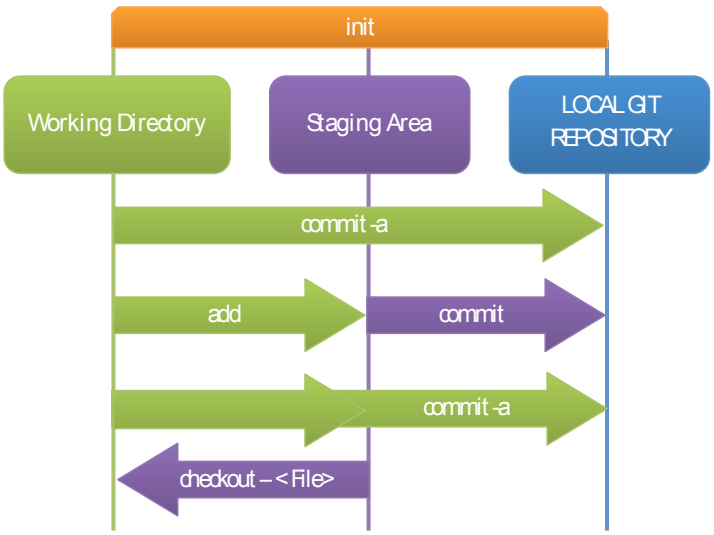
GIT CHEAT SHEET (part I)

This is covered in part I of the git introduction:



CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```
Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```
Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```
Enables helpful colorization of command line output

CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

```
$ git init [project-name]
```
Creates a new local repository with the specified name

```
$ git clone [url]
```
Downloads a project and its entire version history

MAKE CHANGES

Review edits and craft a commit transaction

```
$ git status
```
Lists all new or modified files to be committed

```
$ git diff
```
Shows file differences not yet staged

```
$ git add [file]
```
Snapshots the file in preparation for versioning

```
$ git diff --staged
```
Shows file differences between staging and the last file version

```
$ git reset [file]
```
Unstages the file, but preserve its contents

```
$ git commit -m "[descriptive message]"
```
Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine completed efforts

```
$ git branch
```
Lists all local branches in the current repository

```
$ git branch [branch-name]
```
Creates a new branch

```
$ git checkout [branch-name]
```
Switches to the specified branch and updates the working directory

```
$ git merge [branch]
```
Combines the specified branch's history into the current branch

```
$ git branch -d [branch-name]
```
Deletes the specified branch

REFACTOR FILENAMES

Relocate and remove versioned files

```
$ git rm [file]
```
Deletes the file from the working directory and stages the deletion

```
$ git rm --cached [file]
```
Removes the file from version control but preserves the file locally

```
$ git mv [file-original] [file-renamed]
```
Changes the file name and prepares it for commit

REVIEW HISTORY

Browse and inspect the evolution of project files

```
$ git log
```
Lists version history for the current branch

```
$ git log --follow [file]
```
Lists version history for a file, including renames

```
$ git diff [first-branch]...[second-branch]
```
Shows content differences between two branches

```
$ git show [commit]
```
Outputs metadata and content changes of the specified commit

SUPPRESS TRACKING

Exclude temporary files and paths

```
*.log
build/
temp-*
```
A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

```
$ git ls-files --other --ignored --exclude-standard
```
Lists all ignored files in this project

REDO COMMITS

Erase mistakes and craft replacement history

```
$ git reset [commit]
```
Undoes all commits after `[commit]`, preserving changes locally

```
$ git reset --hard [commit]
```
Discards all history and changes back to the specified commit