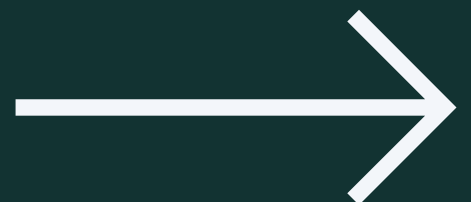


# Git Cheatsheet For Small Business Owners *Supercharge Your Workflow*

Swipe through to learn Git basics and commands  
you can use today!

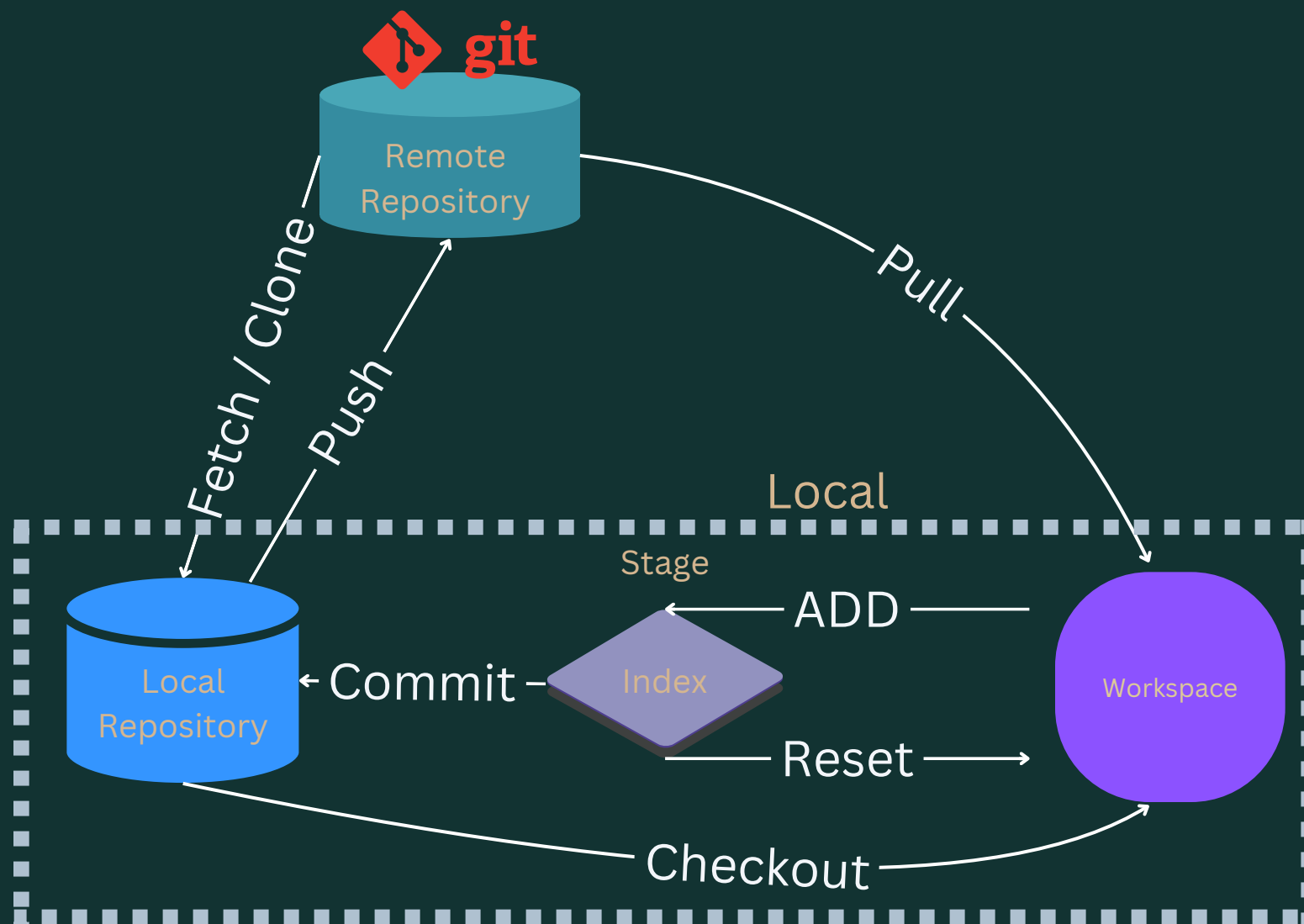


# Why small businesses need Git?

- Track changes to files effortlessly.
- Collaborate with your team seamlessly.
- Safeguard your data with version control.
- Cost-effective and open source!



# What is Git?



A version control system that helps manage changes in your files and projects.

Local vs. Remote Repositories:

- **Local:** Stored on your computer.
- **Remote:** Stored on platforms like GitHub or GitLab.

Version control is essential for managing changes in any project. Whether you're a developer or a small business owner exploring tech, understanding Git can help you streamline collaboration and track your progress. Here's a quick Git cheatsheet to get you started, with a detailed guide linked at the end.



# Quick Git Basics

## Key Terms:

- **Local Repo (Repository):** A directory on your computer that stores all the files and history of your project.
- **Remote Repository:** An online version of your local repository, hosted on platforms like GitHub, GitLab, or BitBucket.
- **Cloning:** The act of making a duplicate of a repository in a new directory.
- **Commit:** A saved snapshot of your project that you can revisit.
- **Branch:** A separate copy of the project for isolated development work without affecting the main code.
- **Merge:** The process of combining changes from different branches into one.

## Advanced Terms:

- **.gitignore File:** A configuration file that specifies files or directories Git should ignore (e.g., private files or large datasets).
- **Staging Area:** A temporary space where changes are held before being committed.
- **Git Stash:** A temporary storage for changes you're not ready to commit but want to save for later.
- **Commit ID (Hash):** A unique identifier for each commit, used to reference specific snapshots.
- **HEAD:** A pointer to the most recent commit in the current branch.



# Setting Up Git

Installation:

- **Mac:** \$ brew install git
- **Linux:** \$ sudo apt-get install git
- **Windows:** Download from [Git for Windows](#). follow the steps.



# Credentials & Configuration:

# Set your email

\$ git config --global user.email "you@example.com"

# Set your name

\$ git config --global user.name "Your Name"

# Create an alias named gc for the git commit command

\$ git config --global alias.gc commit

\$ gc -m "New commit"

# Create an alias named ga for the git add command

\$ git config --global alias.ga add

# List configurations

\$ git config --list



# Daily Git Commands

## Starting a Repo:

# Create a new repository

\$ git init

# Clone an existing repository

\$ git clone <repo\_url>

## Tracking Changes:

# Add files to staging

\$ git add <file\_name>

# Save a snapshot of staged changes

\$ git commit -m "Your commit message"

# Combine adding and committing

\$ git add . && git commit -m "Quick commit"

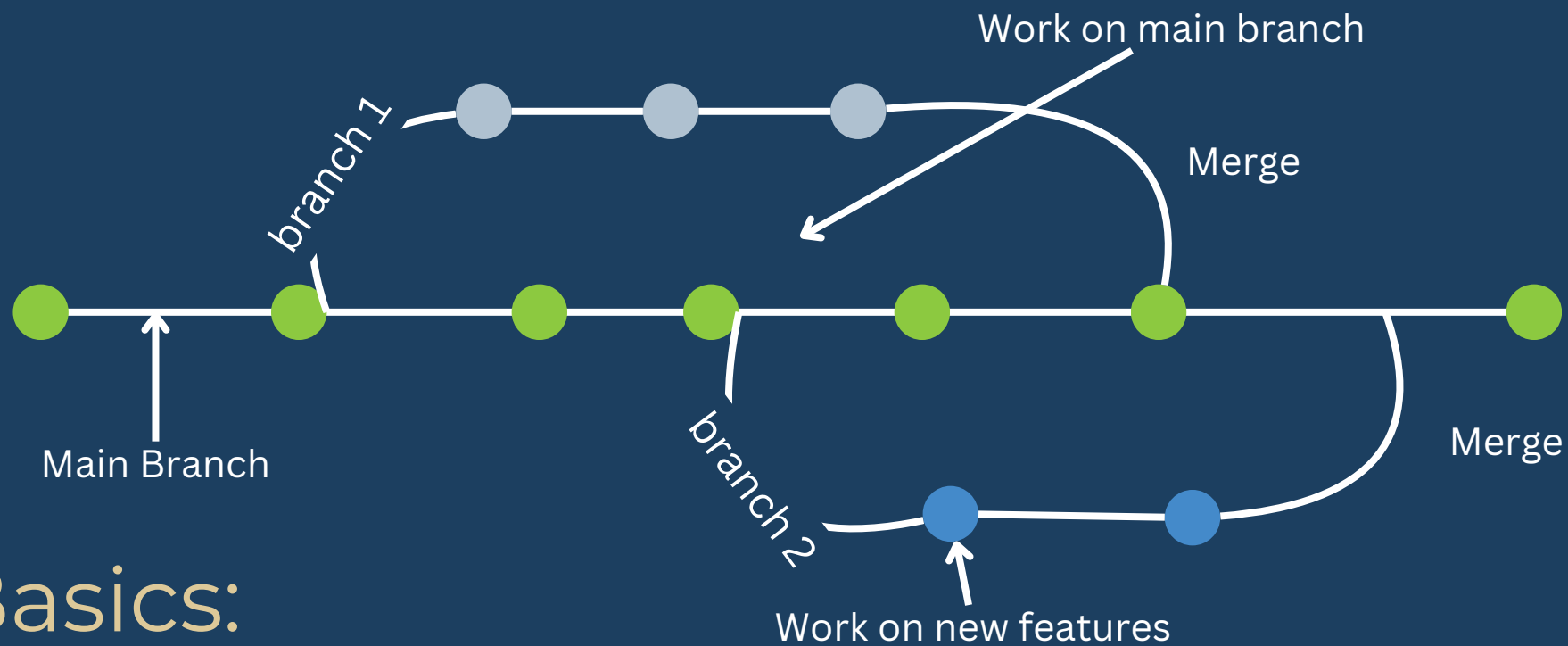
## Viewing Changes:

# Compare changes since last commit

\$ git diff

Tip: Remember to always write meaningful commit messages!

# Working with Branches



## Basics:

# List branches

\$ git branch

# Create a new branch

\$ git branch <branch\_name>

# Switch to a branch

\$ git checkout <branch\_name>

# Merge a branch into the main branch

\$ git checkout main

\$ git merge <branch\_name>

# Safe delete (merged branches)

\$ git branch -d <branch\_name>

# Force delete (unmerged branches)

\$ git branch -D <branch\_name>



# Collaborating with a Team



## Syncing Changes:

# View changes to your files.

\$ git status

# Pull updates from the remote repo

\$ git pull

# Push commits to the remote repo

\$ git push

# Undo a specific commit without affecting later

git revert <commit\_id>

## Managing Remotes:

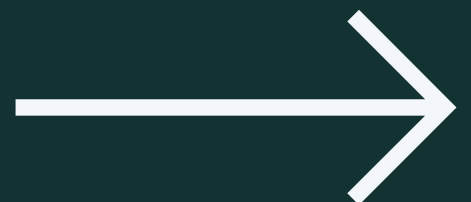
# Add a remote repository

\$ git remote add origin <url>

# Remove a remote repository

\$ git remote rm origin

Tips: Always check git status before committing!



# Advanced Commands

## Fixing Mistakes:

# Undo the latest commit (keep changes)

\$ git reset HEAD~1

# Discard all changes since last commit

\$ git reset --hard HEAD~1

## Stashing:

# Temporarily save changes

\$ git stash

# Reapply saved changes

\$ git stash pop

## Logs and History:

# View commit history

\$ git log

# Recover lost commits

\$ git reflog

Start using Git today and streamline your business!

[www.aitransformers.org](http://www.aitransformers.org)



# Learn More ...

For a deeper dive into Git workflows, troubleshooting conflicts, and advanced commands, check out our detailed blog post [here](#). You can also read it on our website [here](#)

Empower your small business with effective version control today! Git isn't just for developers; it's a tool for anyone managing projects and collaborations.

