

branch commit repository remote

## Git Commands

### git clone

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

usage:

```
git clone {repository url}
git clone https://github.com/solarknightsra
cing/SKR-Core.git
git clone https://github.com/solarknightsra
cing/1_Solaris.git
```

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

### git pull

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

usage:

```
git pull
git pull --ff-only #use this command 99% of
the time as it will prevent you from
encountering a "merge conflict"
```

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.

### git fetch

This command will download all of the changes made to the repository .

usage:

```
git pull
git pull --ff-only #use this command 99% of
the time as it will prevent you from
encountering a "merge conflict"
```

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.

### git push

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

usage:

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

note:

### git commit

This command records staged changes to your local repository. You must include a short commit message.

usage:

```
git commit -m "{commit message}"
git commit -m "Created part1.SLDPR" # For
the love of god do not call something
"part1"
```

note:

### git add & git 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, excluding them in the next commit.

usage:

```
git add {files}
git add .
git add file.txt
```

note: all files are unstaged by default.

### git 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.

## git 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.

## git 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.

## git 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

## git lfs locks

This command displays all currently locked files.

usage:

```
git status
```

note: all files are unstaged by default.

## git lfs lock & git lfs unlock

This command will lock whatever files you pass it. Places a lock on a file to prevent others from modifying it

usage:

```
git status
```

note: all files are unstaged by default.

## Example git workflows

### cloning a remote repository & viewing it's history

git clone cd git log

### fixing a typo in a document

fix typo git add . git commit -m "fixed typo" git push

### 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
```