AURORA ROBOTICS CORE WORKSHOP 1.0

GIT AND GITHUB

GENERAL CLASS RULES

- All mics should be muted
- I am always looking at the chat box so feel free drop questions anytime
- Feel free to use the “raise your hand” option anytime
- I recommend you have a note (could be digital), I tend to drop random knowledge casually

All slides, code and materials will be shared in the training repo. Feel free to use and share but do not modify

(I am very good with lawsuits)

WHAT IS GIT

Git is all about code control and tracking changes over time

- It is a version control system (VCS)
- It helps you track changes in code over time
- Think of it as a “time machine” for your project
- You can go back to an older version anytime
- Supports collaboration, multiple developers can work on the same project without overwriting each other’s work
- Git keeps a history (commits) of every change, who made it, and why

WHAT IS GITHUB

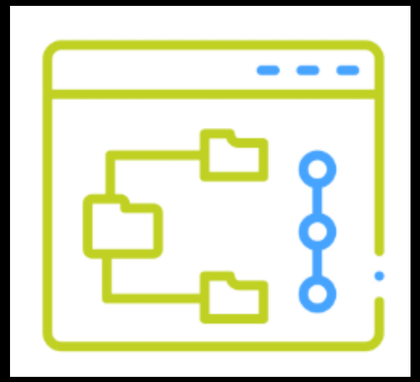
- GitHub is a cloud platform built on top of Git
- It hosts your repositories online so you can access them from anywhere
- Makes it easy to collaborate with others on projects
- Provides tools for:
  - Pull Requests (suggesting & reviewing changes)
  - Issues (tracking bugs & tasks)
  - Actions (automation like CI/CD)
- Acts as a social network for developers, you can share, fork, and star projects

GIT VS. GITHUB

- Git
  - A tool (version control system)
  - Runs on your local machine
  - Tracks changes in files (commits, branches, history)
  - Works offline
- GitHub
  - A platform (online service)
  - Hosts repositories in the cloud
  - Provides extra features (pull requests, issues, CI/CD, project boards), enables collaboration & sharing
  - Works online

Analogy:
Git = your personal diary where you write daily updates.
GitHub = Instagram for code, where you post your diary online so others can read, comment, and collaborate.

WHAT IS VERSION CONTROL



- A system that records changes to files over time
- Lets you go back to previous versions if needed
- Keeps a history of who changed what, when, and why
- Useful for tracking bugs, experimenting, and managing large projects

OTHER 'GITS' 😅

- GitLab: Similar to GitHub, but also includes DevOps tools
- Bitbucket: Git hosting platform by Atlassian
- SourceForge: One of the older platforms for hosting code with Git/SVN support
- Azure Repos: Git repositories offered inside Microsoft's Azure DevOps
- Gitea / Gogs: Lightweight, self-hosted Git services for private repos
- AWS CodeCommit: Amazon's managed Git service
- GitKraken / Tower: Git desktop clients with visual interfaces for managing repos

CREATE A GITHUB ACCOUNT

- Open github.com in your browser
- Click “Sign up” (top-right or wherever the button appears for you)
- Enter your email address, password and username

- Choose if you want to receive updates
- Verify and solve the quick CAPTCHA puzzle (my motivation to build a robot to pass these thingssss😭)

- Confirm your email by clicking the link GitHub sends
- Choose free or paid plan (`"Paid" if person.is_related_to("Otedola") else "Free"`)

- Select your interests (optional)
- Ready to Use

- You now have a GitHub account
- Next: Create your first repository (intro-to-aurora or whatever)

INSTALL GIT

- On Ubuntu 👍
  - sudo apt update
  - sudo apt install git -y
  - git --version

- On Windows 😭
  - Go to git-scm.com and download the Git for Windows
  - Run the installer, keep defaults unless specific needs
  - Verify installation in Command Prompt or Git Bash:
  - git --version

INSTALL GIT

- On Mac 😖
  - Option A – Using Homebrew (recommended):
    - `brew install git`
    - `git --version`
  - Option B – Using Xcode Command Line Tools:
    - `xcode-select --install`

ESSENTIAL GIT TERMINOLOGIES

- Concepts
  - Working directory: your files on disk (untracked/modified).
  - Staging area (index): the “shopping cart” of changes selected for the next commit.
  - Local repository: your .git history (commits, branches) on your machine.
  - Remote repository – the copy hosted on GitHub

- Data flow:

Working Dir → git add → Staging → git commit → Local Repo → git push → Remote Repo

CONFIGURATION OF GIT

- All OS
  - `git config --global user.name "Your Github username"`
  - `git config --global user.email "yourGithub@email.com"`

PUSH FROM TERMINAL!

- All OS:
  - `ssh-keygen -t ed25519 -C "you@example.com"`
- Start agent & add key (Linux/macOS/no WSL yet!):
  - `eval "$(ssh-agent -s)"`
  - `ssh-add ~/.ssh/id_ed25519`
- On Windows (PowerShell, Git Bash):
  - `eval $(ssh-agent -s)`
  - `ssh-add ~/.ssh/id_ed25519`

PUSH FROM TERMINAL!

Copy the public key and add to GitHub
Go to Github → Settings → SSH and GPG keys → New SSH key

Terminal:

```
cat ~/.ssh/id_ed25519.pub
```

Test the connection:

```
ssh -T git@github.com
```

Expected outcome;
“Hi <username>! You've successfully authenticated…"

PUSH FROM TERMINAL!

- Initialize & first commit:
  - cd your-project
  - git init
  - git add .
  - git commit -m "Initial commit"
- Add remote (pick SSH or HTTPS):
  - git remote add origin git@github.com:<user>/<repo>.git #SSH
  - git remote add origin https://github.com/<user>/<repo>.git #HTTPS
- Push the main branch:
  - git push -u origin main
- Daily workflow
  - git add <files> # or: git add .
  - git commit -m "Message"
  - git push # pushes current branch

VS CODE

DO NOT ATTEMPT TO SET UP VS CODE IN WSL

USE VS CODE ON WINDOWS AND WE WILL PORT IT INTO WSL

- Windows
  - Go to code.visualstudio.com
  - Download the Windows Installer (.exe)
  - Run the installer
    - Select “Add to PATH” (recommended)
    - Finish setup and launch VS Code
    - Verify installation: open Command Prompt and run
    - code --version

VS CODE

- macOS
  - Option A — Installer
    - Go to code.visualstudio.com
    - Download the macOS .zip
    - Extract and drag Visual Studio Code.app to Applications
  - Option B — Homebrew (recommended)
    - brew install --cask visual-studio-code
    - Verify:
      - code --version

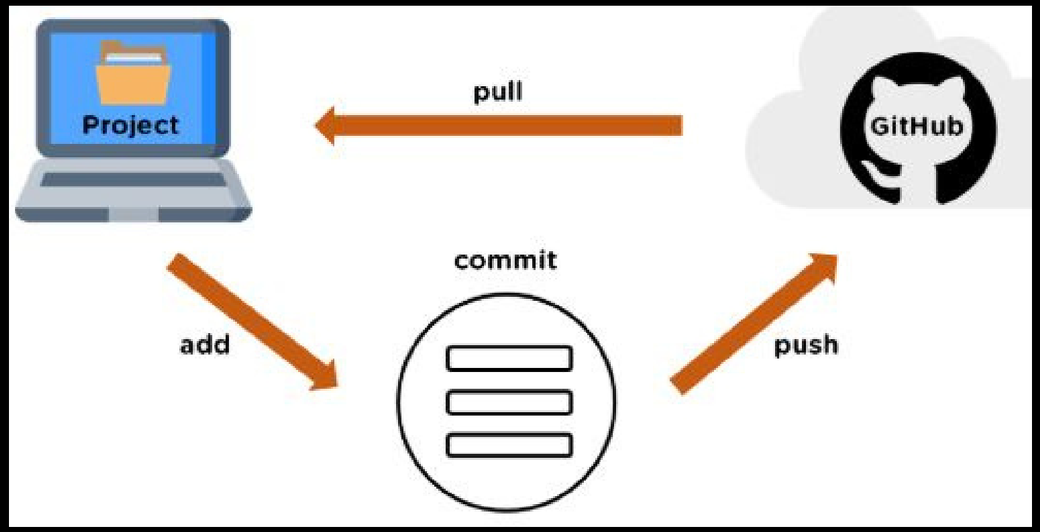
GIT WITH VS CODE

I'll show how.

ESSENTIAL GIT CONCEPTS

1. Project (Working Directory → Staging Area)
Concept: “Add” means put these changes in the basket for the next commit.
You edit files in your project (on your laptop).To tell Git which changes you want to save:

```
git add filename        # stage a single file
git add .               # stage all modified files
```



2. Staging Area → Local Repository
Concept: A commit is a permanent checkpoint in your local Git history.

```
git commit -m "Meaningful message about the change"
```

3. Local Repository → Remote Repository (GitHub)
Concept: Push uploads your commits from your local repository to the remote repository (GitHub).
To share your commits with GitHub:

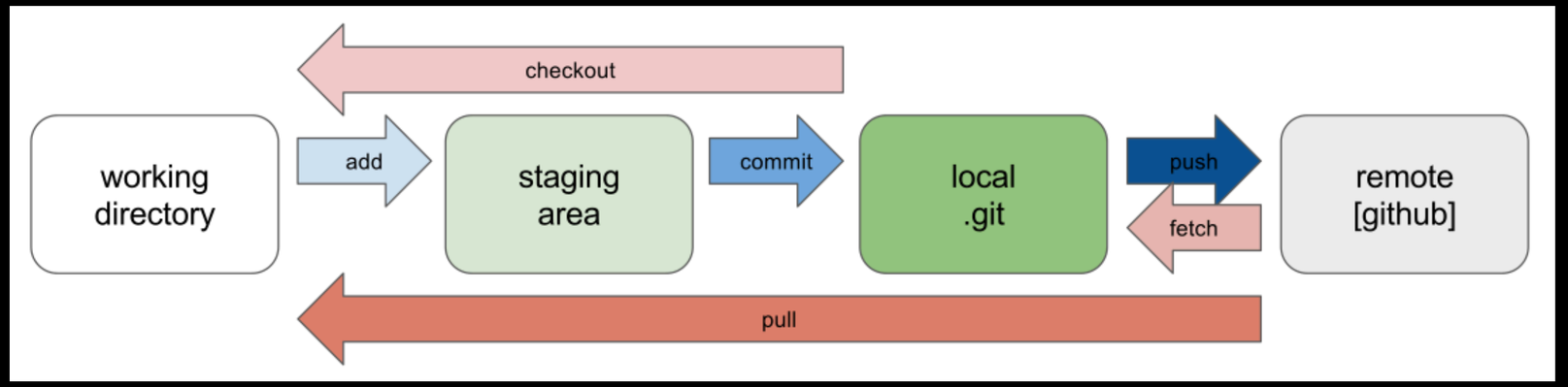
```
git push origin main
```

4. Remote Repository (GitHub) → Local Repository
Concept: Pull downloads commits from GitHub and merges them into your local branch, keeping you up to date.
If teammates have pushed changes to GitHub, you bring them into your local repo with:

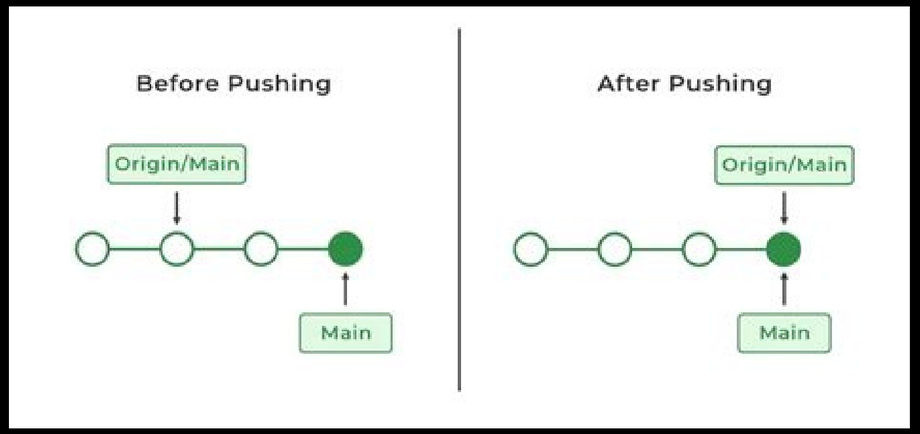
```
git pull origin main
```

- Putting it all together (full flow)
  - Edit files in your project, then stage changes → `git add .`
  - Save snapshot → `git commit -m "message"`
  - Upload → `git push origin main`
  - Download updates → `git pull origin main`

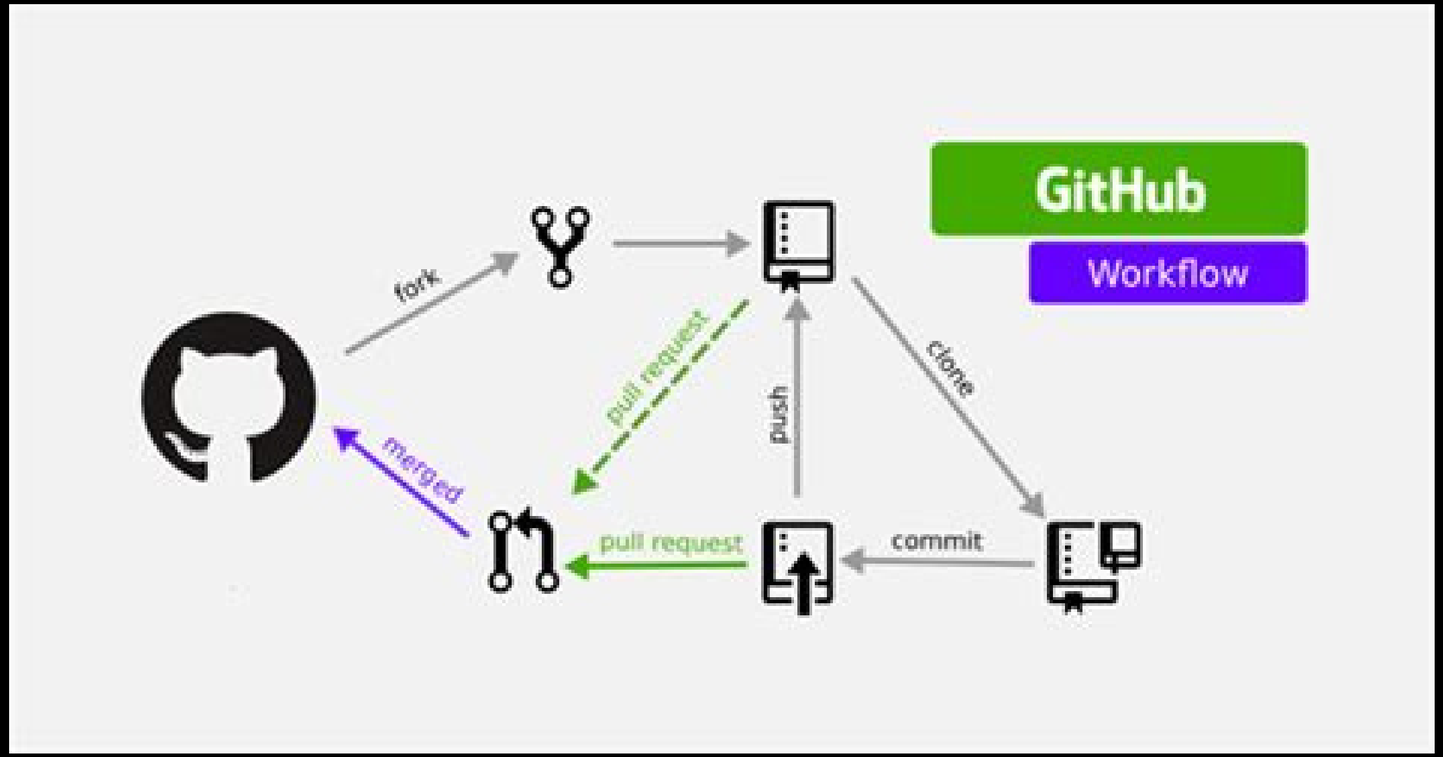
ESSENTIAL GIT CONCEPTS



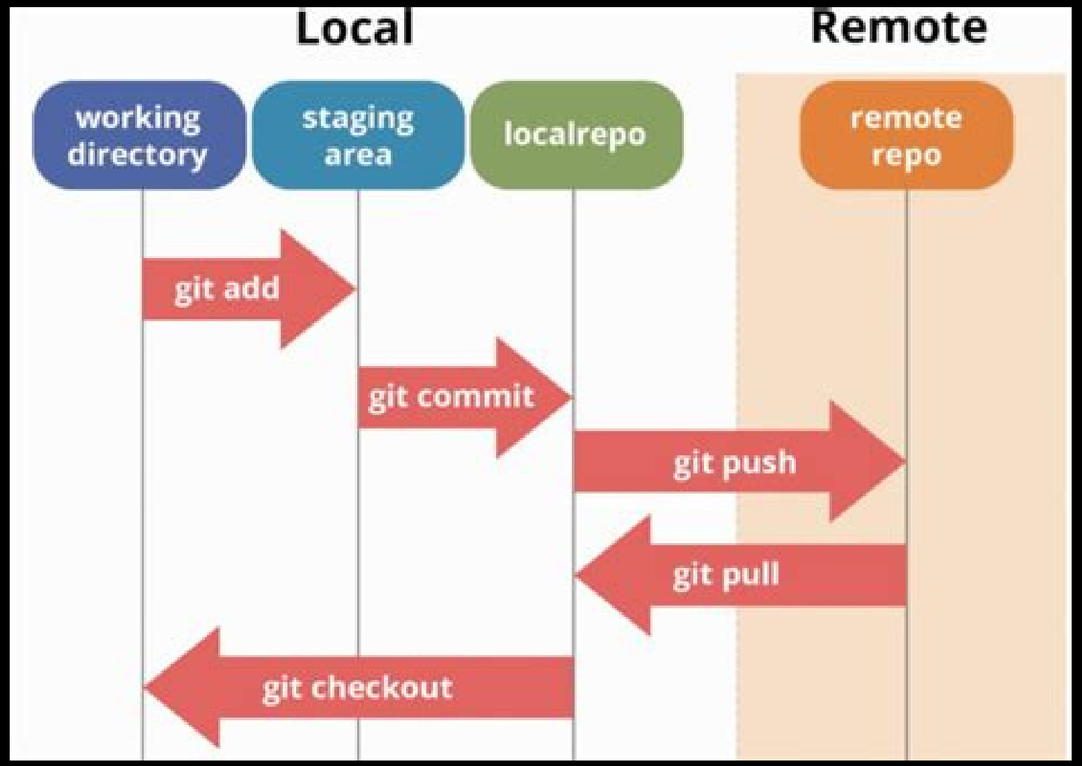
ESSENTIAL GIT COMMANDS



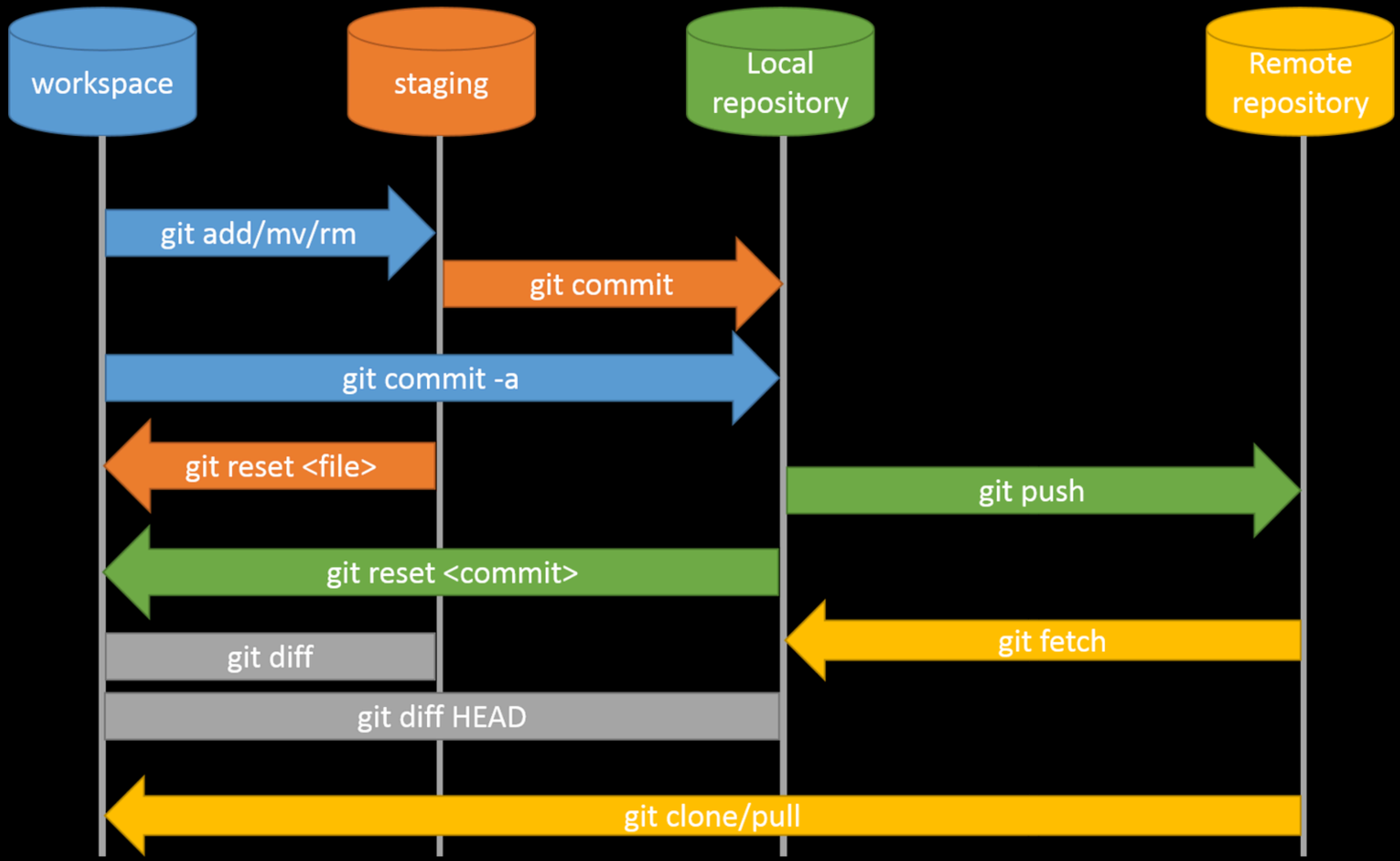
ESSENTIAL GIT COMMANDS



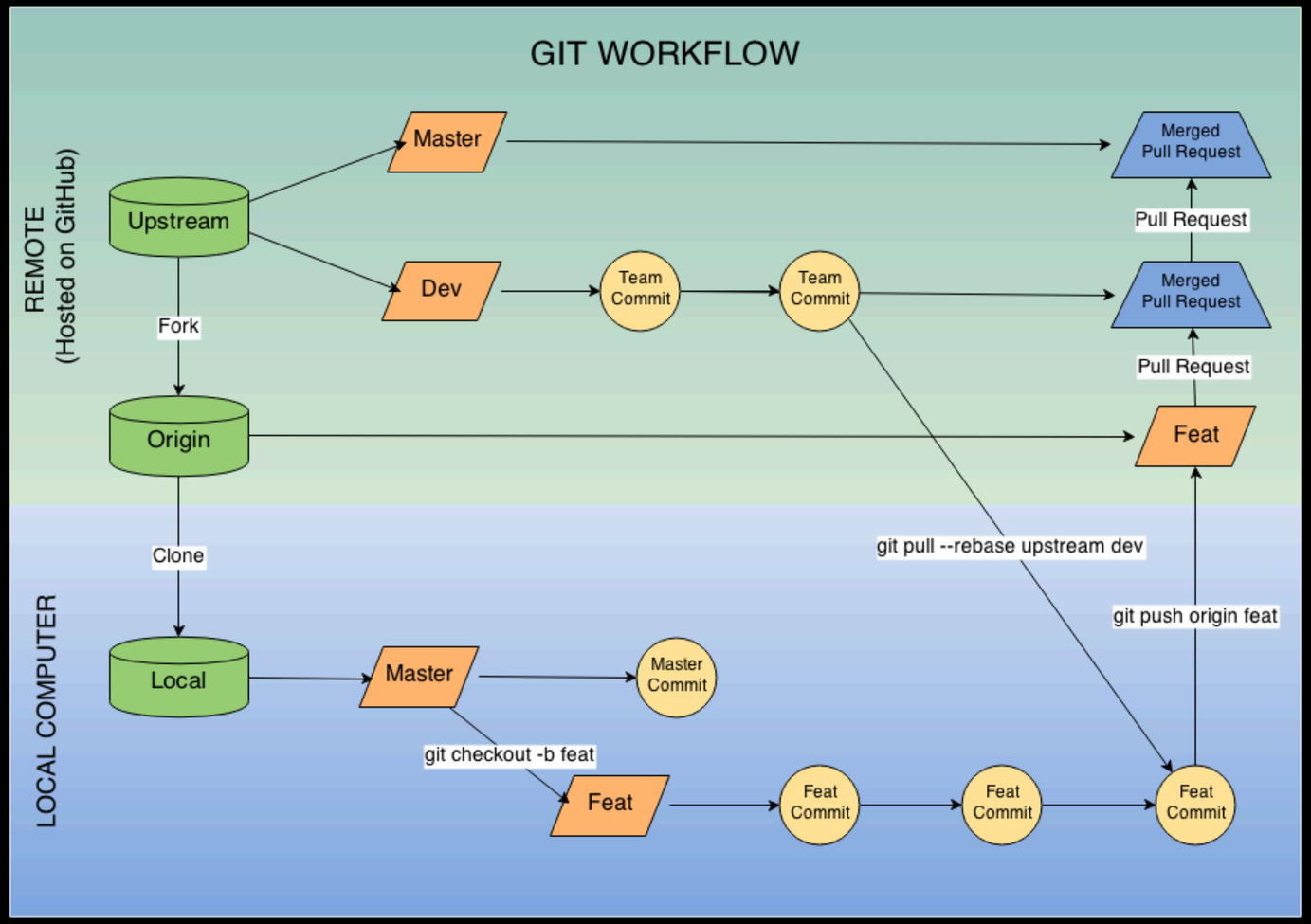
ESSENTIAL GIT COMMANDS



ESSENTIAL GIT COMMANDS



ADVANCED CONCEPTS



GIT AND PROBLEMS

Common pitfalls (quick fixes)

- “Permission denied (publickey)” → SSH key not added to GitHub or not loaded in ssh-agent. Re-add with ssh-add and confirm ssh -T git@github.com.
- Wrong remote URL → Check with git remote -v. Switch method (SSH/HTTPS) if needed:
  - git remote set-url origin <new-url>

TASK

Add Yourself to the Notable Mentions

In this task, you will practice the Git & GitHub workflow: clone, branch, commit, push, and create a Pull Request (PR).

You'll add your details into the repository:

- Go to the repository: https://github.com/Bakel-Bakel/robotics-core-workshop-oct2025
- Fork the repository (click the Fork button on GitHub).

- Clone your fork to your local machine:
  - git clone https://github.com/<your-username>/robotics-core-workshop-oct2025.git
  - cd robotics-core-workshop-oct2025

- Create a new branch with your name:
  - git checkout -b add-yourname

- Navigate to the folder:
  - cd notable-mention
  - Create your text file and add your details:
  - Firstname-Lastname.txt

TASK

- Inside the folder notable-mention/, create a new text file named after yourself using the
  - Firstname-Lastname.txt
  - Example:
  - Bakel-Bakel.txt

- Open the file and add your details in the following format:
- Full-Name , Linkedin-url , Role (student)
- git add notable-mention/Firstname-Lastname.txt
- git commit -m "Add my name to notable mentions"
- Push your branch:
- git push origin add-yourname
- On GitHub, open a Pull Request from your fork’s branch → main repository.

Completion Criteria

- Your .txt file is added in the notable-mention/ folder.
- It contains your full name and LinkedIn link.
- A Pull Request is submitted and visible on the original repository.

ASSIGNMENT

Assignment 1: “My Setup Journey”

- Each student documents how week 1 setup was for them in a README.md
- Steps:
  a. Create a new repo on GitHub (name: week1-setup or whatever).
  b. Clone it to their computer.
  c. Add a README.md file describing their installation process.
  d. Stage + commit + push the file.

Assignment 2: “Tools I Learned”

- Each student writes a short paragraph in tools.md about the tools they used/learned in class (e.g., Git, GitHub, VS Code, WSL, etc).
- They then push this file to GitHub.

BONUS KNOWLEDGE

CENTRALIZED VS DISTRIBUTED VERSION CONTROL

- Centralized Version Control (e.g., SVN, CVS)
  - One central server stores the code.
  - Developers must connect to that server to commit or update.
  - If the server goes down → nobody can collaborate.

- Distributed Version Control (e.g., Git, Mercurial)
  - Every developer has a full copy of the repository (with complete history).
  - Work can continue offline, commits are local first.
  - Collaboration happens by syncing changes (push/pull) to a remote repo like GitHub.
  - No single point of failure – if GitHub goes down, you still have the repo locally.

- Takeaway
- Centralized: Server is the boss → single point of truth.
- Distributed (Git): Everyone is a boss → multiple copies, resilient, faster.


THANK YOU
Adobe Stock | #625194419