Git Commands

Remote Commands

clone

This command downloads the contents of a repository to the current folder.

usage:

```
git clone {repository url}

git clone https://github.com/solarknightsracing/SKR-Core.git

git clone https://github.com/solarknightsracing/1_Solaris.git
```

note: This command will only be used the first time that you copy a repository to your computer.

fetch

This command will download all of the changes made to the repository since the last time you fetched.

usage:

```
git fetch
```

note: This command is distinct from “git pull” because it does not apply any changes to your machine’s repository.

pull

This command essentially runs a “git fetch” and then preforms a “merge” which syncs the current state of the Github repository with your local machine.

usage:

```
git pull --ff-only
```

note: This command is distinct from “git clone” because it does not require the repository’s url. It is also distinct from “git fetch” because it applies the changes that were retrieved from the server.

push

This command will upload the commits you have created on your local machine to the server. Effectively, applying your changes.

usage:

```
git push

git push -u origin my-change-branch
```

note: the -u flag is neccesary if the branch that you are pushing to does not exist on the remote yet. to use it,

Local commands

commit

This command records the changes you have staged into your local repository. You must include a short commit message.

usage:

```
git commit -m "{commit message}"

git commit -m "Created part1.SLDPRT" # For the love of god do not upload something called "part1"
```

note:

add & reset

The “git add” command stages files to be included in the next commit. The “git reset” command is the exact opposite of “git add”. It unstages files, exluding them in the next commit.

usage:

```
git add {files}

git add .

git add file.txt
```

note: all files are unstaged by default.

status

This command displays information about the current state of your machine’s repository. Useful information like: which files have been changed, which of those files are staged to be included in the next commit, and which branch you are currently on.

usage:

```
git status
```

note: all files are unstaged by default.

log

This command shows the history of previous commits with details like the author, the date, and any commit messages.

usage:

```
git log

git log --oneline --graph --all
```

note: all files are unstaged by default.

Branching commands

branch

This command lists, creates, or deletes branches. With no other arguments, it will list all branches.

usage:

```
git branch

git branch my-change-branch

git branch -d my-change-branch
```

note: all files are unstaged by default.

switch

This command displays information about the current state of your machine's repository. Useful information like: which files have been changed, which of those files are staged to be included in the next commit, and which branch you are currently on.

usage:

```
git switch my-change-branch

git switch main
```

note: all files are unstaged by default.

Git LFS Commands

locks

This command displays all currently locked files.

usage:

```
git lfs locks
```

note: all files are unstaged by default.

lock & unlock

This command will lock whatever file(s) you pass it. An lfs lock will prevent others from modifying a file at the same time as you.

usage:

```
git lfs lock Part1.SLDPRT

git lfs lock ./folder-name/
# this will lock everything in the "folder-name" folder

git lfs unlock .
# this will unlock everything within the current folder
```

note: all files are unstaged by default.

Example git workflows

cloning a remote repository & viewing it's history

First, clone the repository. Navigate to the repository's page on github and copy the URL from the address bar of your browser.

```
git clone https://github.com/SolarKnightsRacing/1_solaris.git
Cloning into '1_solaris'...
remote: Enumerating objects: 328, done.
remote: Counting objects: 100% (99/99), done.
remote: Compressing objects: 100% (75/75), done.
remote: Total 328 (delta 65), reused 44 (delta 24), pack-reused 229 (from 1)
Receiving objects: 100% (328/328), 47.53 MiB | 8.37 MiB/s, done.
Resolving deltas: 100% (103/103), done.
Filtering content: 100% (27/27), 28.30 MiB | 2.84 MiB/s, done.
```

This will create a "working copy" of the repository on your machine. In this case the folder will be named "1_solaris". To move into that folder without leaving the terminal you can type:

```
cd 1_solaris
```

Check to see if everything worked by running git log.

```
git log --oneline
f4eca50 (HEAD -> main, origin/main, origin/HEAD) Delete Chassis/Assemblies directory
e4ef2c5 Fix ADR template
40df873 delete onboarding example
```

fixing a typo in a document

fix typo

```
git add .
```

```
git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
  modified:   document-with-a-typo
```

```
git commit -m "fixed typo"
[main 0499459] Fixed typo
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 701 bytes | 701.00 KiB/s, done.
Total 8 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/solarknightsracing/1_solaris.git
  main -> origin/main
```

creating a new part

create a new part

```
git branch new-part
```

```
git switch new-part
```

```
git add .
```

```
git commit -m "created a new part"
```

```
git push -u origin new-branch
```

Finalizing a part's changes

```
git switch new-part
```

```
git lfs lock part1.SLDPRT
```

include revisions from team members

```
git add .
```

```
git commit -m "Final changes to new part
```

```
git push
```

```
git lfs unlock part1.SLDPRT
```

Glossary

branch commit repository remote