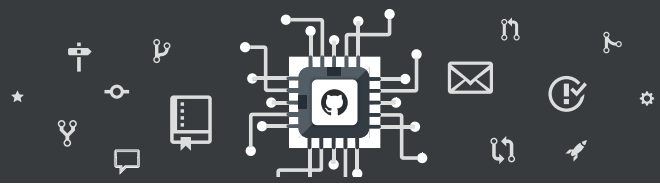


Git Cheat Sheet

/dschetel/GitCheatSheet

Installation & Configuration

Install Git

http://git-scm.com

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]`
Sets the email you want attached to your commit transactions

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

`$ git config --global core.editor vim`
Set standard editor for crafting of commit messages

Create a Repository

Start a new repository or obtain one from an existing URL

`$ git init [project name]`
Creates a new local repository

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

Supress tracking

```
build/
*.aux
```

A textfile named **.gitignore** suppresses accidental versioning of files and paths matching the specified patterns.

Use https://www.gitignore.io/ to generate .gitignore files.

Basic vi / vim commands

`i`	Enter insert mode
`ESC`	Exit insert mode
`:w`	Save file
`:q`	Exit if no changes have been made
`:q!`	Exit and undo made changes

Make changes

`$ git status`
Show the working tree status

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

`$ git add [file]`
Add file contents to the staging area

`$ git add .`
Add all modified contents to the staging area

`$ git add -p`
Add modified contents to the staging area in parts

`$ git diff --cached`
Shows file differences between staged files to HEAD

`$ git diff [commit1] [commit2]`
Shows file differences between commit1 and commit2

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

`$ git commit -m „commit message“`
Record changes to the repository

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

`$ git tag`
Create, list, delete or verify a tag object

Working with Branches

`$ 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-name]`
Combines the specified branch's history into the current branch

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

Stashing

`$ git stash`
Temporarily stores all modified tracked files

`$ git stash pop`
Restores the most recently stashed file

`$ git stash list`
Lists all stashed change-sets

`$ git stash drop`
Discards the most recently stashed change-set

Redo commits

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

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

Synchronize

`$ git remote add [bookmark] [url]`
Set a new repository

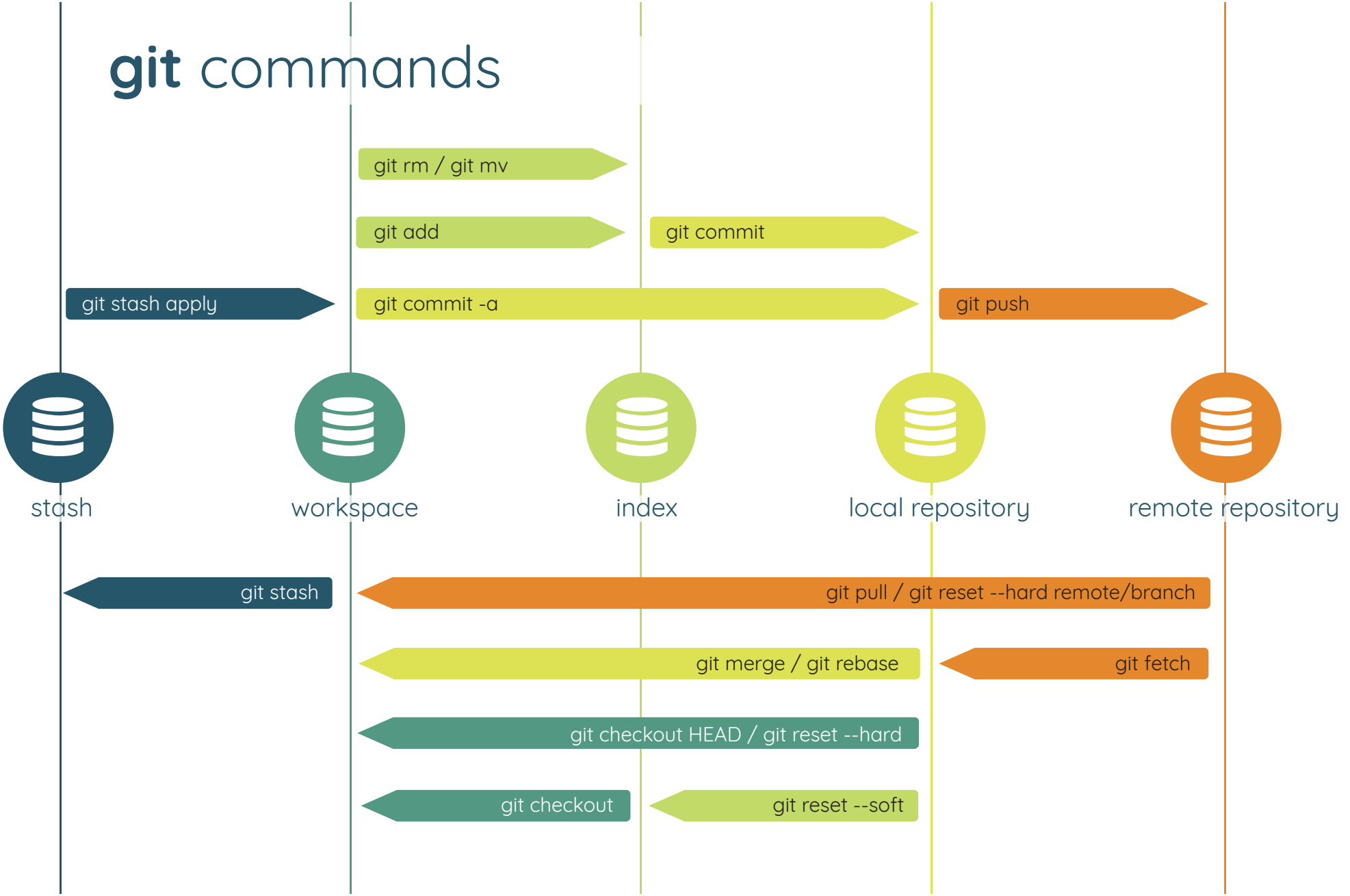
`$ git fetch [bookmark]`
Downloads all history from the repository

`$ git merge [bookmark]/[branch-name]`
Combines bookmark's branch into local branch

`$ git pull`
Downloads bookmark history and incorporates changes

`$ git push [bookmark] [branch-name]`
Uploads all local branch commits to the remote repository

master is the default mainline branch
origin is the default upstream repository
HEAD is the current branch


git commands
git rm / git mv
git add
git commit
git stash apply
git commit -a
git push
stash
workspace
index
local repository
remote repository
git stash
git pull / git reset --hard remote/branch
git merge / git rebase
git fetch
git checkout HEAD / git reset --hard
git checkout
git reset --soft