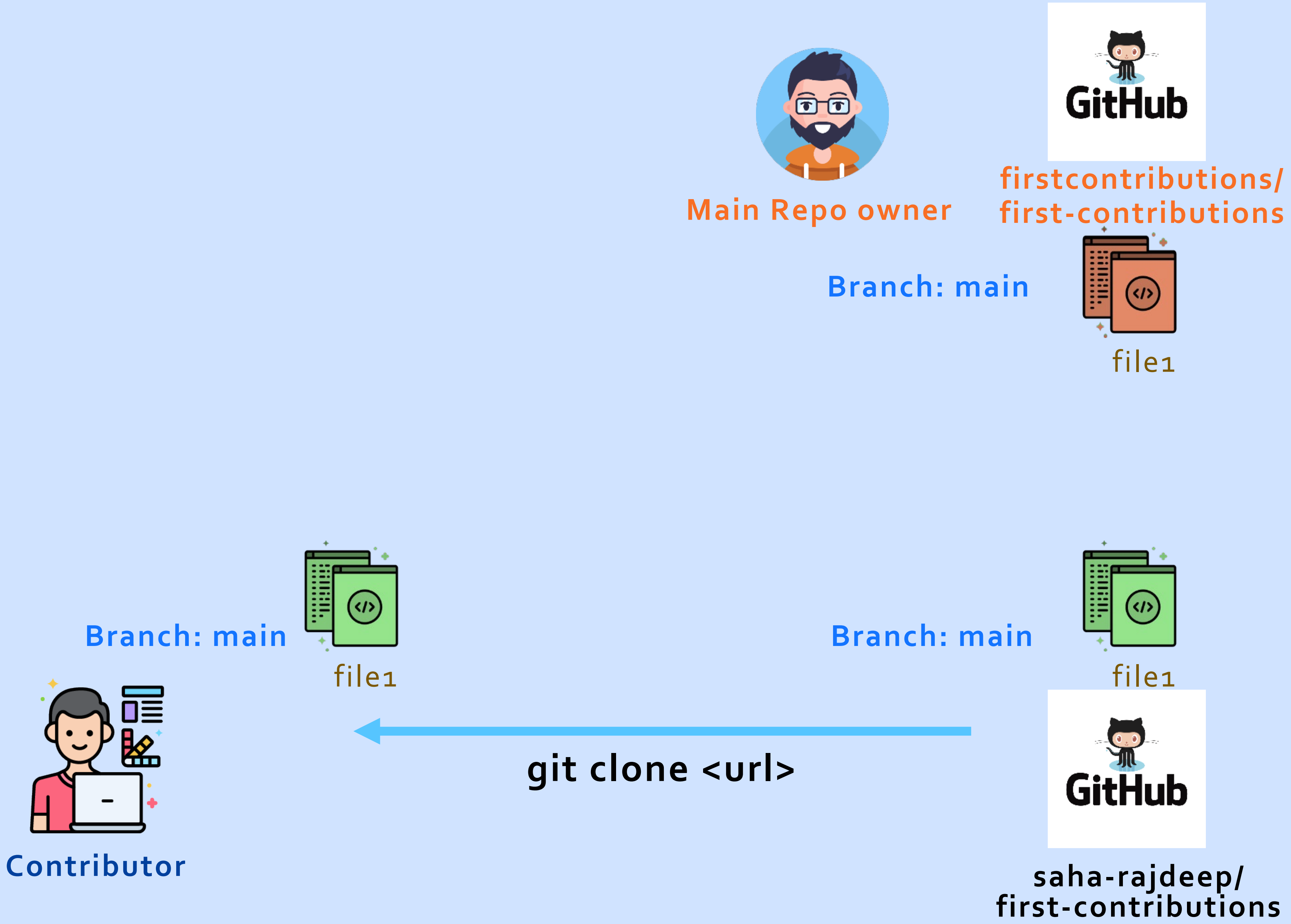
Raj Saha
cloudwithraj.com
Cloud With Raj


GitHub
Main Repo owner
firstcontributions/
first-contributions
Branch: main
file1
Fork
Branch: main
GitHub
Contributor
saha-rajdeep/
first-contributions

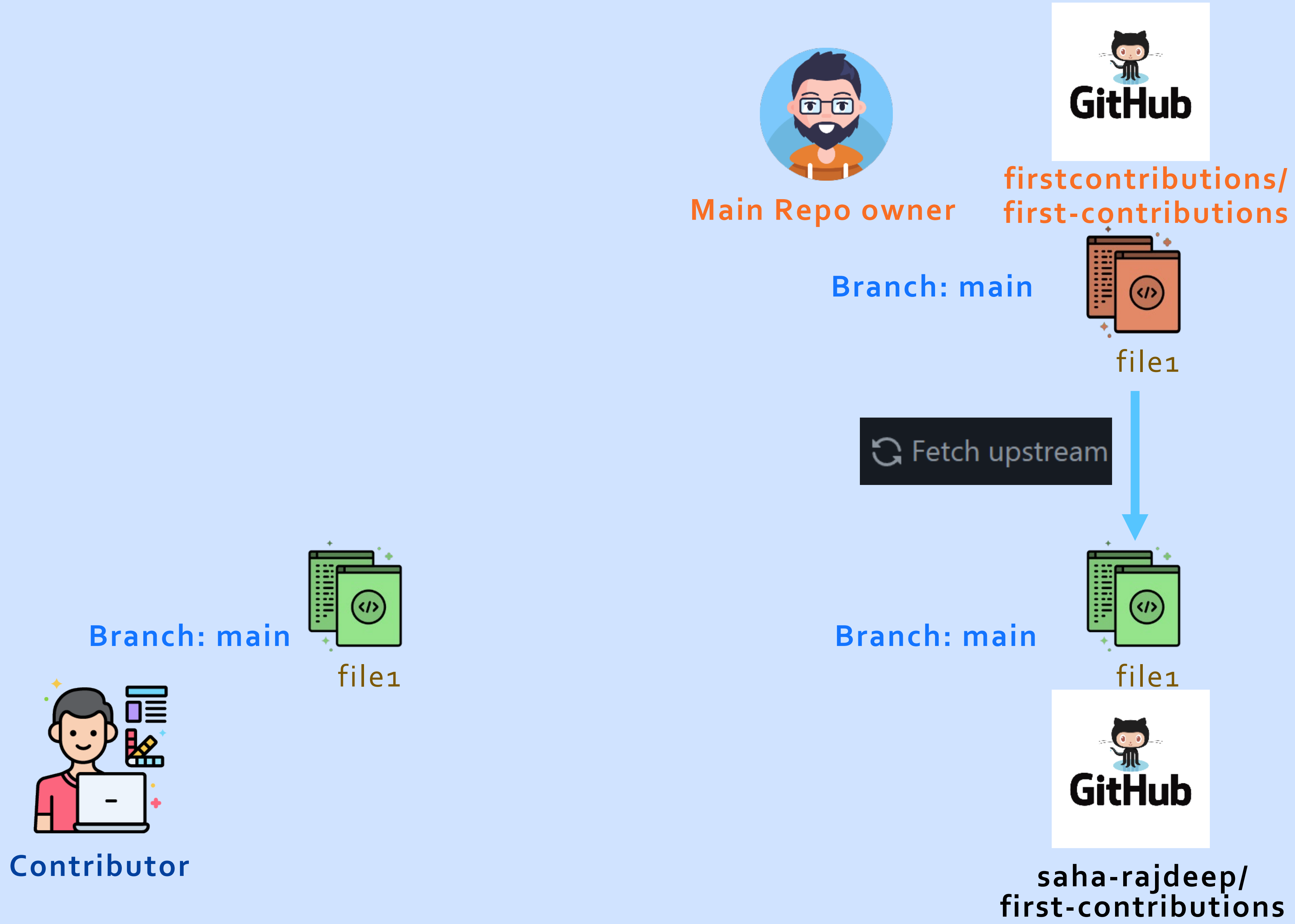

Main Repo owner
firstcontributions/
first-contributions
Branch: main
file1
Fork
Branch: main
file1
Contributor
saha-rajdeep/
first-contributions

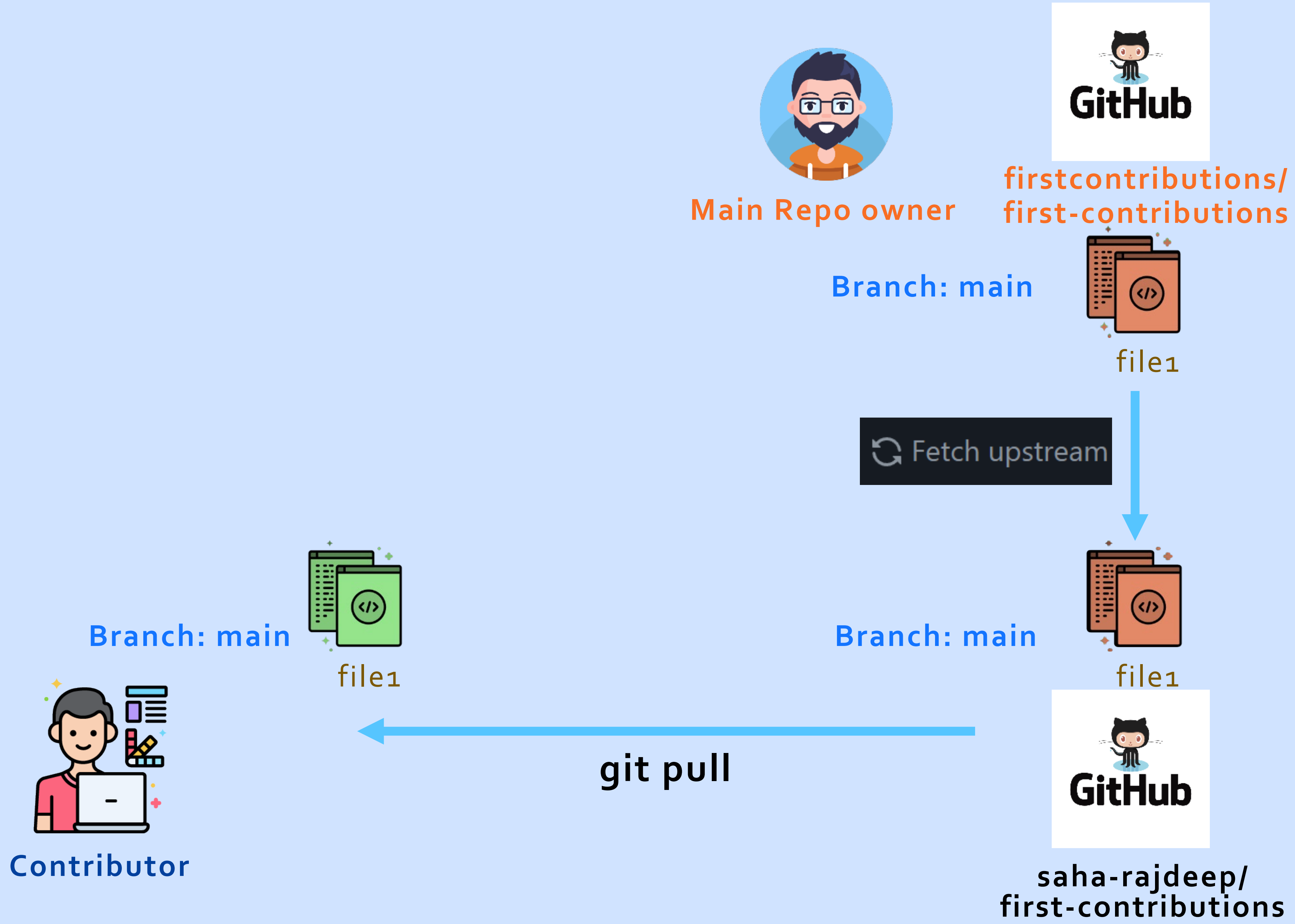

Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Branch: main
file1
GitHub
saha-rajdeep/
first-contributions
git clone <url>
Contributor

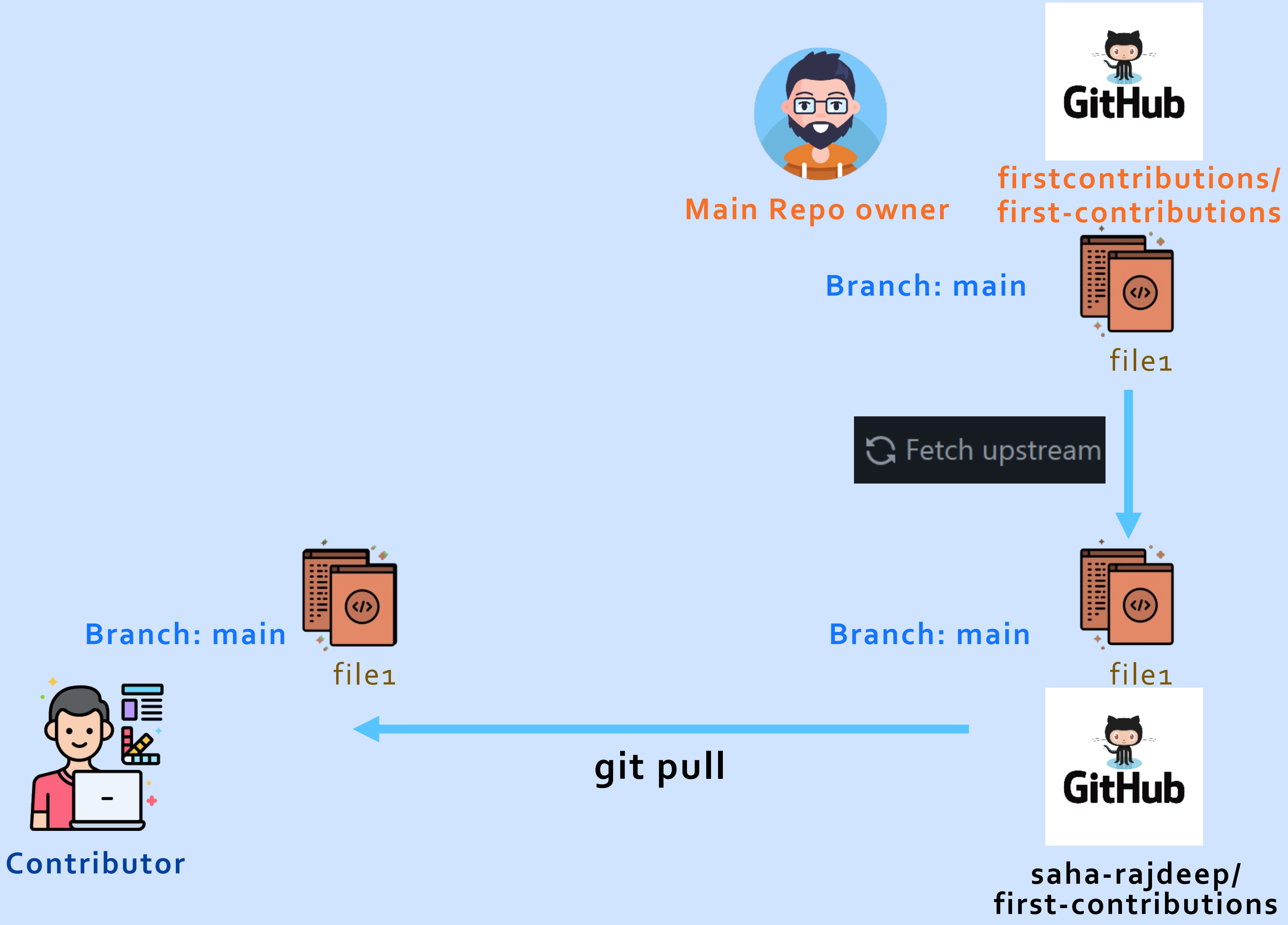

Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Branch: main
file1
Contributor
Branch: main
file1
git clone <url>
GitHub
saha-rajdeep/
first-contributions

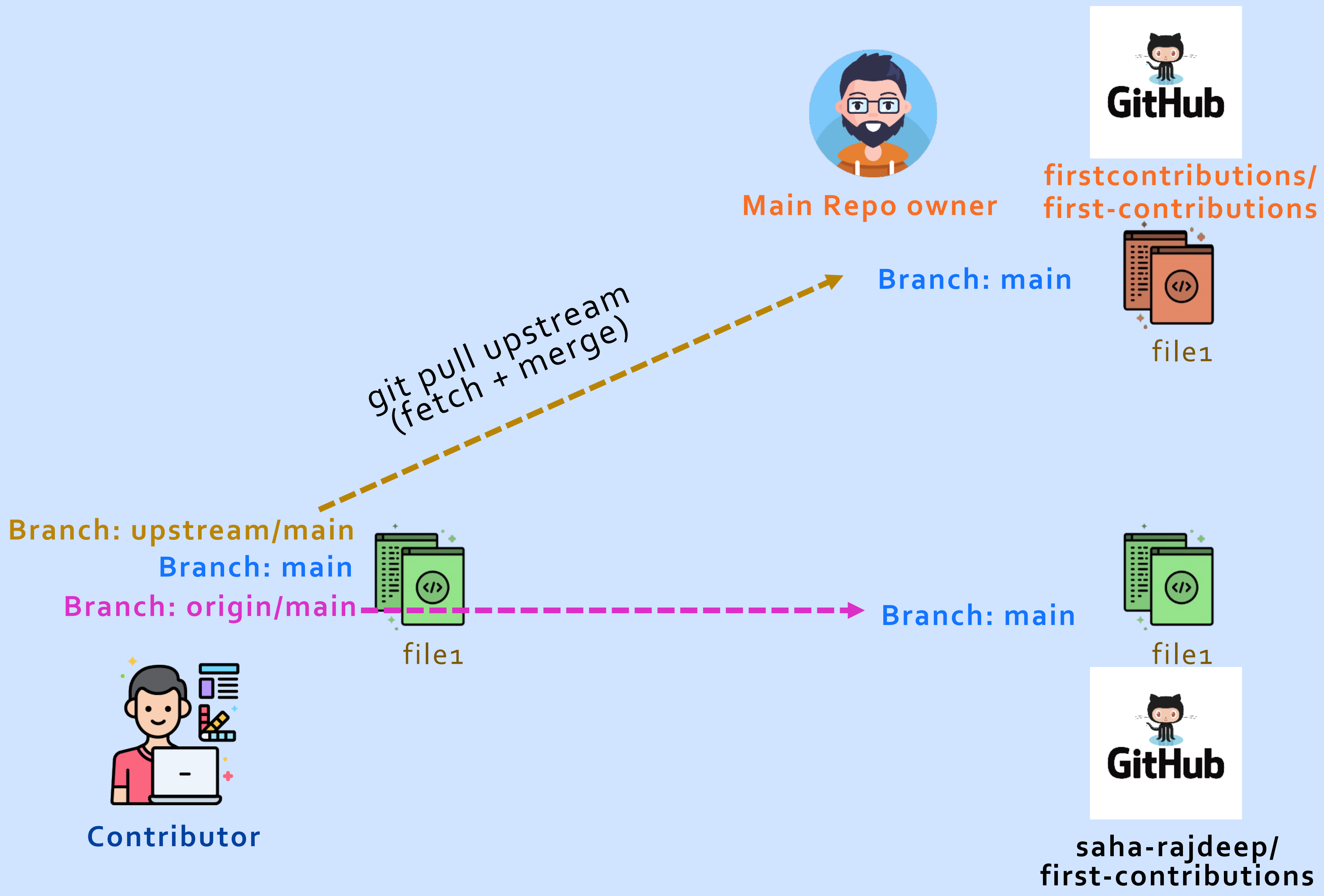


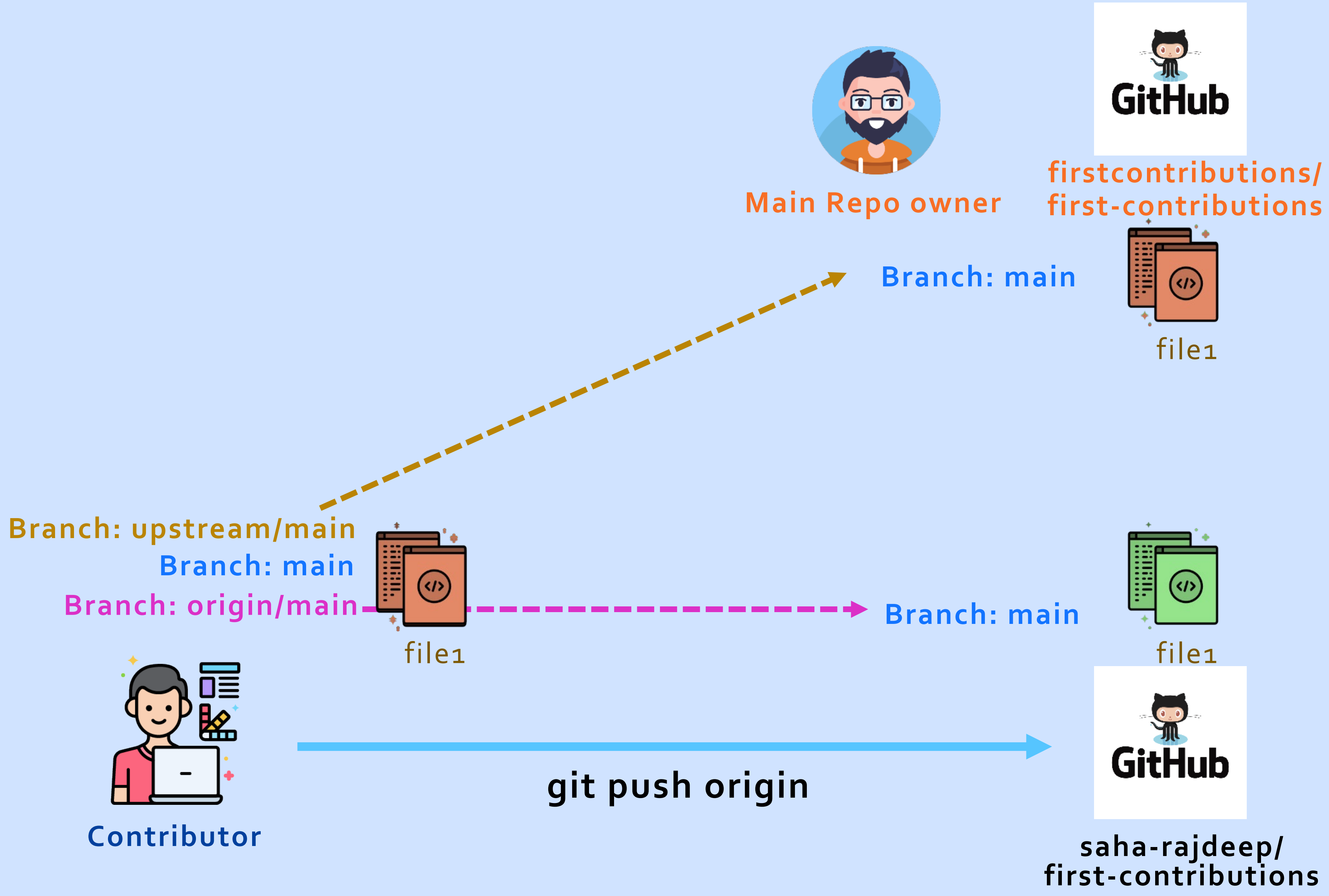
Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Branch: main
file1
Branch: main
file1
git clone <url>
GitHub
Contributor
saha-rajdeep/
first-contributions

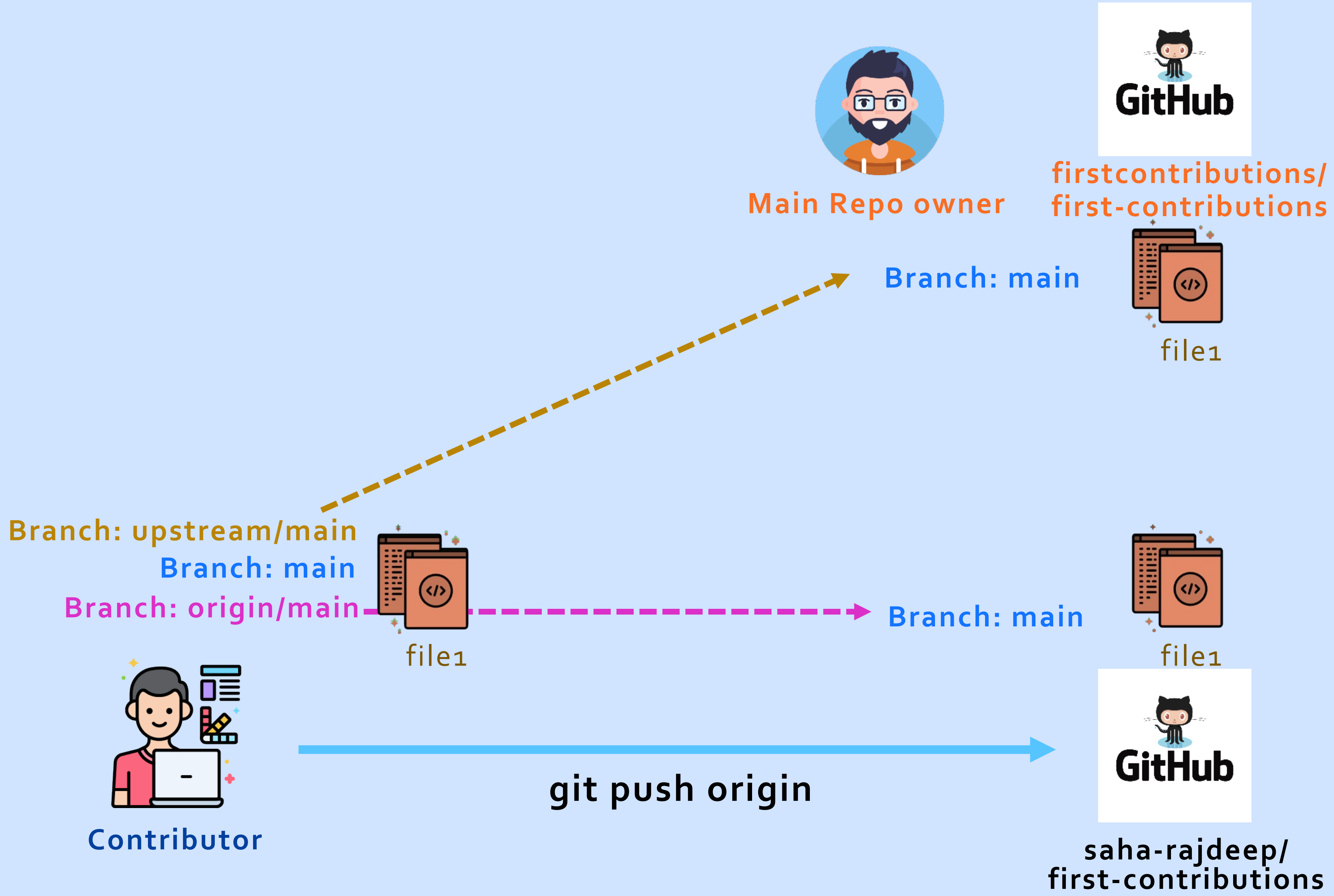

Main Repo owner
firstcontributions/
first-contributions
Branch: main
file1
Fetch upstream
Branch: main
file1
Branch: main
file1
Contributor
saha-rajdeep/
first-contributions


Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Fetch upstream
Branch: main
file1
Branch: main
file1
git pull
GitHub
saha-rajdeep/
first-contributions
Contributor


Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Fetch upstream
Branch: main
file1
Branch: main
file1
git pull
GitHub
saha-rajdeep/
first-contributions
Contributor

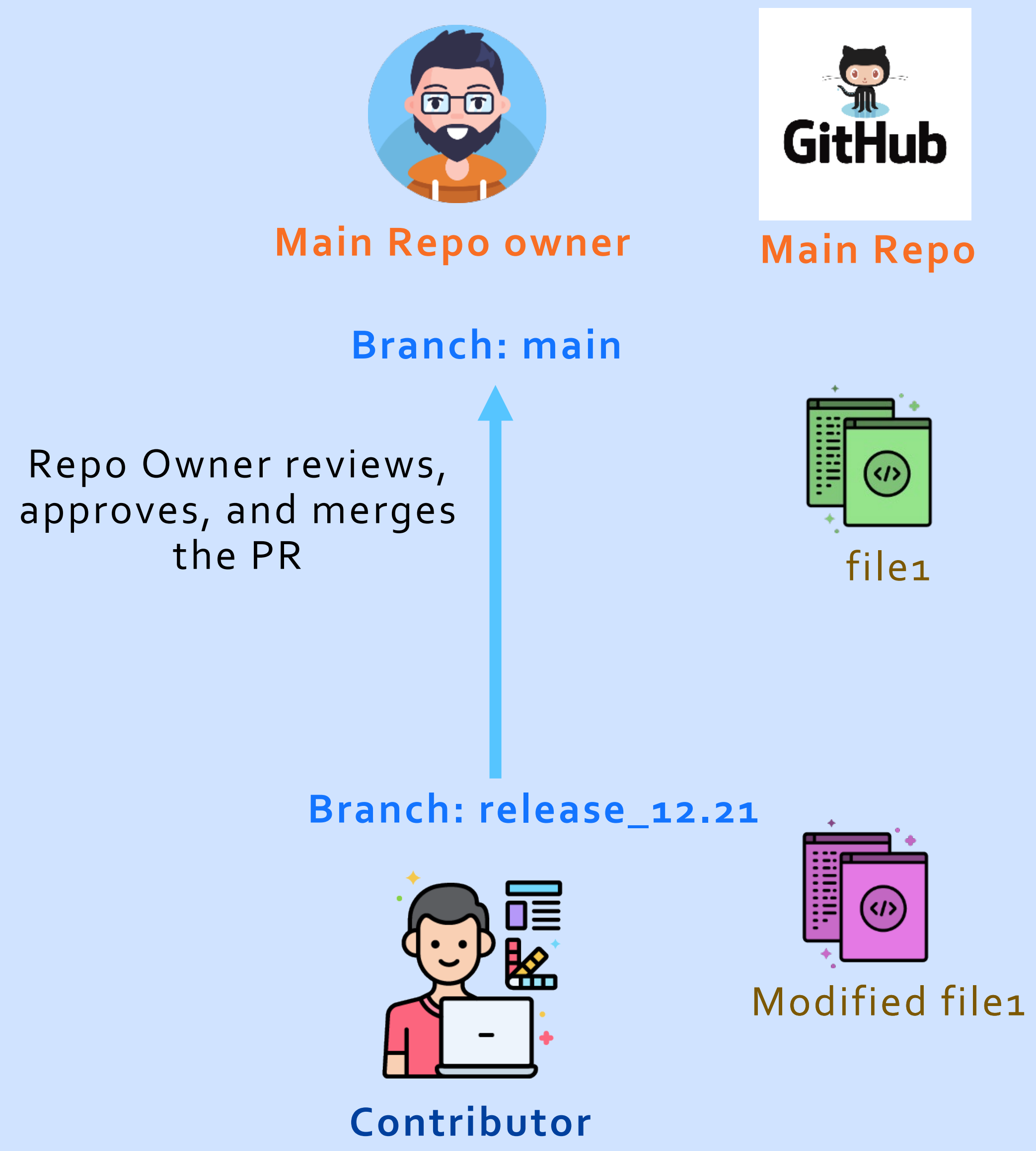

Main Repo owner
firstcontributions/
first-contributions
Branch: main
file1
git pull upstream
(fetch + merge)
Branch: upstream/main
Branch: main
Branch: origin/main
file1
Branch: main
file1
saha-rajdeep/
first-contributions
Contributor

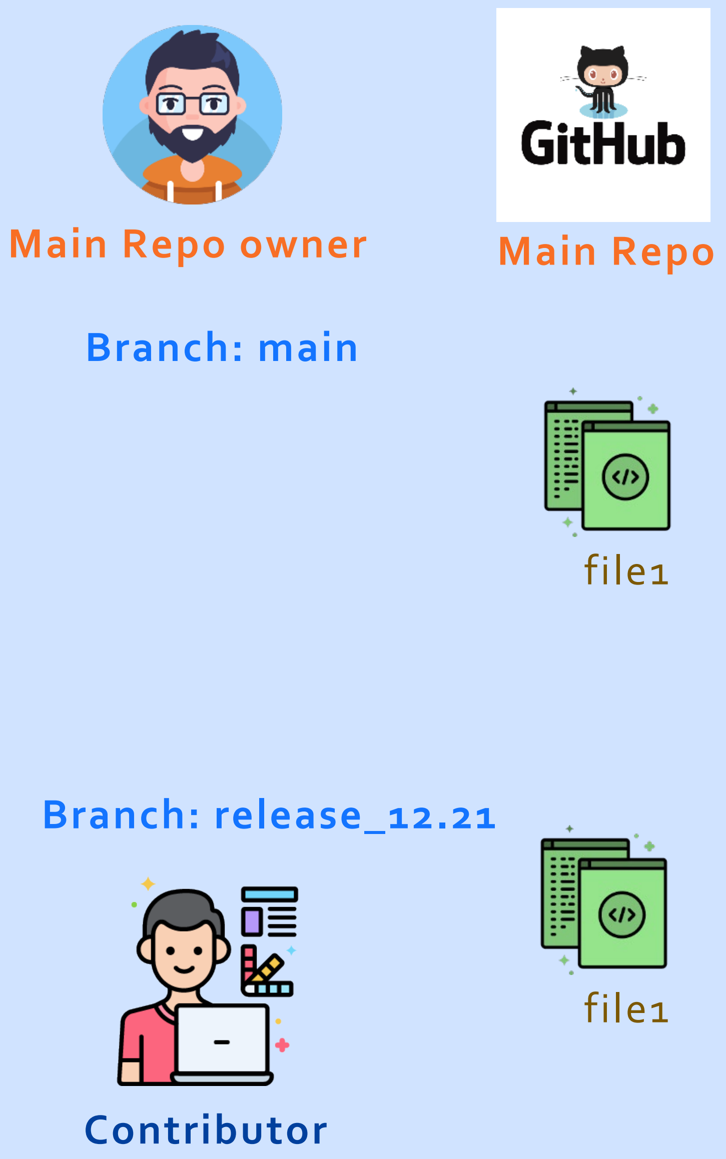

Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Branch: upstream/main
Branch: main
Branch: origin/main
file1
Branch: main
file1
GitHub
Contributor
git push origin
saha-rajdeep/
first-contributions

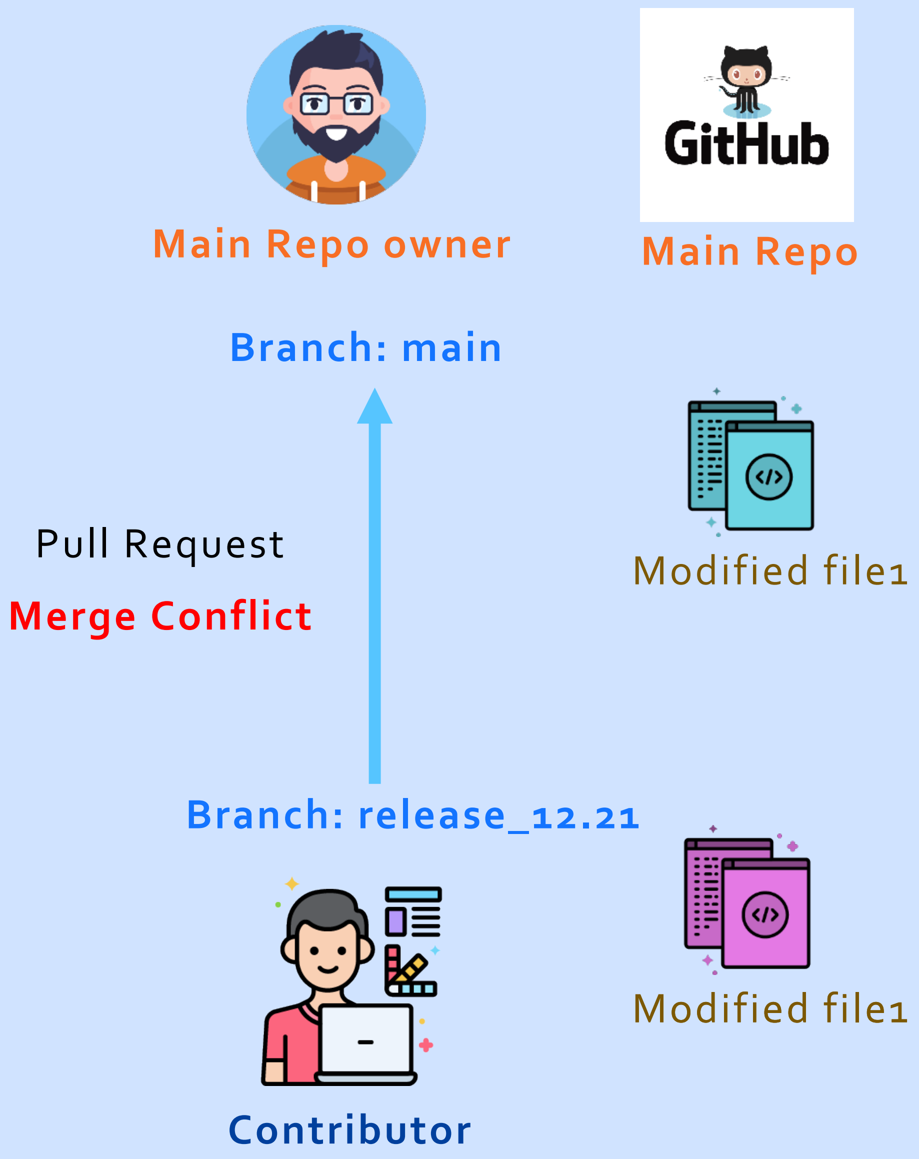

Main Repo owner
GitHub
firstcontributions/
first-contributions
Branch: main
file1
Branch: upstream/main
Branch: main
Branch: origin/main
file1
Branch: main
file1
Contributor
git push origin
GitHub
saha-rajdeep/
first-contributions

Merge Conflict

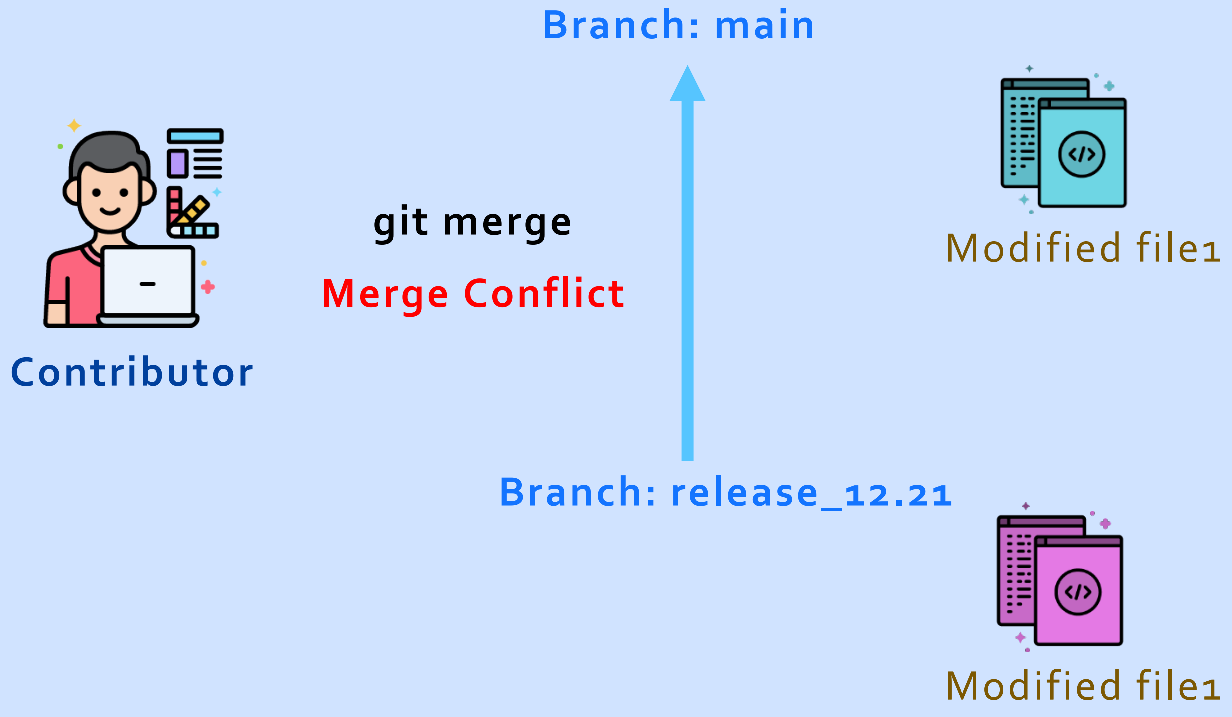
Raj Saha
cloudwithraj.com
Cloud With Raj


Main Repo owner
GitHub
Main Repo
Branch: main
Repo Owner reviews, approves, and merges the PR
file1
Branch: release_12.21
Modified file1
Contributor


Main Repo owner
GitHub
Main Repo
Branch: main
file1
Branch: release_12.21
file1
Contributor


Main Repo owner
GitHub
Main Repo
Branch: main
Modified file1
Pull Request
Merge Conflict
Branch: release_12.21
Modified file1
Contributor

Merge Conflict in Local



Markdown

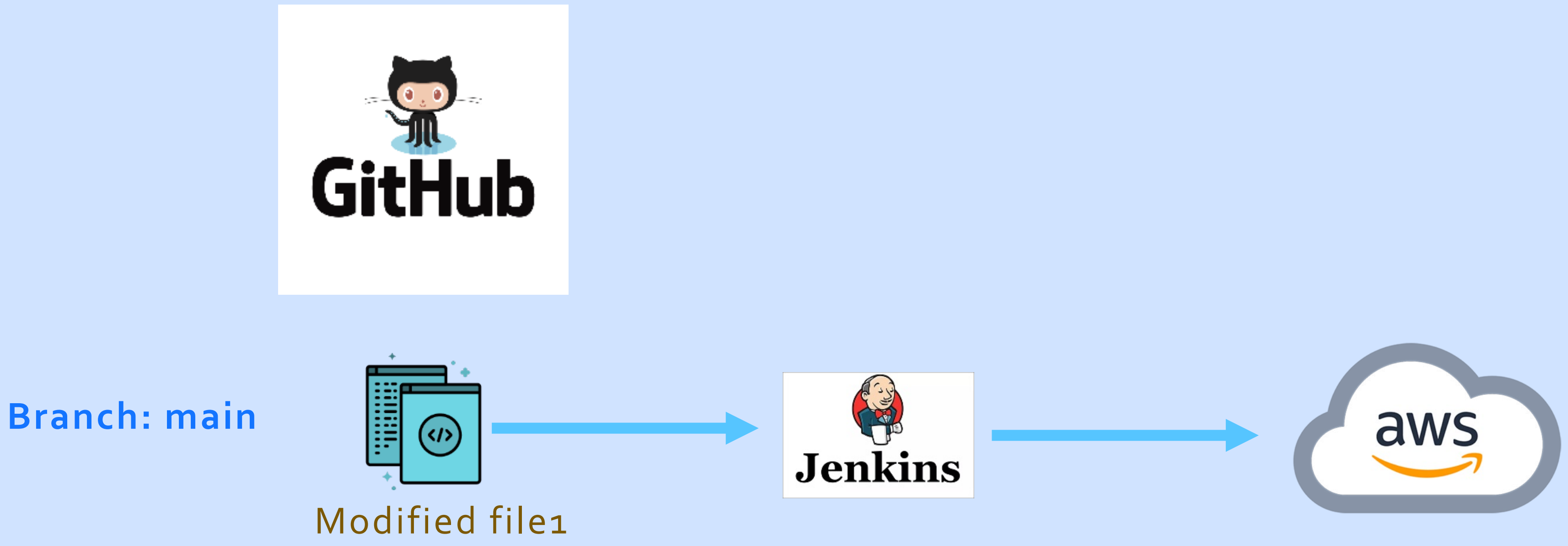
Raj Saha
cloudwithraj.com
Cloud With Raj

Markdown

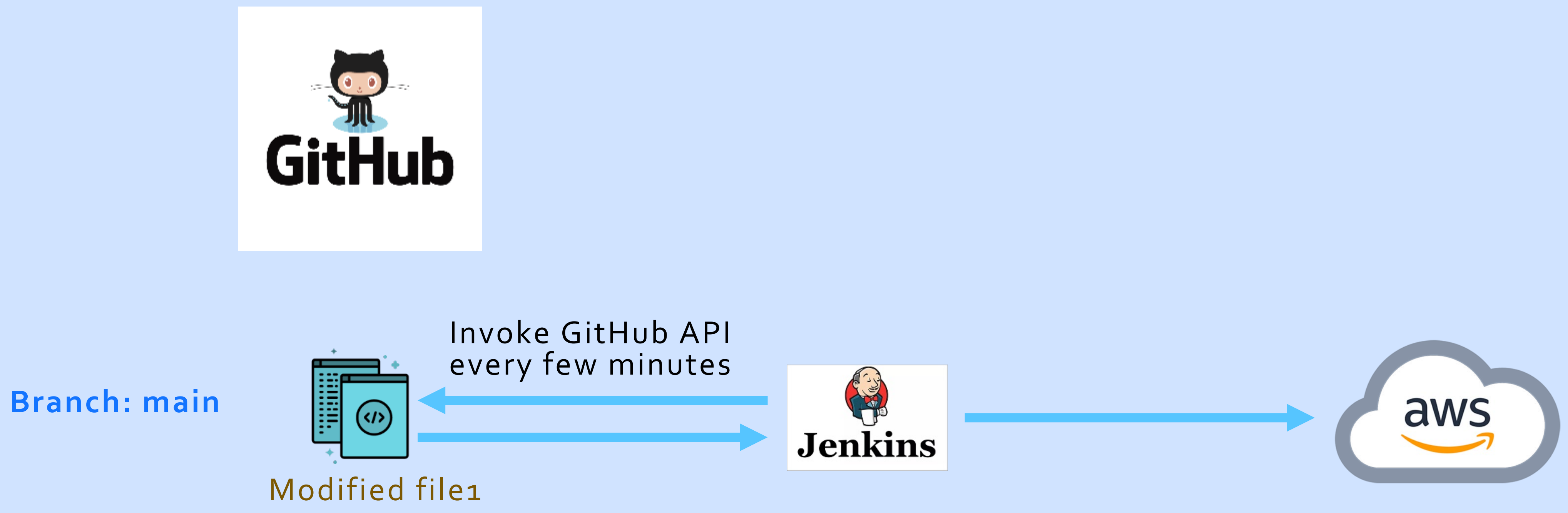
- Add formatting element to plaintext documents
- Default for showing readme in GitHub
- Why not use Word/ WYSIWYG editor?

GitHub Webhook vs API

Raj Saha
cloudwithraj.com
Cloud With Raj

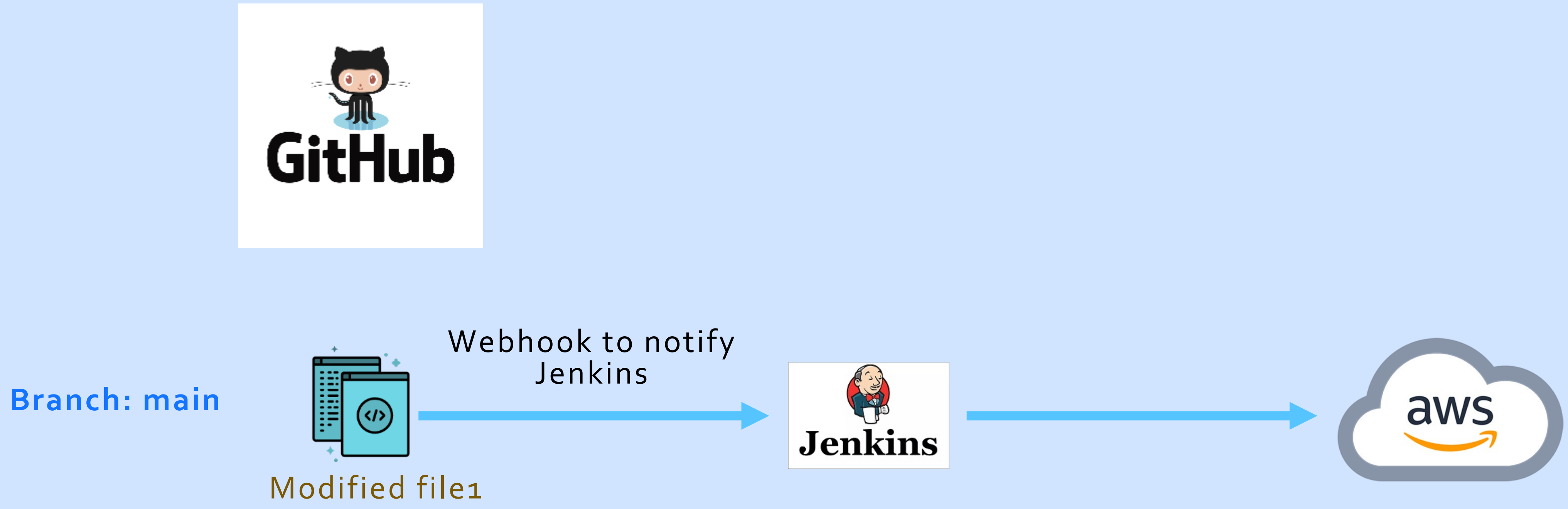

Branch: main
Modified file1

Calling API



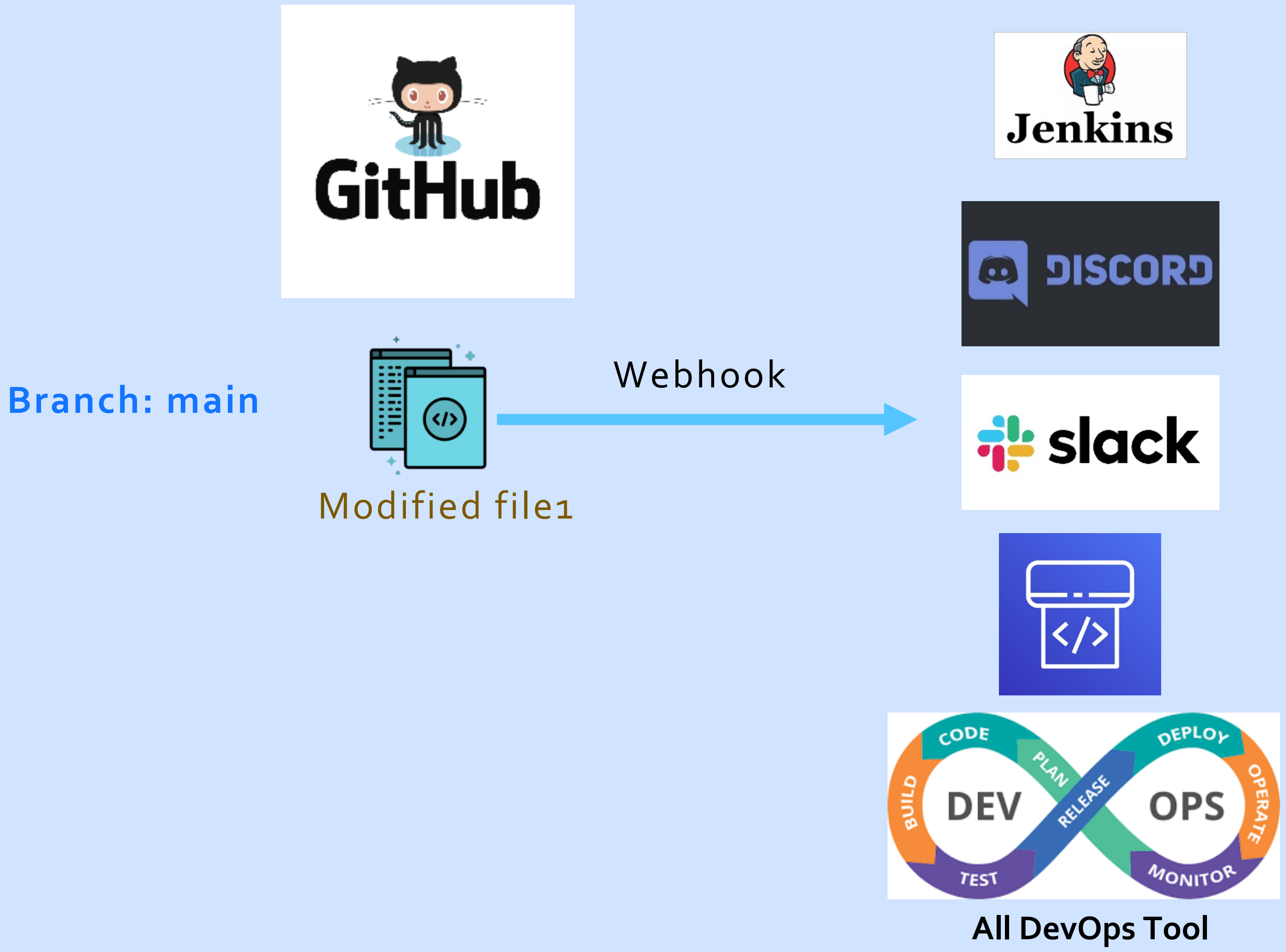
- Most of the times API will return stagnant data
- GitHub server will be bombarded
- Apps will exceed API limits

Webhook



- GitHub will do a POST call to your app if repo changes
- Lightweight
- Realtime

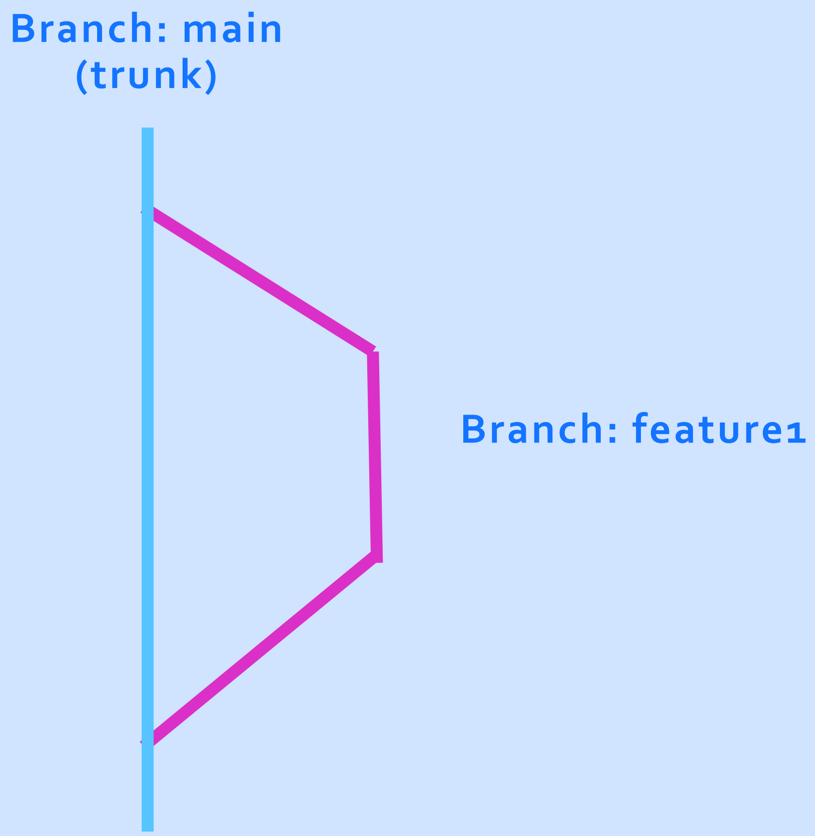
Implementation



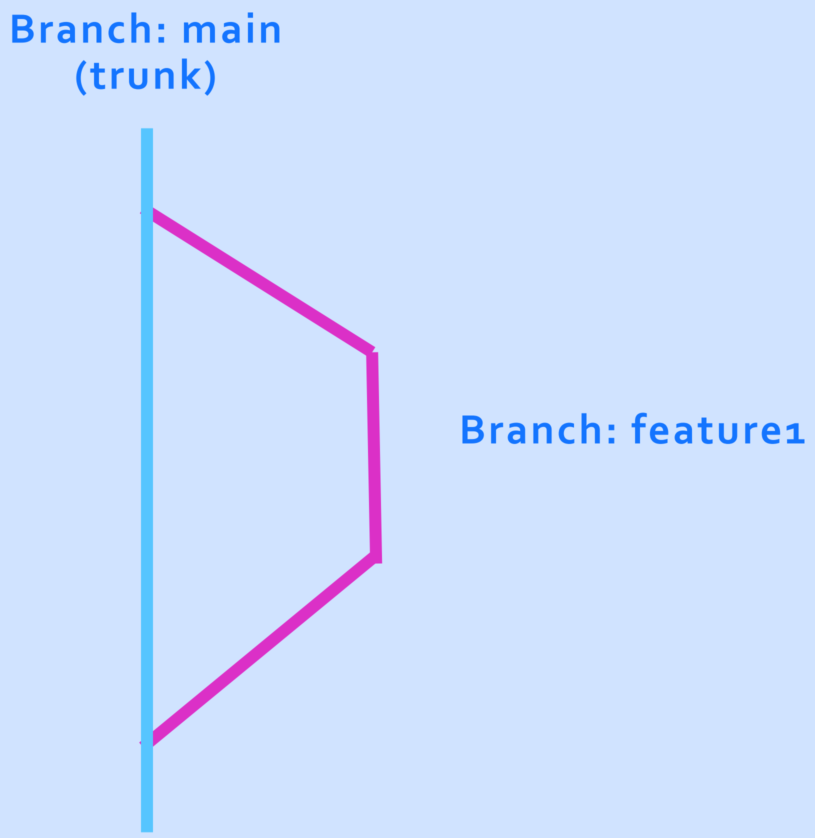
Git Branching Strategy: Trunk vs Git Flow

Raj Saha
cloudwithraj.com
Cloud With Raj

Trunk Based Development

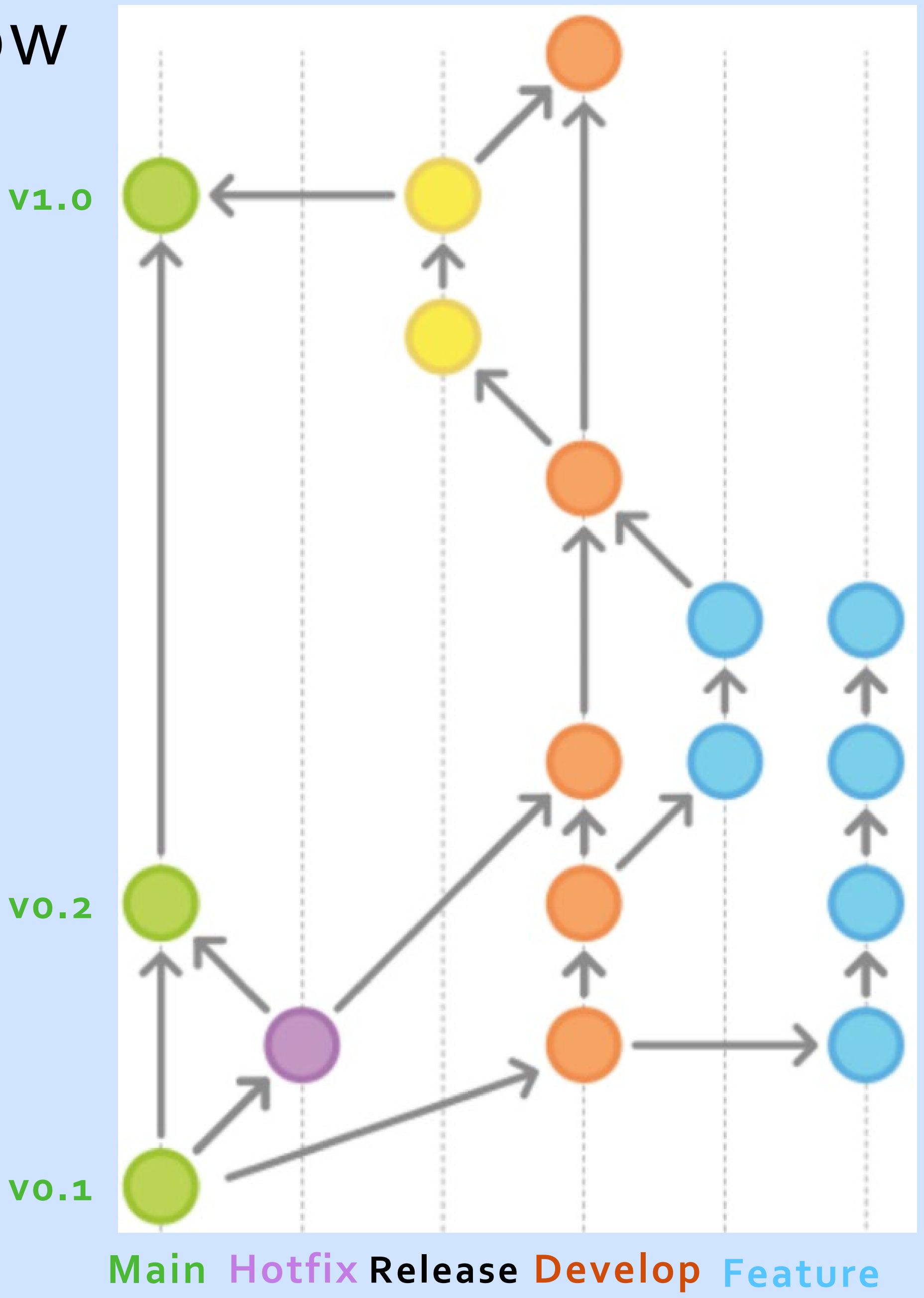


Trunk Based Development



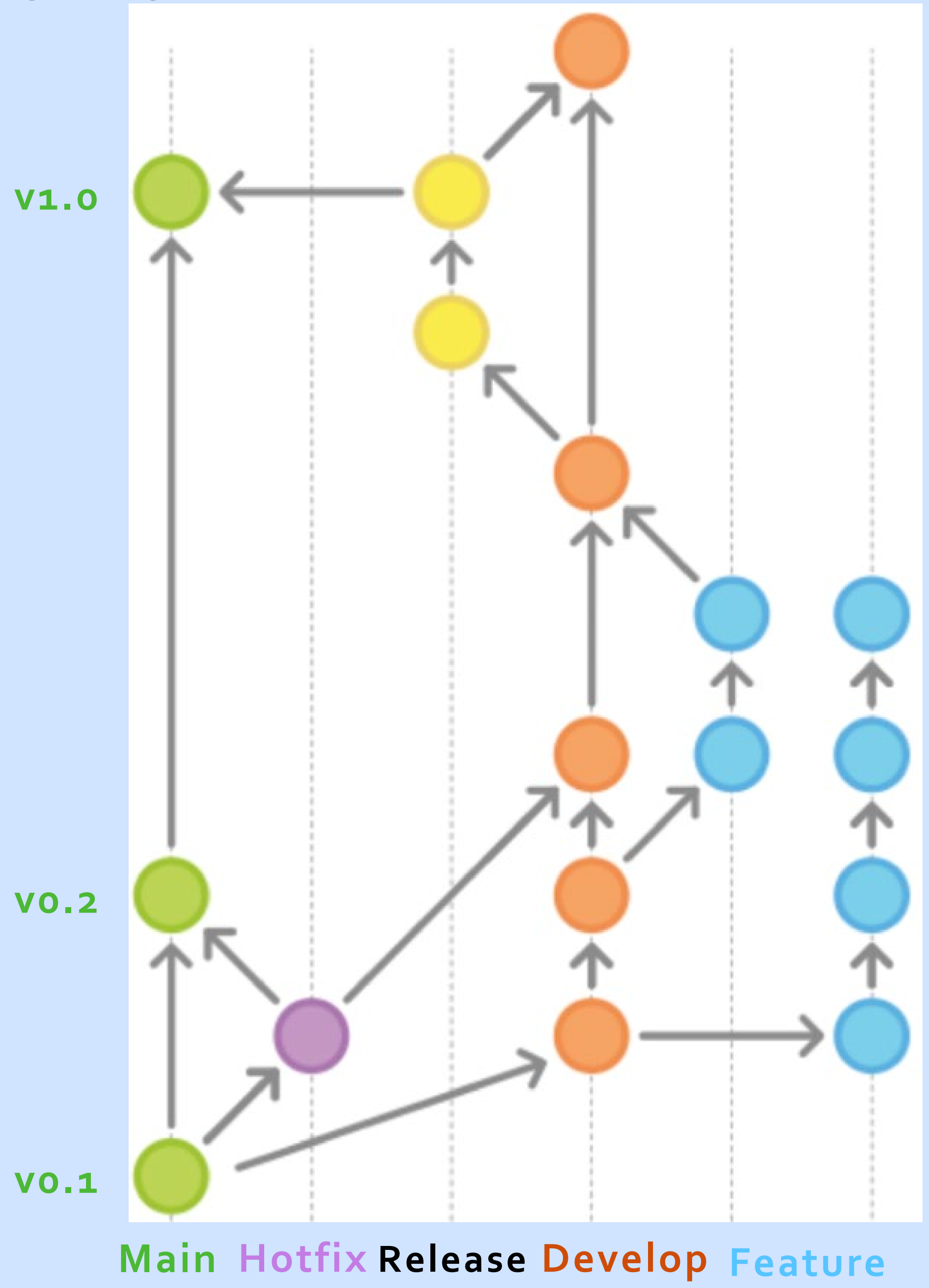
- Small frequent updates to a core trunk
  - Code review, feedback done via Pull Request
- Feature branches are short lived
- Use feature flags
- Most popular branching strategy for DevOps practices
- Notable companies using trunk-based development – GitHub (Microsoft), Google, Facebook, Amazon

Git Flow



- Specific roles to different branches
  - Develop branch created from Main
  - Release branch created from Develop
  - Feature branches created from Develop
  - When a feature is complete, that Feature branch is merged into Develop
  - Release branch created from Develop for final review
  - Release branch merged to both Main and Develop
  - If an issue is found, Hotfix created from Main
  - Once fixed, Hotfix merged to both Develop and Main

Git Flow

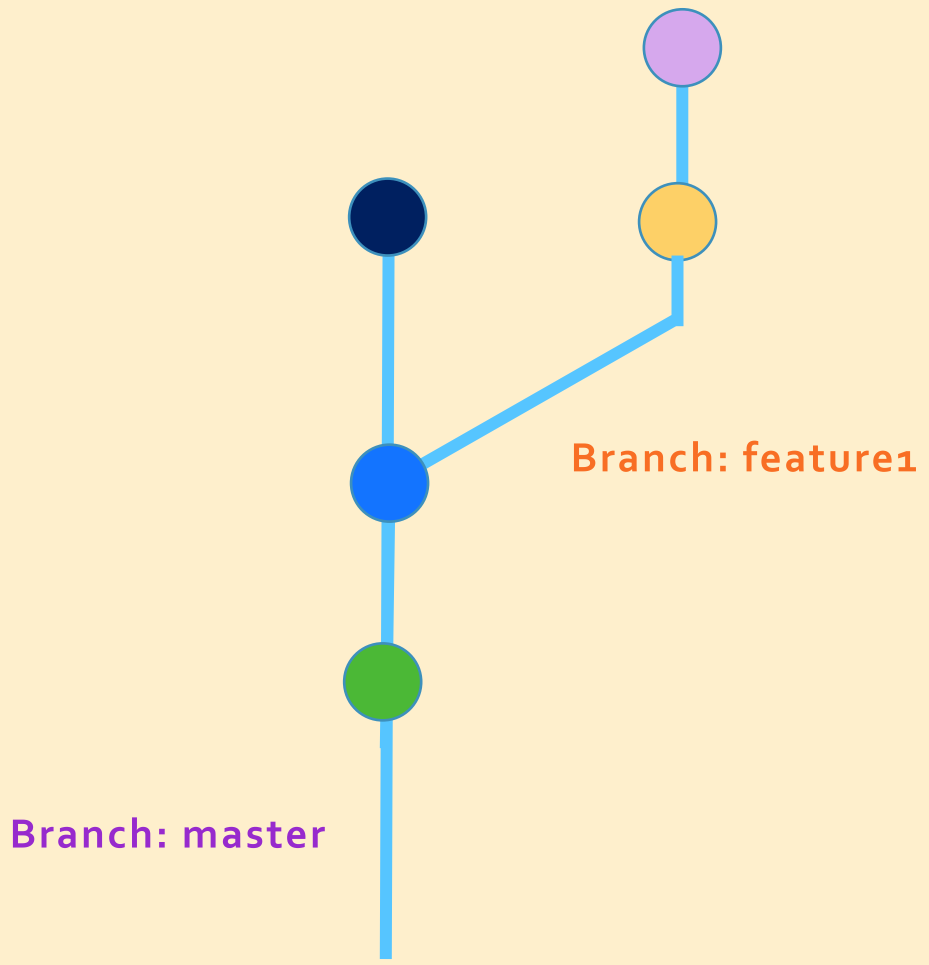


- Specific roles to different branches
  - Develop branch created from Main
  - Release branch created from Develop
  - Feature branches created from Develop
  - When a feature is complete, that Feature branch is merged into Develop
  - Release branch created from Develop for final review
  - Release branch merged to both Main and Develop
  - If an issue is found, Hotfix created from Main
  - Once fixed, Hotfix merged to both Develop and Main
- Works well in release-based workflow
- Works well with onboarding teams new to Git
  - Generally, requires a release manager

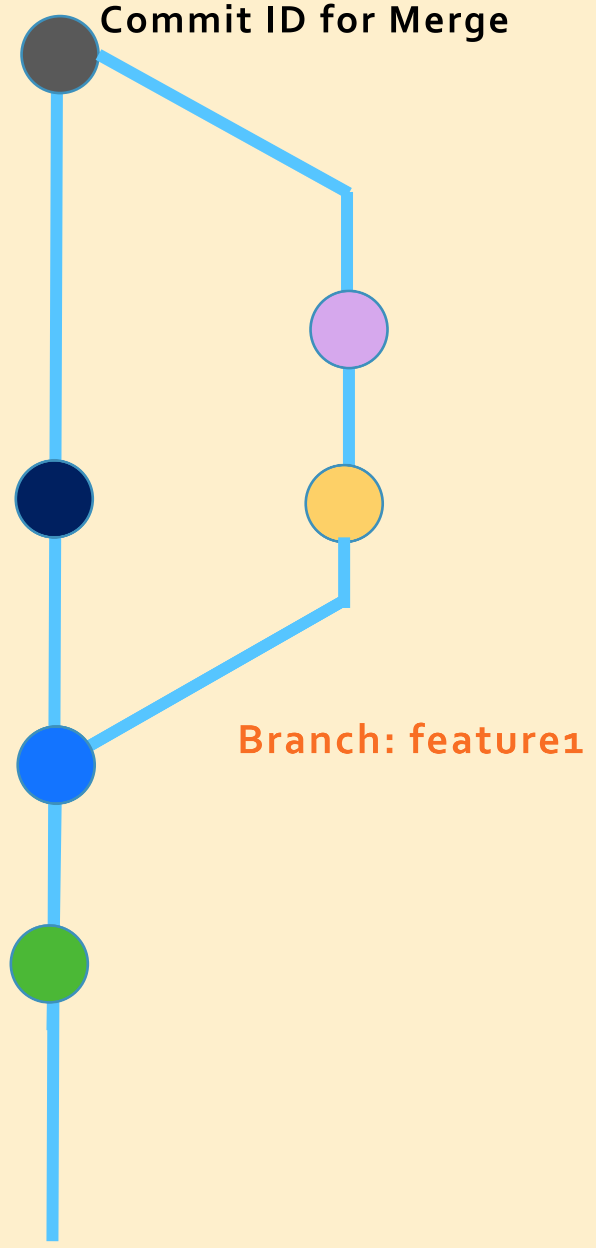
Rebase and Rebase vs Merge

Raj Saha
cloudwithraj.com
▶ Cloud With Raj

Regular Merge

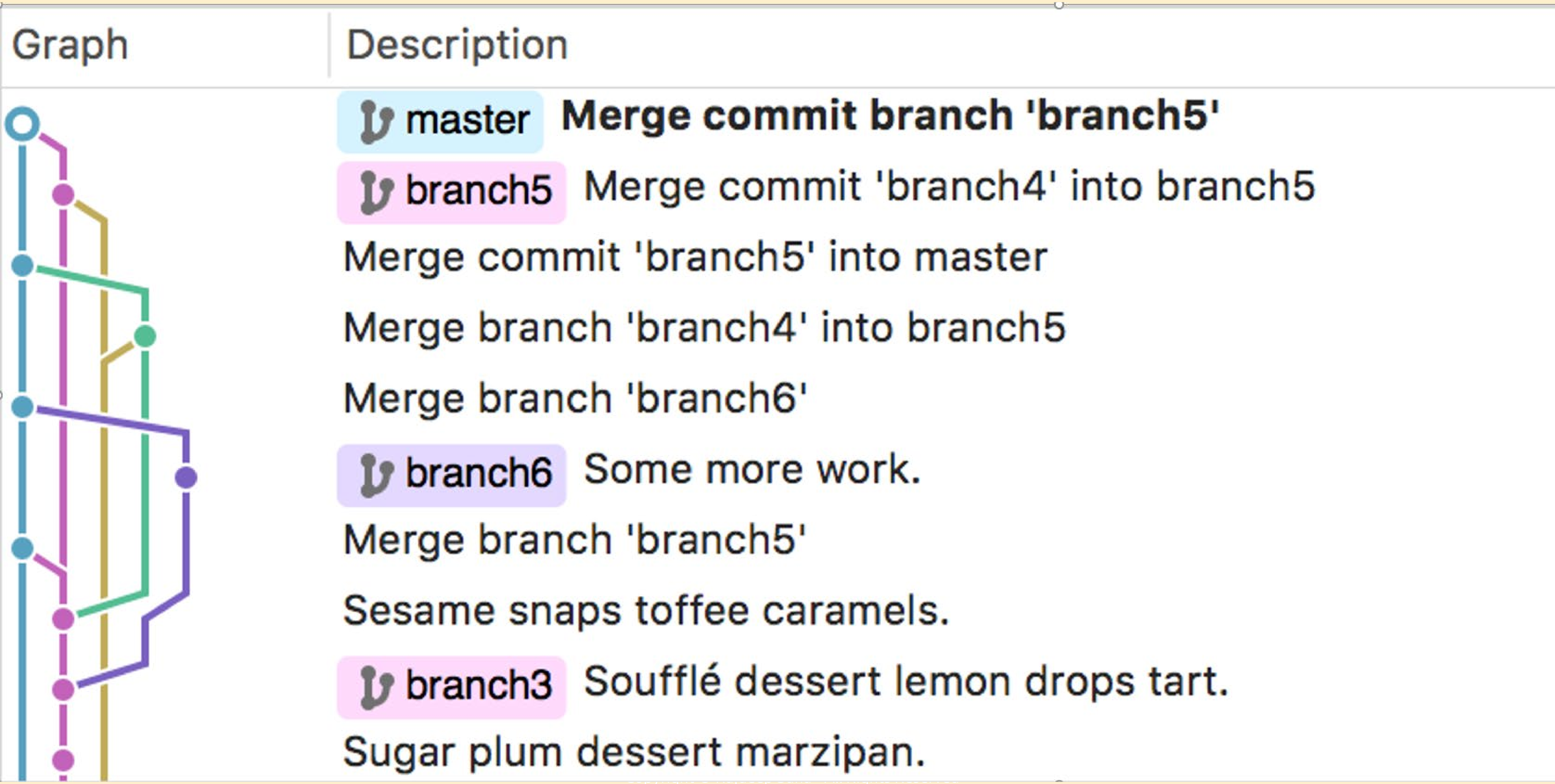


Regular Merge



git merge

Regular Merge Over Time



Graph | Description

master **Merge commit branch 'branch5'**

branch5 Merge commit 'branch4' into branch5

Merge commit 'branch5' into master

Merge branch 'branch4' into branch5

Merge branch 'branch6'

branch6 Some more work.

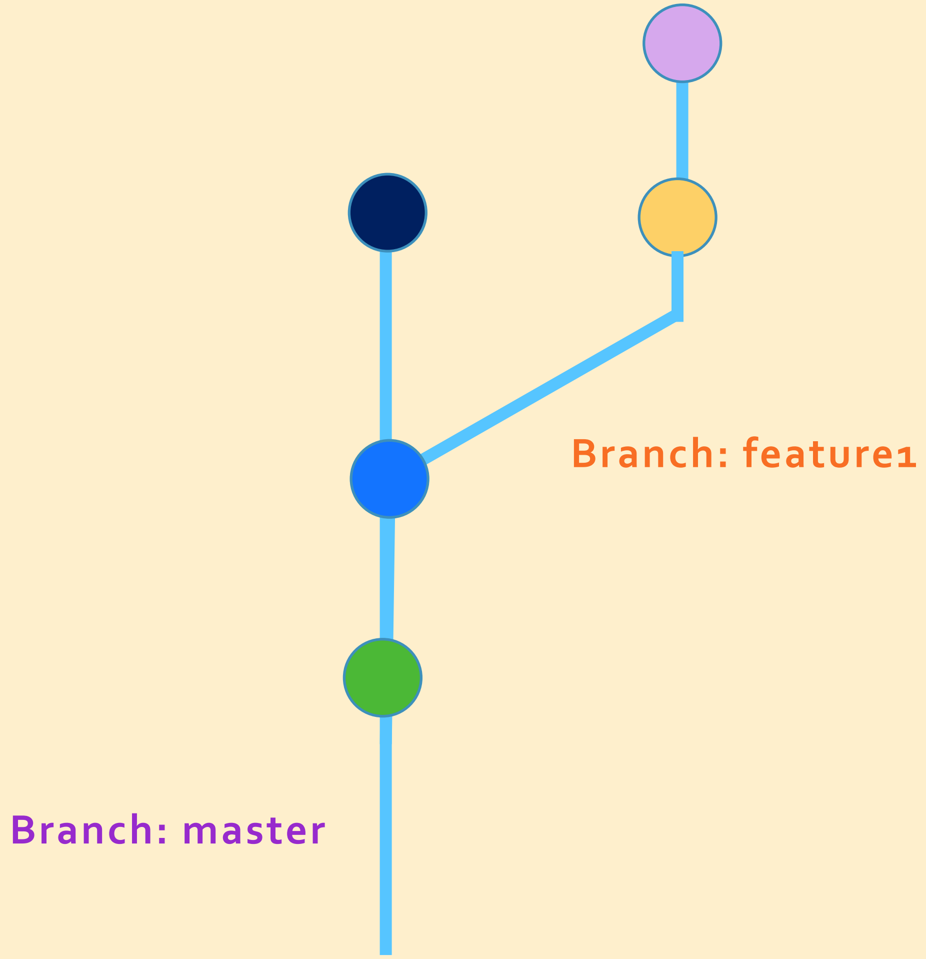
Merge branch 'branch5'

Sesame snaps toffee caramels.

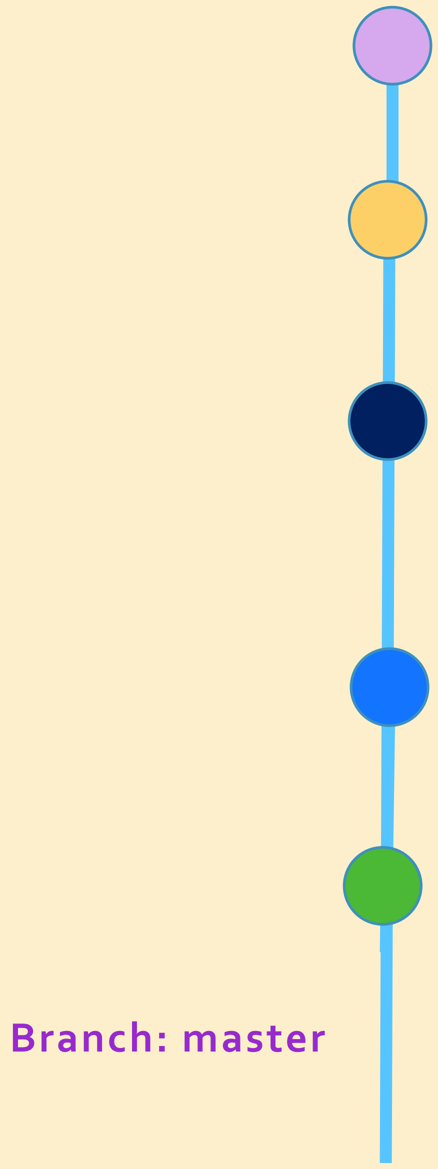
branch3 Soufflé dessert lemon drops tart.

Sugar plum dessert marzipan.

Rebase

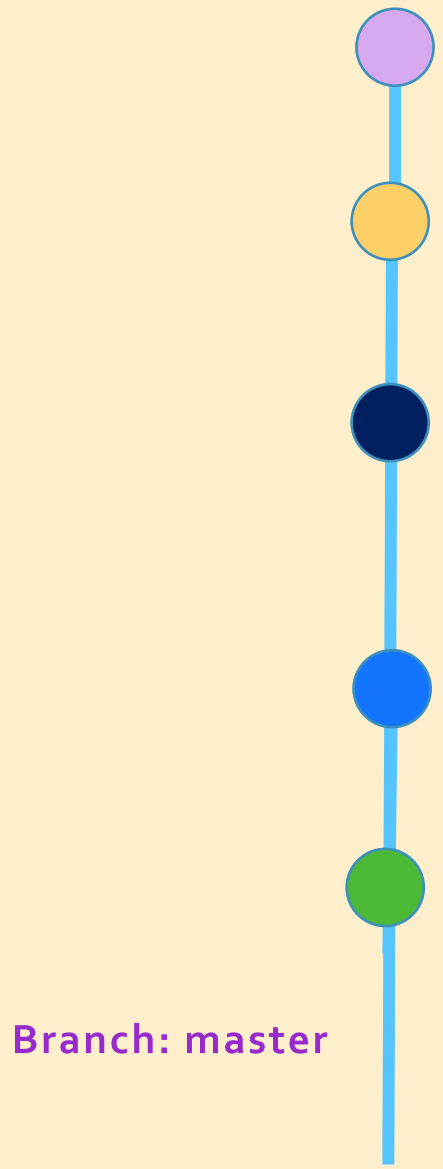


Rebase



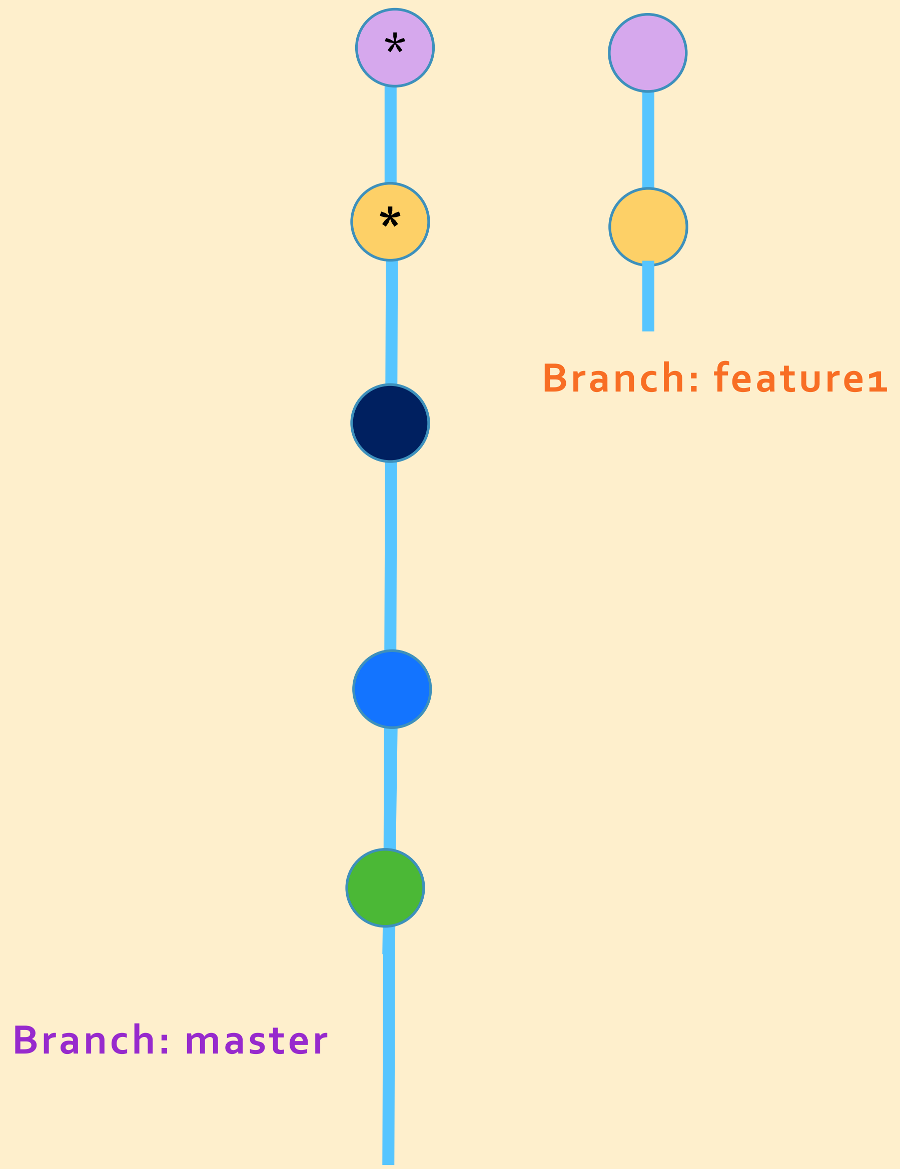
git rebase

Rebase



- Creates cleaner commit history
  - No additional merge commits
  - Easier to navigate with “git log”

Rebase

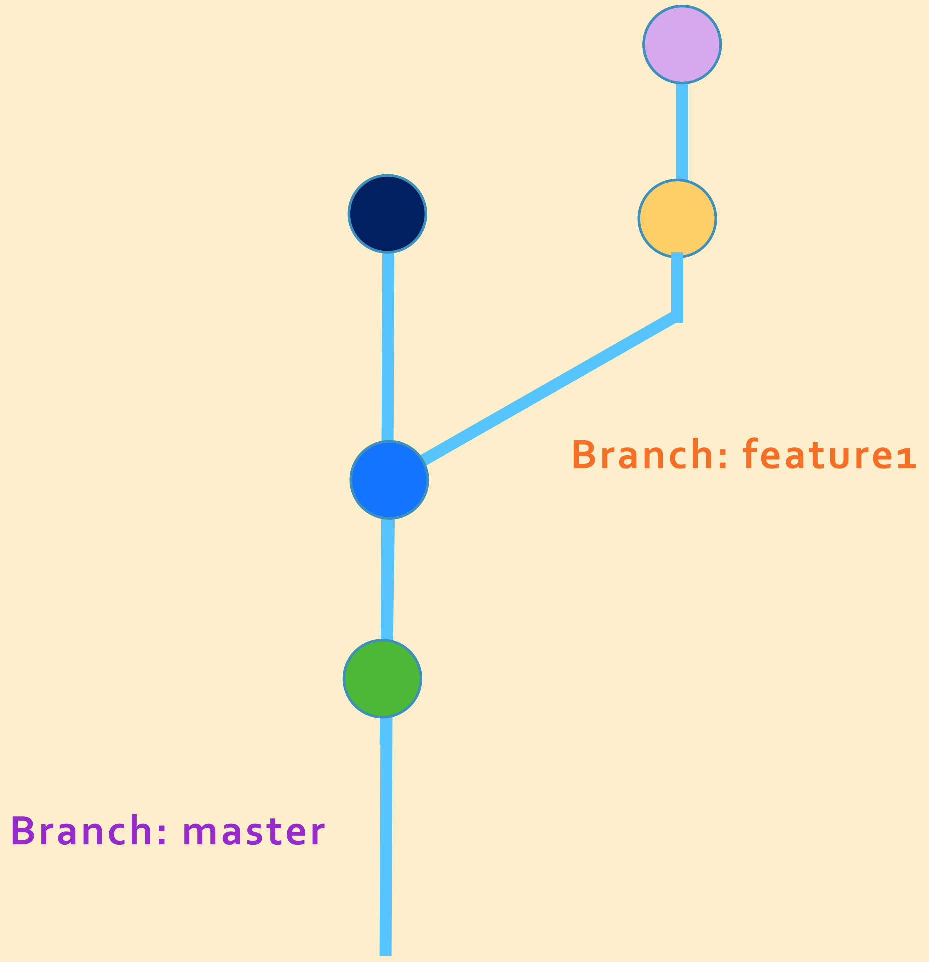


- Creates cleaner commit history
  - No additional merge commits
  - Easier to navigate with “git log”
- Re-writes commits
  - Rewrites history

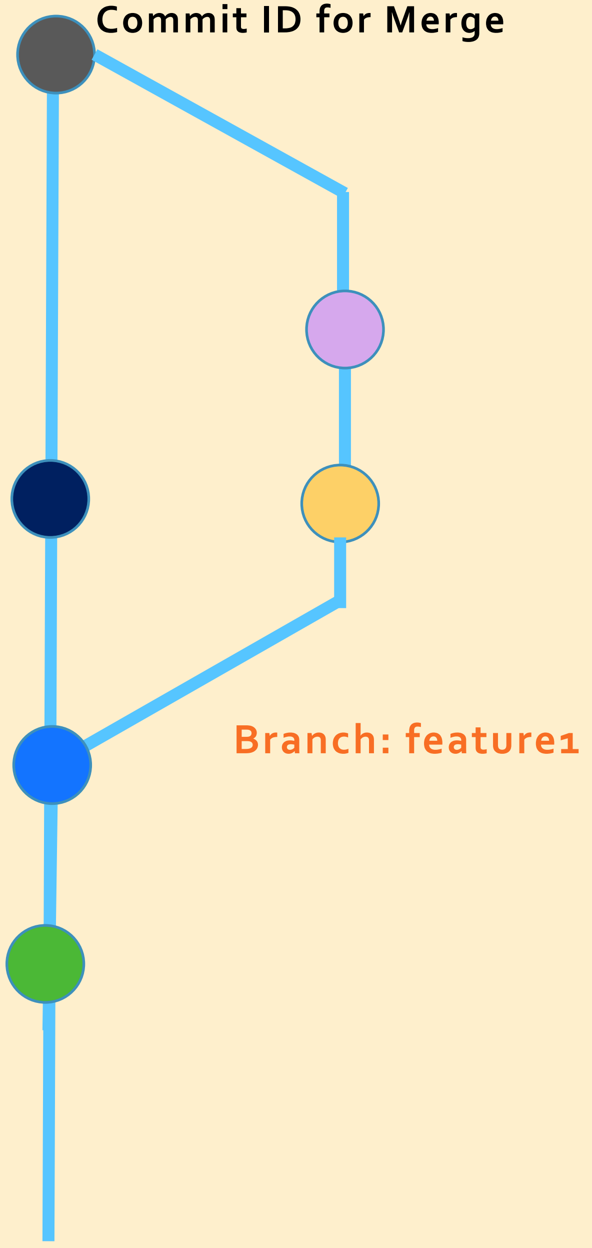
Git Cherry Pick

Raj Saha
cloudwithraj.com
Cloud With Raj

Regular Merge

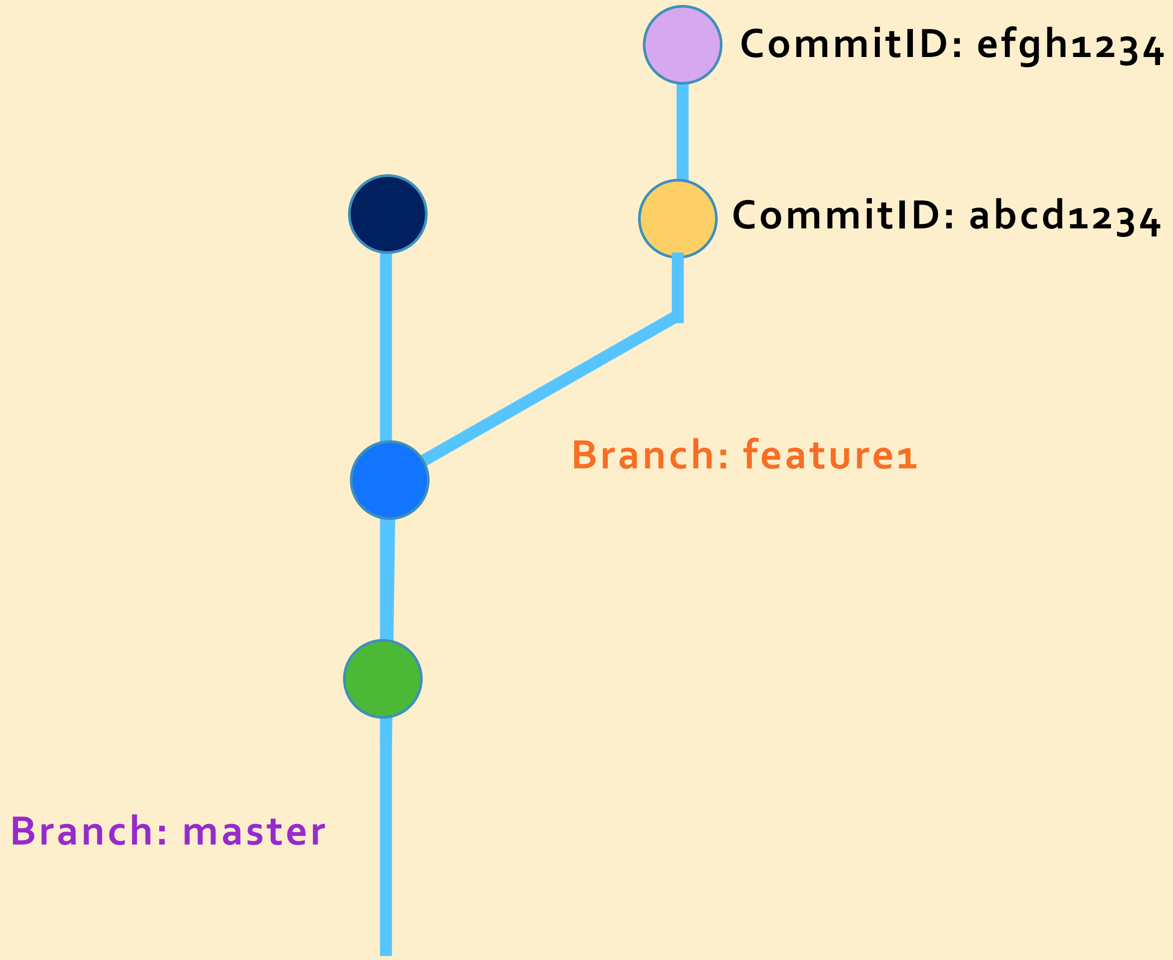


Regular Merge

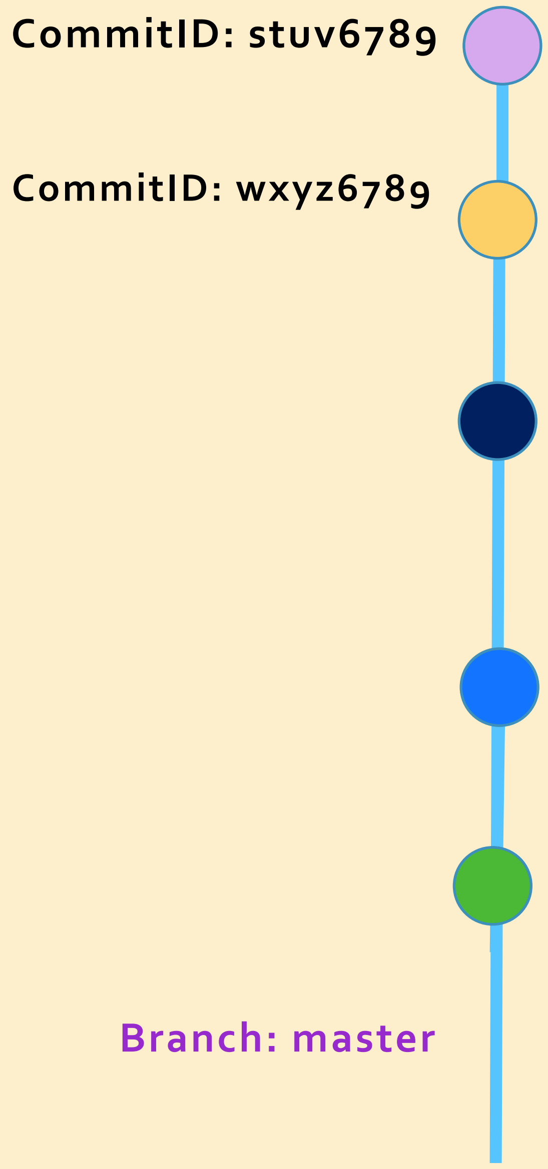


git merge

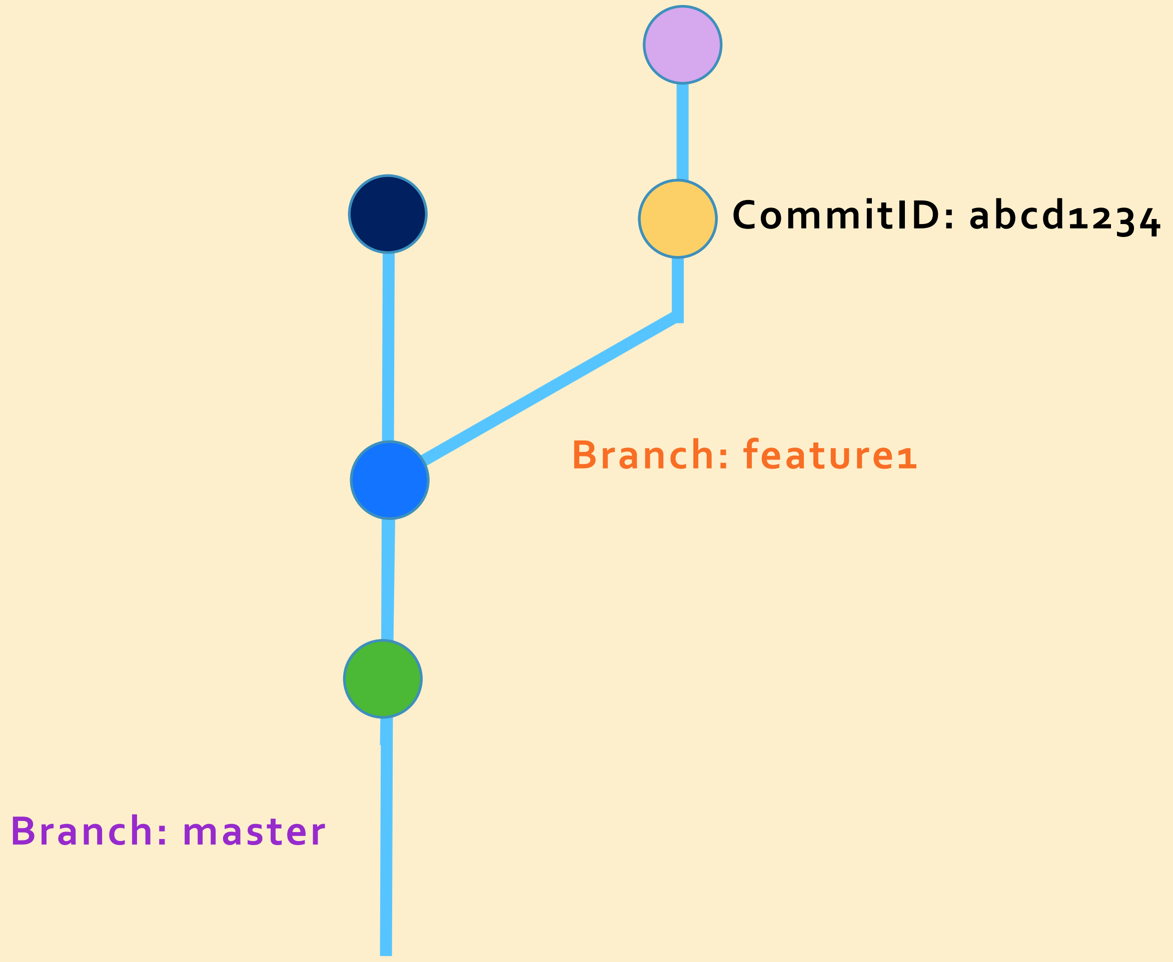
Rebase


CommitID: efgh1234
CommitID: abcd1234
Branch: feature1
Branch: master

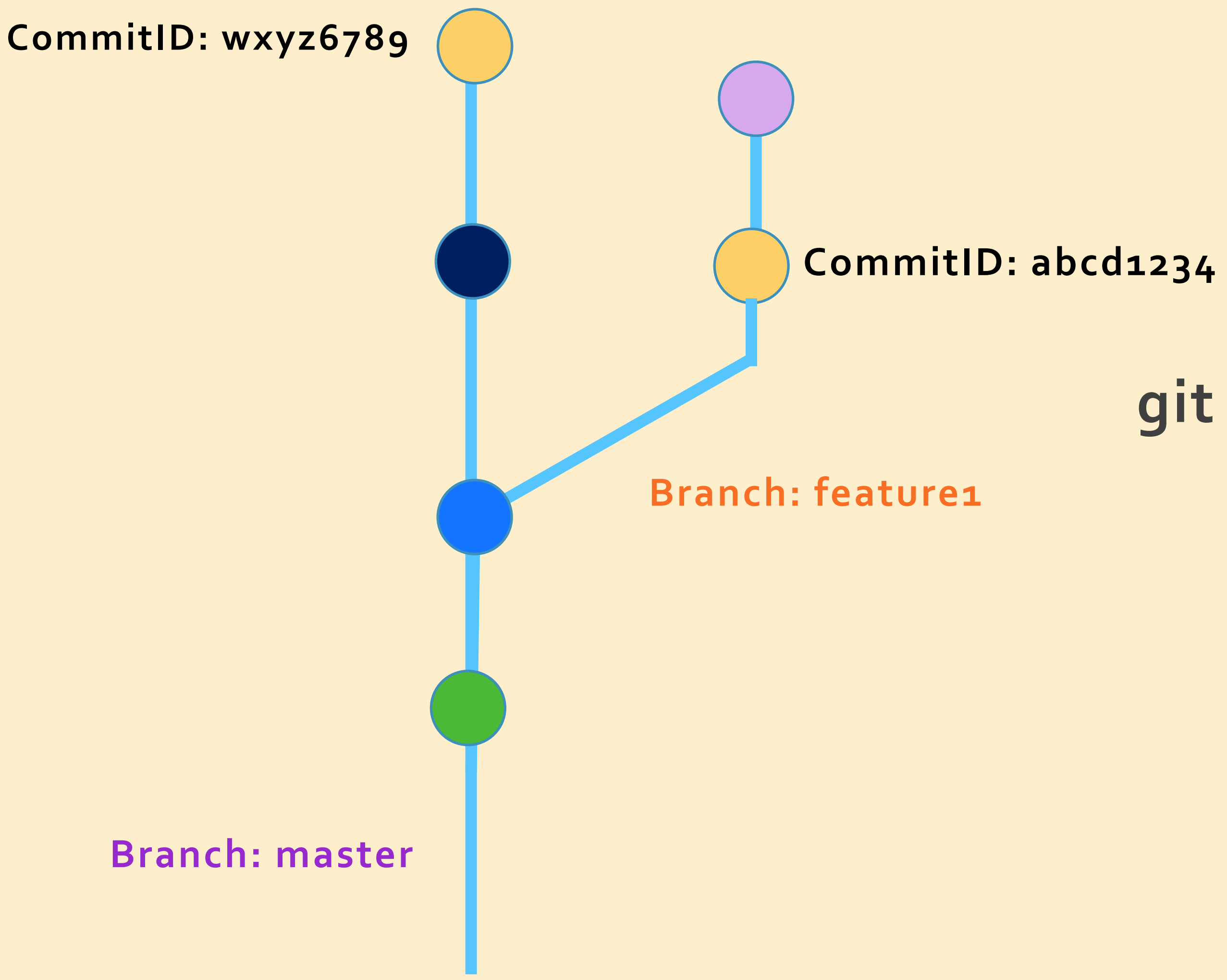
Rebase



Select Specific (Cherry Pick!) Commit(s)



Select Specific (Cherry Pick!) Commit(s)



```
git cherry-pick abcd1234
```

GitHub Tips to Get Selected by Recruiter

Raj Saha
cloudwithraj.com
Cloud With Raj

General Tips

- Don’t do analysis paralysis – done is better than perfect!
    - Explain use case, what you learned, how to run

- Mention any noteworthy PRs
    - Open-Source project contribution
    - Even a PR to a friend’s project with labels and issues helps

- Have a professional profile pic and name

- Don’t just fork clone, personalize it
    - Should be able to explain any artifact

5 GitHub Projects to Get Selected by Recruiter

Raj Saha
cloudwithraj.com
Cloud With Raj

5 GitHub Projects to Get Recruiter Attention

- Consuming a popular API
  - Bitcoin price, random news, random cat/dog facts
  - List of free APIs - https://github.com/public-apis/public-apis
  - Use periodic process, reports if no front end knowledge
- Frontend Website
  - Simple online store, pizza restaurant
  - https://github.com/aws-samples/aws-serverless-airline-booking
- Create an API
  - Swagger, Backend, IaC, Authn/Z
- DevOps Pipeline
  - Associated artifacts
- Your Resume Website

Git Best Practices

Raj Saha
cloudwithraj.com
Cloud With Raj

- Write meaningful commit messages

- Use branches (Do NOT git push to master)
  - Utilize Pull Requests

- Commit often, commit logical chunks

- Do NOT rewrite the main/master history
  - Rebase in the working branch before opening Pull Request

- Create and use meaningful labels

Git Interview Q/A - Basic

Raj Saha
cloudwithraj.com
Cloud With Raj

Git Basic Questions

1. What is a version control system? What is Git?
2. What are the advantages of using Git?
3. What is the difference between Git and GitHub?
4. Can you tell me the three storing areas of Git?
5. What is index?
6. How do you move files from staging to local repo to remote repo?
7. What is a Git repository?
8. How do you initialize a Git repository?
9. Name your 5 favorite Git commands
10. What are some of the Git hosting repositories?
11. What command do you use to copy the repo from GitHub to local?
12. What is Pull request?

Answers to Git Basic Questions

1. What is a version control system? What is Git?
2. What are the advantages of using Git?
3. What is the difference between Git and GitHub?
4. Can you tell me the three storing areas of Git?
5. What is index?
6. How do you move files from staging to local repo to remote repo?
7. What is a Git repository?
8. How do you initialize a Git repository?
9. Name your 5 favorite Git commands
   - git clone, git add, git commit, git push, git log
10. What are some of the Git hosting repositories?
    - GitHub, Gitlab, CodeCommit, BitBucket
11. What command do you use to copy the repo from GitHub to local?
12. What is a Pull request?

Git Interview Q/A - Moderate

Raj Saha
cloudwithraj.com
Cloud With Raj

Git Moderate Questions

1. How do you remove files from Git repo?
2. What is the difference Git Pull and Pull Request?
3. What is the difference between Git Pull and Git Fetch?
4. What is a Merge Conflict? How do you resolve it?
5. How can you list all the files changed in a particular commit?
6. What is the difference between fast forward merge and three way merge?
7. How can you tell if a branch has been merged or not?
8. I want to ignore certain files in my folder to be tracked by Git. How do I achieve this?
9. What are some of the Git best practices?
10. What is the difference between git remote add and git clone?
11. How can you list all the files changed with each commit?

Answers to Git Moderate Questions

1. How do you remove files from Git repo?
2. What is the difference Git Pull and Pull Request?
3. What is the difference between Git Pull and Git Fetch?
4. What is a Merge Conflict? How do you resolve it?
5. How can you list all the files changed in a particular commit?
6. What is the difference between fast forward merge and three way merge?
7. How can you tell if a branch has been merged or not?
8. I want to ignore certain files in my folder to be tracked by Git. How do I achieve this?
9. What are some of the Git best practices?
10. What is the difference between git remote add and git clone?
11. How can you list all the files changed with each commit?

Git Interview Q/A - Advanced

Raj Saha
cloudwithraj.com
Cloud With Raj

Git Advanced Questions

1. What is Cherry picking?
2. What is Squash Merge?
3. What is the difference between Cherry Picking and Merge?
4. What is the difference Rebase and Merge?
5. How do you restore a commit?
6. What is the difference between revert and reset?
7. What is the difference between fork and clone?
8. What is the difference between git stash and git add?
9. How do you keep your forked repo updated with upstream?
10. What are the purpose of GitHub issues?
11. What is a Webhook? How is it different than API?
12. How does DevOps tools get notified of repository changes?
13. How can you integrate GitHub with a Jenkins job?

Answers to Git Advanced Questions

1. What is Cherry picking?
2. What is Squash Merge?
3. What is the difference between Cherry Picking and Merge?
4. What is the difference Rebase and Merge?
5. How do you restore a commit?
6. What is the difference between revert and reset?
7. What is the difference between fork and clone?
8. What is the difference between git stash and git add?
9. How do you keep your forked repo updated with upstream?
10. What are the purpose of GitHub issues?
11. What is a Webhook? How is it different than API?
12. How does DevOps tools get notified of repository changes?
13. How can you integrate GitHub with a Jenkins job?

Raj Saha
cloudwithraj.com
Cloud With Raj
cloudwithraj
linkedin.com/in/rajdeep-sa-at-aws/

Instructor Bio:
Sr. Solutions Architect @ aws
Published Udemy/Pluralsight author
Public speaker
Author of multiple AWS official blogs
Previously - Distinguished Cloud Architect @Verizon

Opinions are my own