

Git and Github Master Class



What is VCS?

VCS stands for **V**ersion **C**ontrol **S**ystem. It is a tool that helps manage and track changes to source code or other files over time. VCS is essential for software development and other projects where maintaining a history of changes, collaboration, and versioning is critical.

Popular VCS Tools:

- **Git:** The most widely used DVCS; supports branching and distributed workflows.
- **Subversion (SVN):** A CVCS used in enterprise applications.
- **Mercurial:** Another DVCS, simpler than Git in some aspects.
- **Perforce:** A CVCS often used for large-scale enterprise projects.

Popular VCS Tools:

- Git: The most widely used DVCS; supports branching and distributed workflows.
- Subversion (SVN): A CVCS used in enterprise applications.
- Mercurial: Another DVCS, simpler than Git in some aspects.
- Perforce: A CVCS often used for large-scale enterprise projects.

Why do we need VCS?

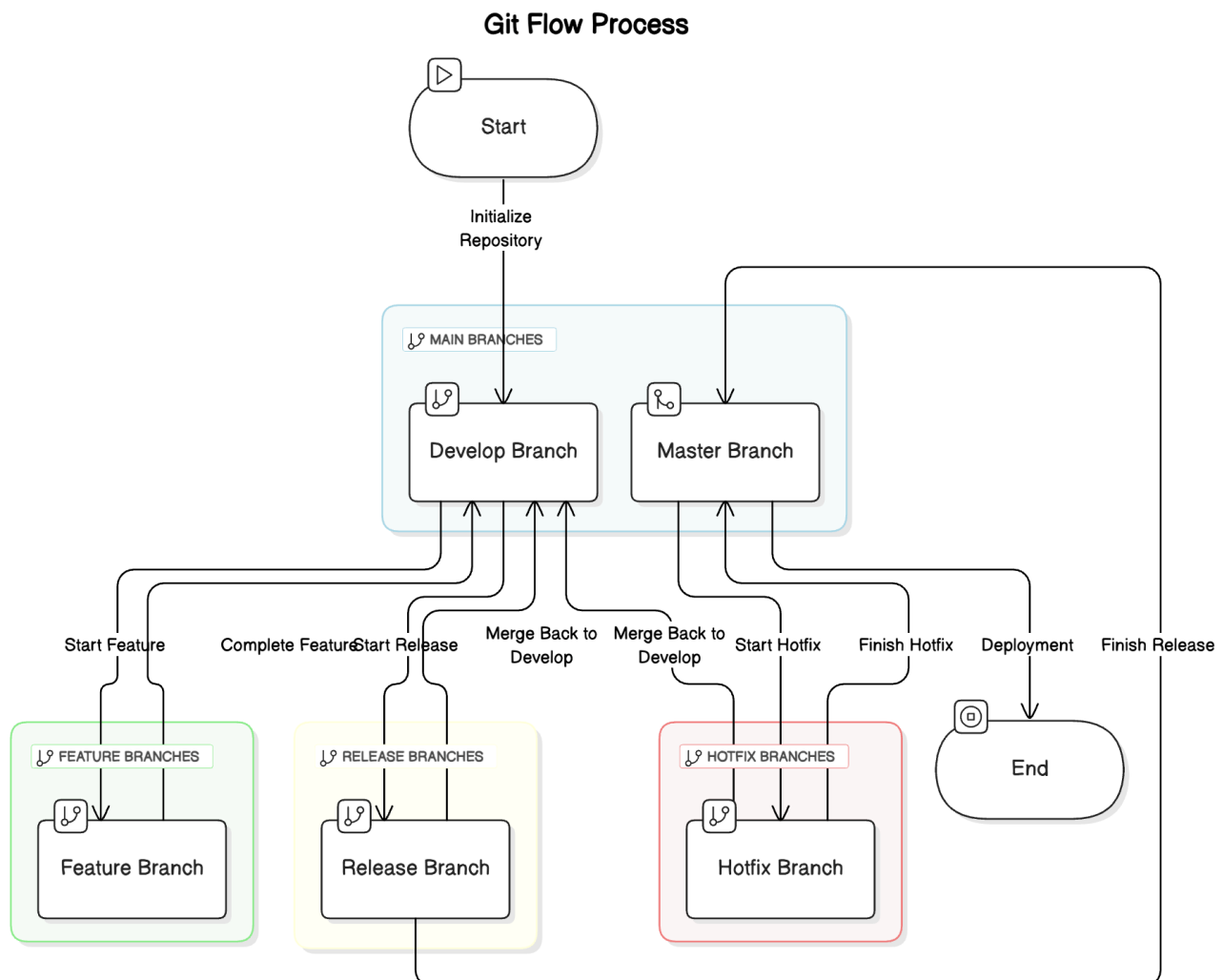
1. **Tracking Changes:** It records changes to files over time, enabling developers to see who changed what and when.

2. **Collaboration:** Multiple people can work on the same project simultaneously without overwriting each other's work.
3. **Branching and Merging:** Developers can create separate branches for different features or experiments and later merge them into the main project.
4. **Version History:** It keeps a history of all changes, making it easy to revert to previous versions if needed.
5. **Conflict Resolution:** Helps manage and resolve conflicts when multiple developers make changes to the same file.
6. **Backup and Recovery:** Acts as a **backup** for the project.

Introduction to Git

Git is a **version control system (VCS)** designed to track changes in source code and collaborate efficiently with others. It is widely used for software development due to its speed, flexibility, and support for non-linear workflows (e.g., branching and merging).

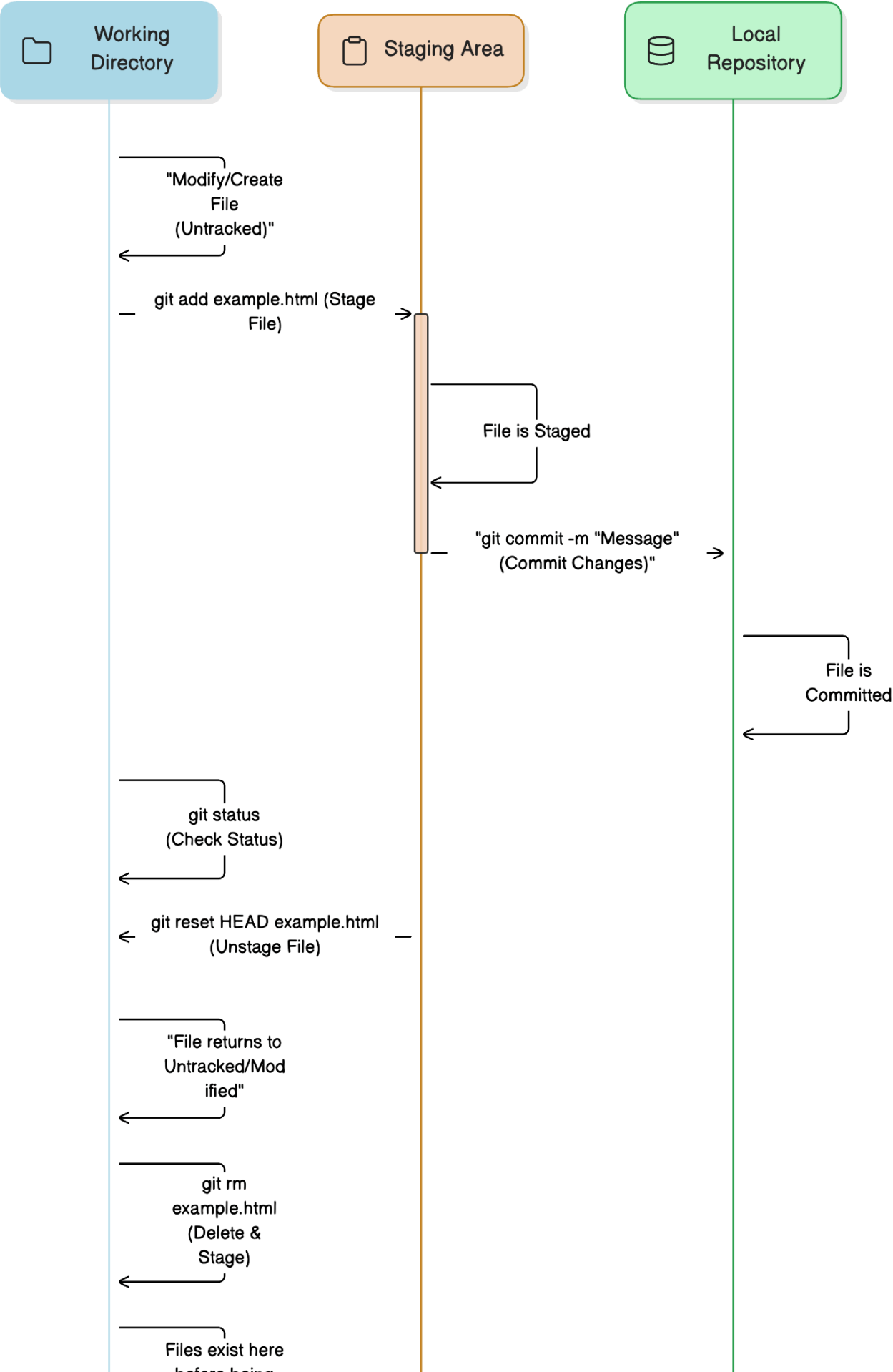
This what a typical git workflow in a startup or enterprise application code could look like.

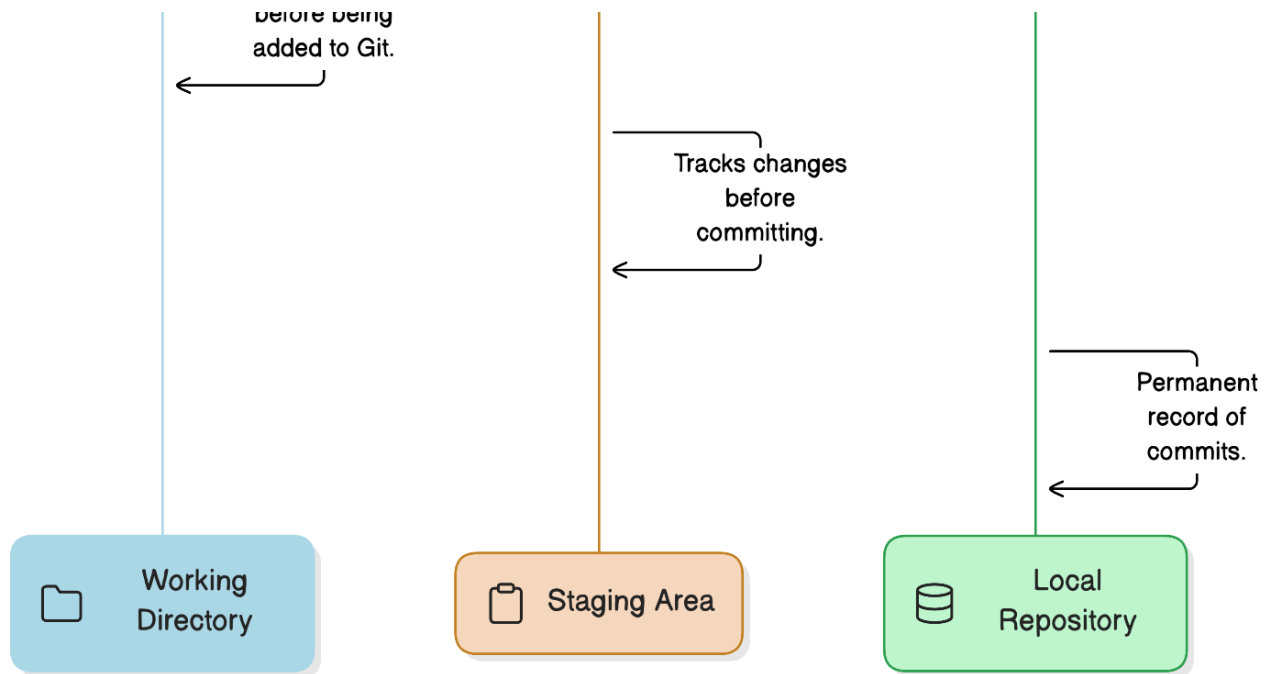


Git CheatSheet

Command	Description
<code>git init</code>	Initialize a new Git repository
<code>git clone <url></code>	Clone a repository from a URL
<code>git status</code>	Show the status of the working directory
<code>git add <file></code>	Stage a file for commit
<code>git add .</code>	Stage all changes in the current directory
<code>git commit -m "message"</code>	Commit staged changes with a message
<code>git push</code>	Push commits to a remote repository
<code>git pull</code>	Fetch and merge changes from a remote repo
<code>git branch</code>	List branches
<code>git branch <name></code>	Create a new branch
<code>git checkout <branch></code>	Switch to a specific branch
<code>git merge <branch></code>	Merge a branch into the current branch
<code>git log</code>	View commit history
<code>git diff</code>	Show differences between working files
<code>git reset <file></code>	Unstage a file
<code>git stash</code>	Save changes without committing
<code>git stash pop</code>	Reapply stashed changes
<code>git remote add <name> <url></code>	Add a remote repository
<code>git fetch</code>	Download objects and refs from another repo
<code>git rebase <branch></code>	Reapply commits on top of another branch

Git Workflow





What is `.git/` Directory?

As soon as we run `git init` command, we get following message in the console:

```
Initialized empty Git repository in /home/Coding/piyushgarg-dev/projects
```

Let's look at what is in the `.git` folder

```
.git
├── config
├── HEAD
├── hooks
│   └── prepare-commit-msg.sample
├── objects
│   ├── info
│   └── pack
├── refs
│   ├── heads
│   └── tags
```

- `config` is a text file that contains your git configuration for the current repo.
- `HEAD` contains the current head of the repo.
- `hooks` contain any scripts that can be run before/after git does anything.
- `objects` contains the git objects, ie the data about the files, commits etc in your repo. We will go in depth into this in this blog.
- `refs` as we previously mentioned, stores references(pointers)

