My terminal command cheatsheet

Anna Thomas

November 16, 2023

Add new console, classlibrary, xunit, solution

```
dotnet new console
dotnet new classlib
dotnet new xunit
dotnet new sln
```

Add e.g. solution with name

```
dotnet new sln --name mySolution
```

List all files in solution

```
dotnet sln list
```

Adds one or more project to the solution file

```
dotnet sln mySolution.sln add projectpath/myProj
```

Read file, listen, create file

```
cat
echo
touch
```

Create new .git repository

```
git init
```

Create a new repository on the command line

```
echo "# mygitrepository" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:fruniia/mygitrepository.git
git push -u origin main
```

Push an existing repository from the command line

```
git remote add origin git@github.com:fruniia/mygitrepository.git
git branch -M main
git push -u origin main
```

Commands in git

```
git status          show status of working directory
    -s          short version
git add             adds file(s) to the staging area
git commit          adds a commit to git
    -m          message
git log             shows previous commits in branch
    --all
    --decorate
    --oneline
    --graph
git show            take a closer look at commit
git diff            shows changes between your working directory and index/staging area
```

Help with a command

```
--h prefix works with most commands and will print the help text for said command

e.g. diff --h       will print instructions for diff command
                    (not to be confused with git diff)
Notable exception:
echo --h            will not print help on the echo command, instead try
/bin/echo --help
```

Branching in git

```
branch              show and create branches
 -d <branch name>   delete branch
checkout            move HEAD to another branch or commit
 -b <branch name>   create new branch and change to that branch
                    does not change working directory
merge               bring changes from one branch into another
```

Remote in git

```
clone               clone a repo to a local .git repo
remote              show, create and change linked repos
push                send local changes to remote
pull                get changes from remote
fetch               update local info about remote
```

Undo in git

When working in a shared repo it is important that the commit history doesn't change, instead of removing the previous commit add the undos into a new commit.

```
checkout            get a commit, undo all changes in workspace
revert              create a new commit which undos all changes
reset               remove files from staging or remove a LOCAL commit
    --soft
    --mixed
    --hard
commit --amend      replaces the last commit with your new, improved commit
```