LARANA COMPANY

VERSION CONTROL

GIT & GITHUB

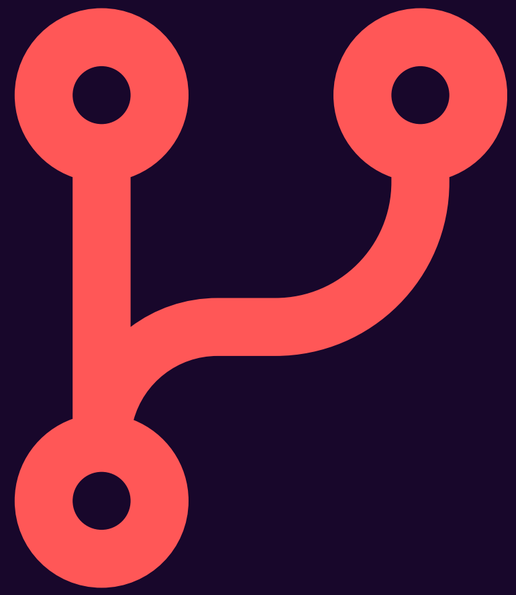
linkedin.com/rishabhio

JIO MLOPS 2024

Why Git Matters?

Version Control

Distributed System



Branching & Merging

Open Source

Git Operations

Basic Commands

Undoing Changes

Remote Repositories

Branching and Merging

History and Logs

Miscellaneous

Basic Commands

Initializes an empty git repo in the current directory

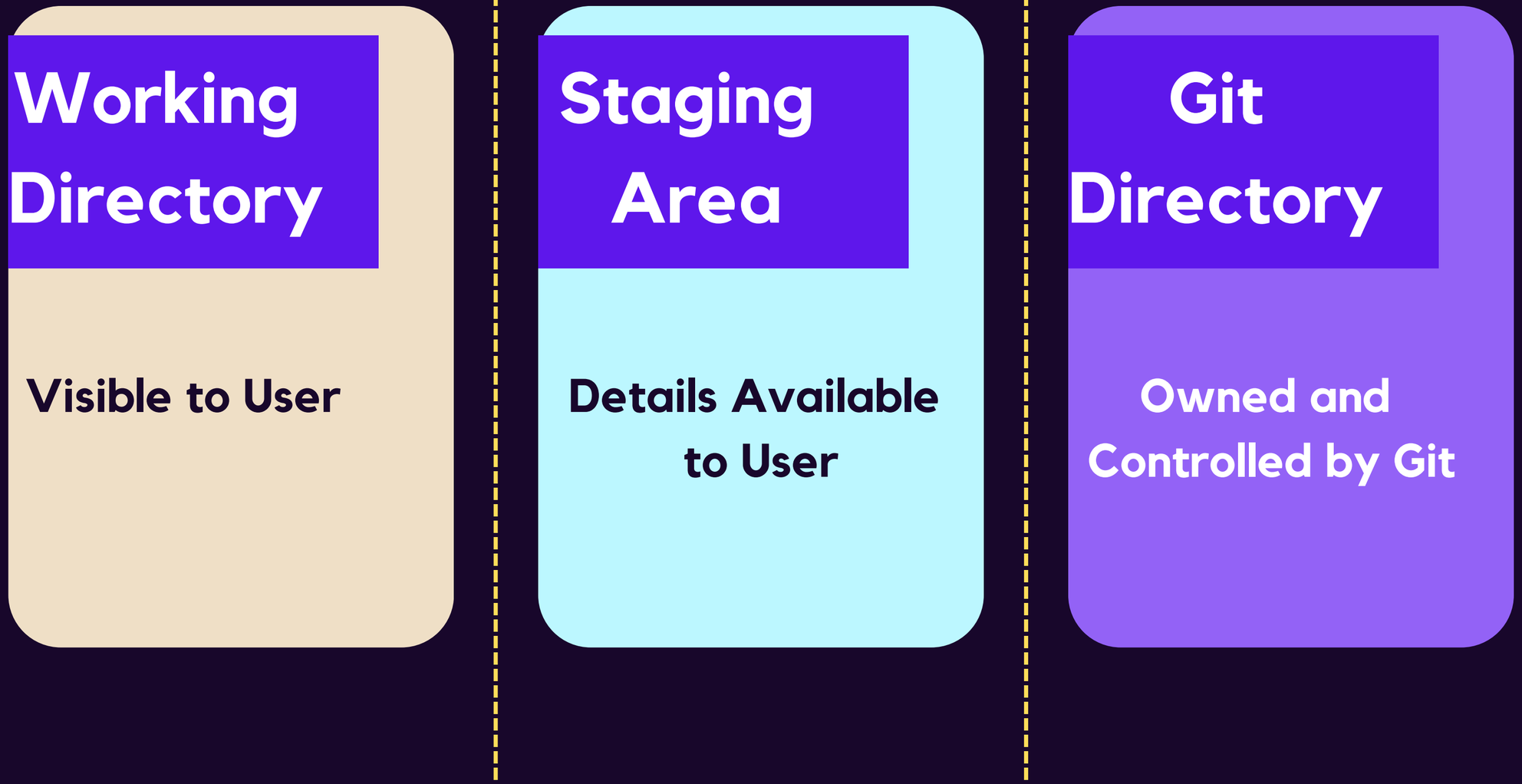
Git creates all files, datastructures etc it needs to track files.

git init

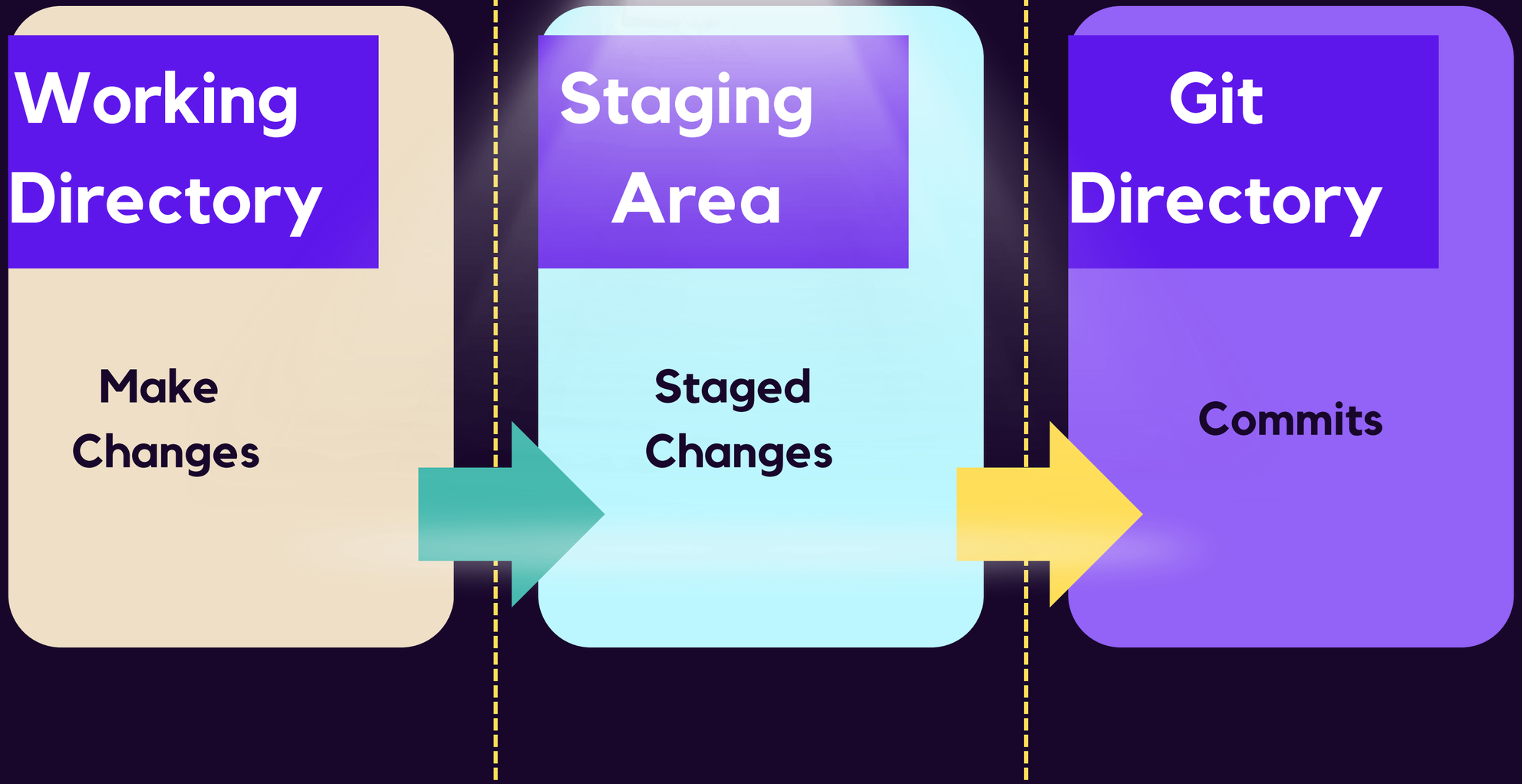
Git Clone

Copies a repository from a server to a local directory

git clone <url>

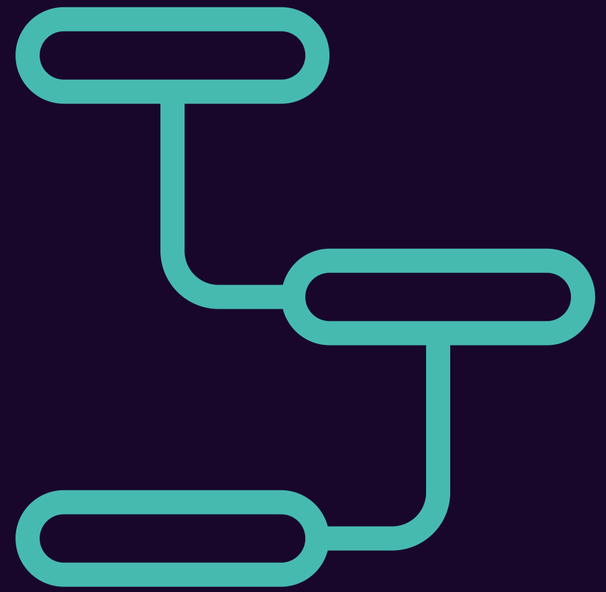


Different Spaces as seen from the Eyes of Git System

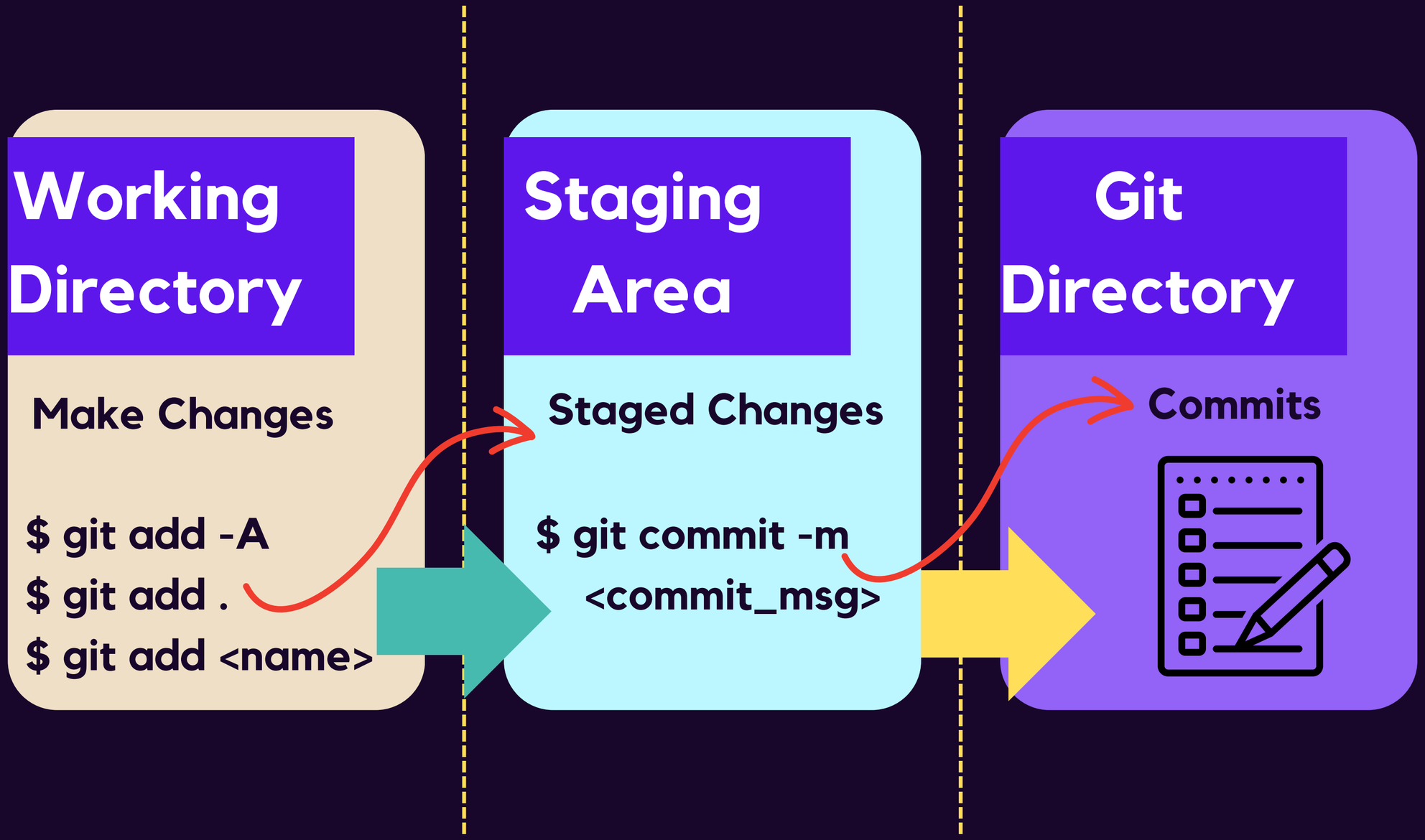


Typical Flow of Data into the Repository

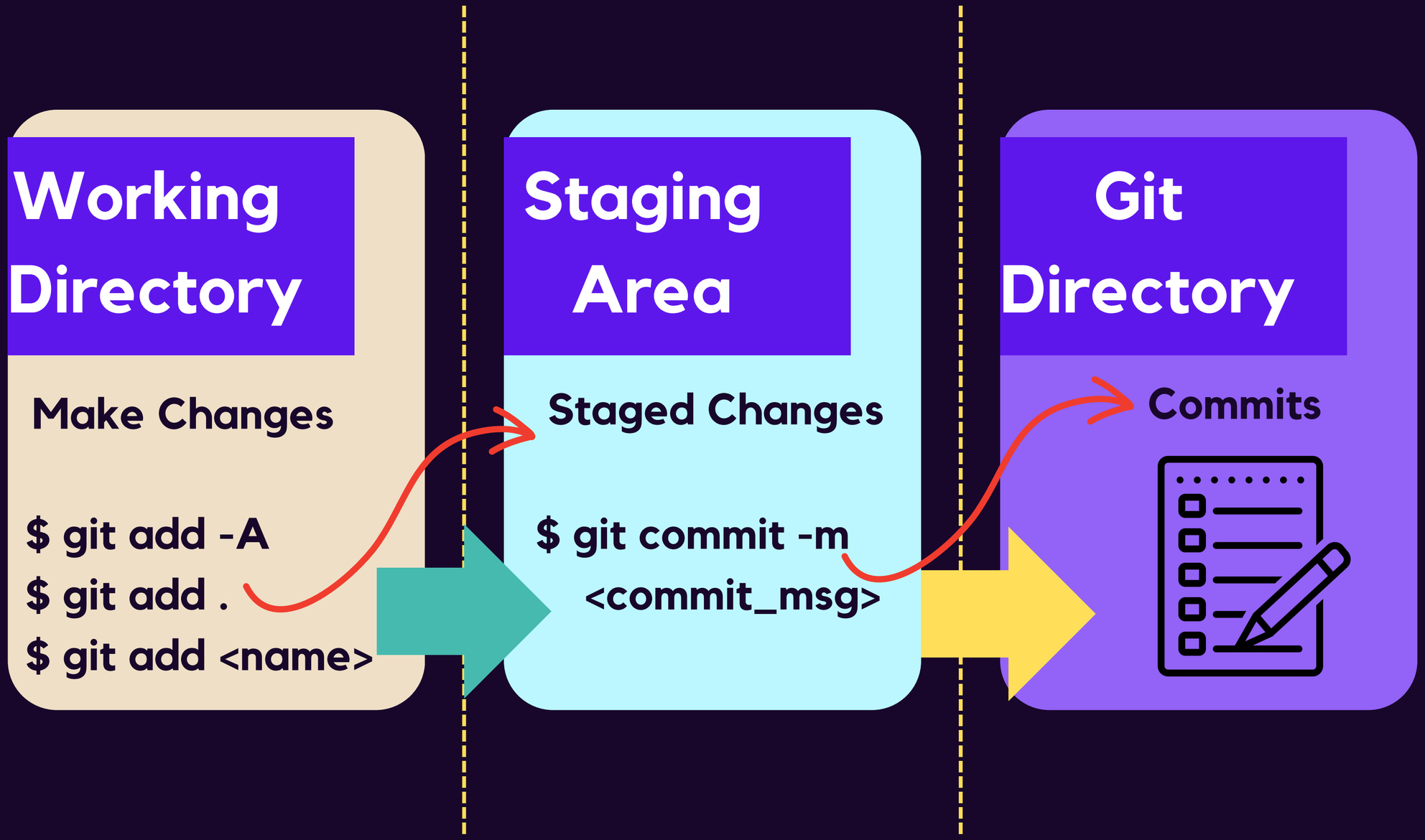
Git Only considers staged changes as eligible to be committed to repository.



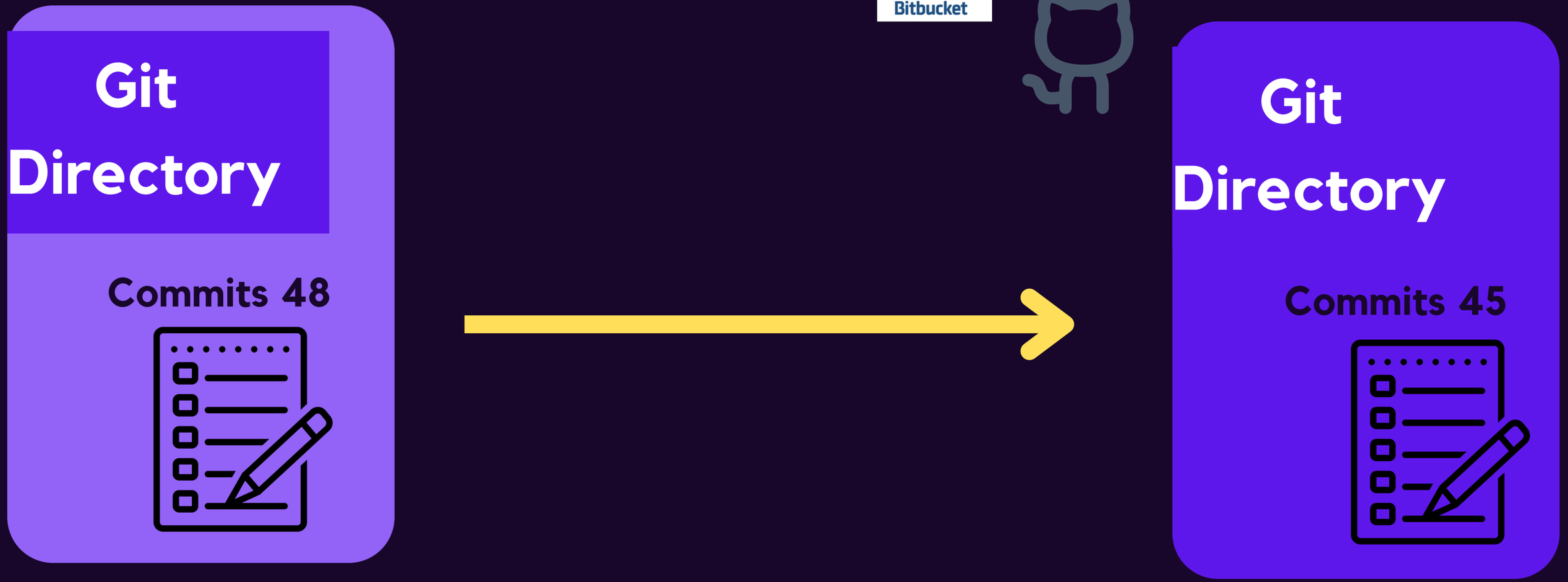
This prevents accidental commit of any edited which is in working directory.



Typical Flow of Data into the Repository



Check the Status of Different Files at Different Stages with git status



Local and Remote are supposed to be replica of each other.

Details about Remote

Get details about the Remote of the repository

```
git remote -v
```

Add a new Remote

Add a new repo.
e.g. if your project is already on local

git remote add [name] [url]

git remote add origin <url>

Push Changes

Push code to the branch of the repo.

git push [remote_name] [branch]

git push origin <branch_name>

Pull Changes

Pull code from the branch of the repo.

git pull [remote_name] dbranch]

git pull origin <branch_name>

Undoing Changes in Git

Unstage Changes

Unstage changes in a file

git reset [file]

git reset abc.txt

Revert a Commit

A new commit is introduced to undo. changes

git revert [commit]

git revert <commit_id>

Note on Revert.

Git creates a new commit that represents the inverse of the changes introduced by the specified commit. In other words, Git applies the "inverse patch" of the specified commit to the current branch, effectively undoing the changes made by that commit.

Reset a commit

```
git reset --hard [commit]
```

How

git reset --hard [commit]

Works ?

Step-1

You specify the commit you want to reset to by providing its identifier (commit hash) as an argument to the git reset --hard command

Step-2

Git moves the HEAD pointer (the reference to the current commit) to the specified commit

Step-3

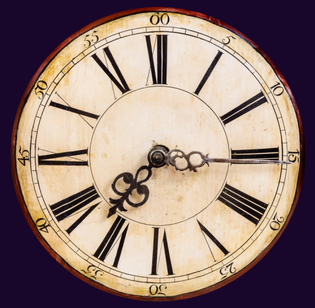
Git resets the staging area and working directory to match the state of the specified commit. This means that any changes in the staging area or working directory that are not part of the specified commit will be discarded.

Step-4

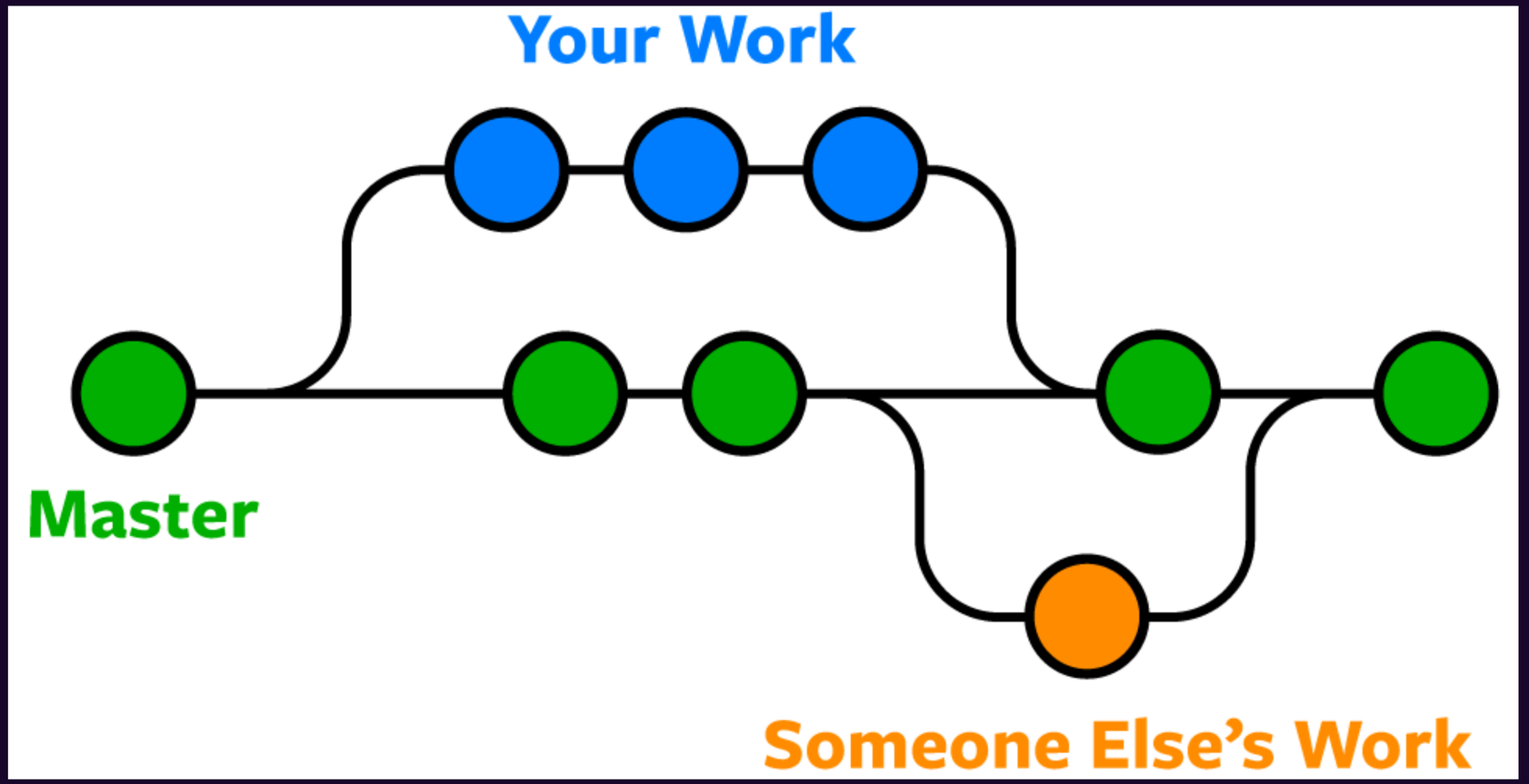
Any commits made after the specified commit, along with their associated changes, are effectively removed from the branch's history.

History & Logs

git log:	**Display commit history.**
git log --oneline:	**Display compact commit history.**
git log --graph --oneline --all:	**Display commit history graph.**



Branching & Merging


Your Work
Master
Someone Else's Work

List all Branches

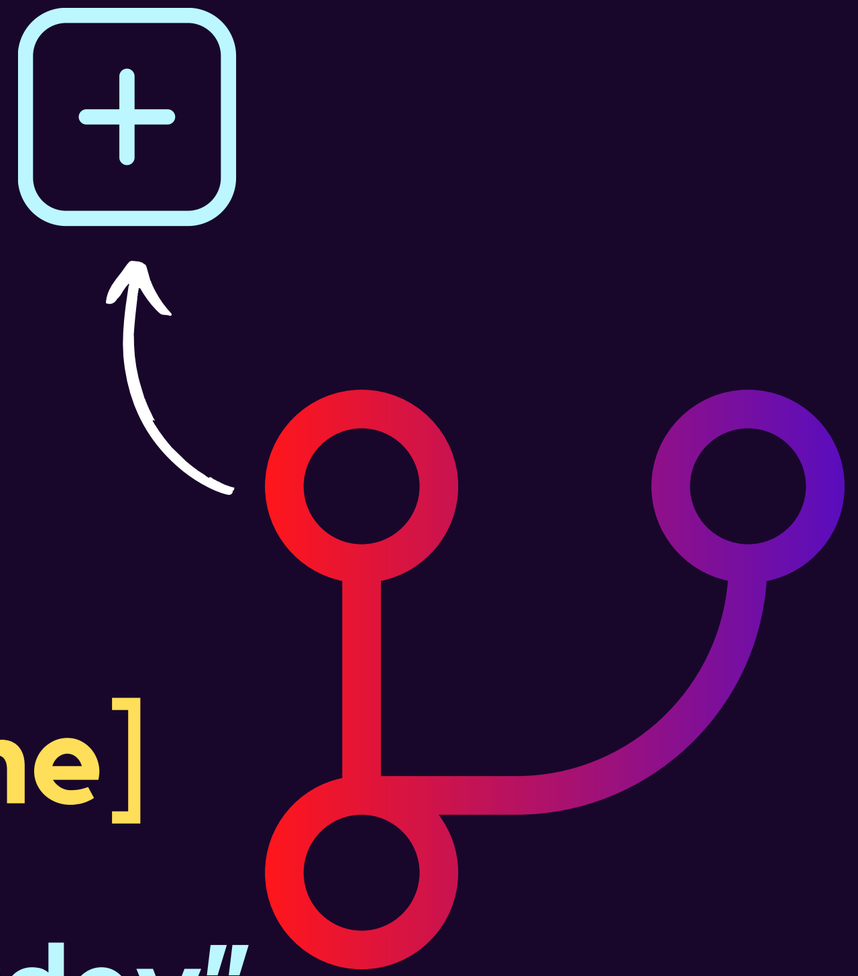
List all the available branches

git branch

Shows all the available branches

Create a Branch

Create a new Branch from the existing ones.



git branch [branch_name]

git branch “dev”

Git Checkout

Switch to a different branch.

git checkout [branch_name]

git checkout “main”

Git Merge

Merge changes from another branch into the current branch

```
git merge [branch_name]
```

```
git merge dev
```

Assignment

Assignment: Git Commands Mastery

Complete each task by executing the appropriate Git command(s) in your local repository.
Ensure that you understand the purpose and effect of each command before executing it.
Feel free to refer to the Git documentation or online resources for additional guidance.
Create commit messages against each of the commands discussed in the presentation.

Thank you!