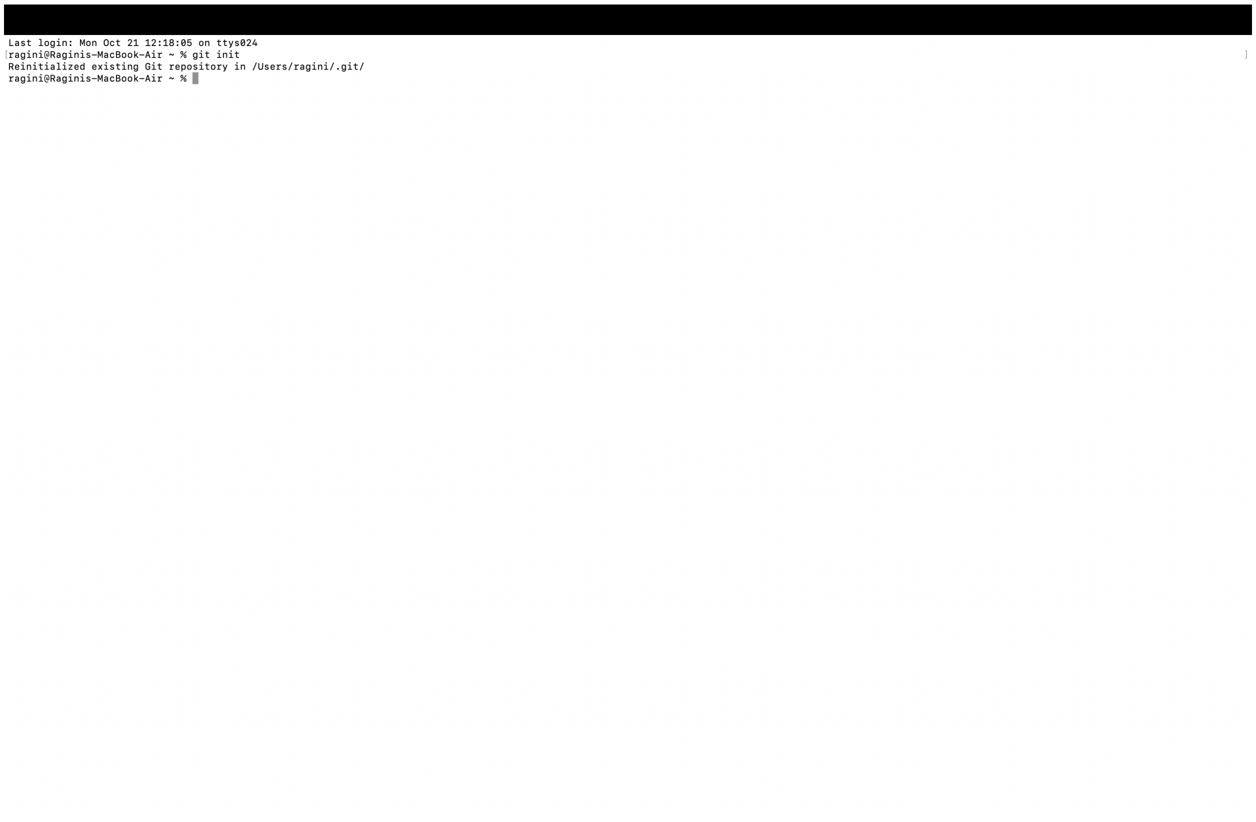
Git - punishment

1 >> Write all initialization command in detailed.

Ans :-

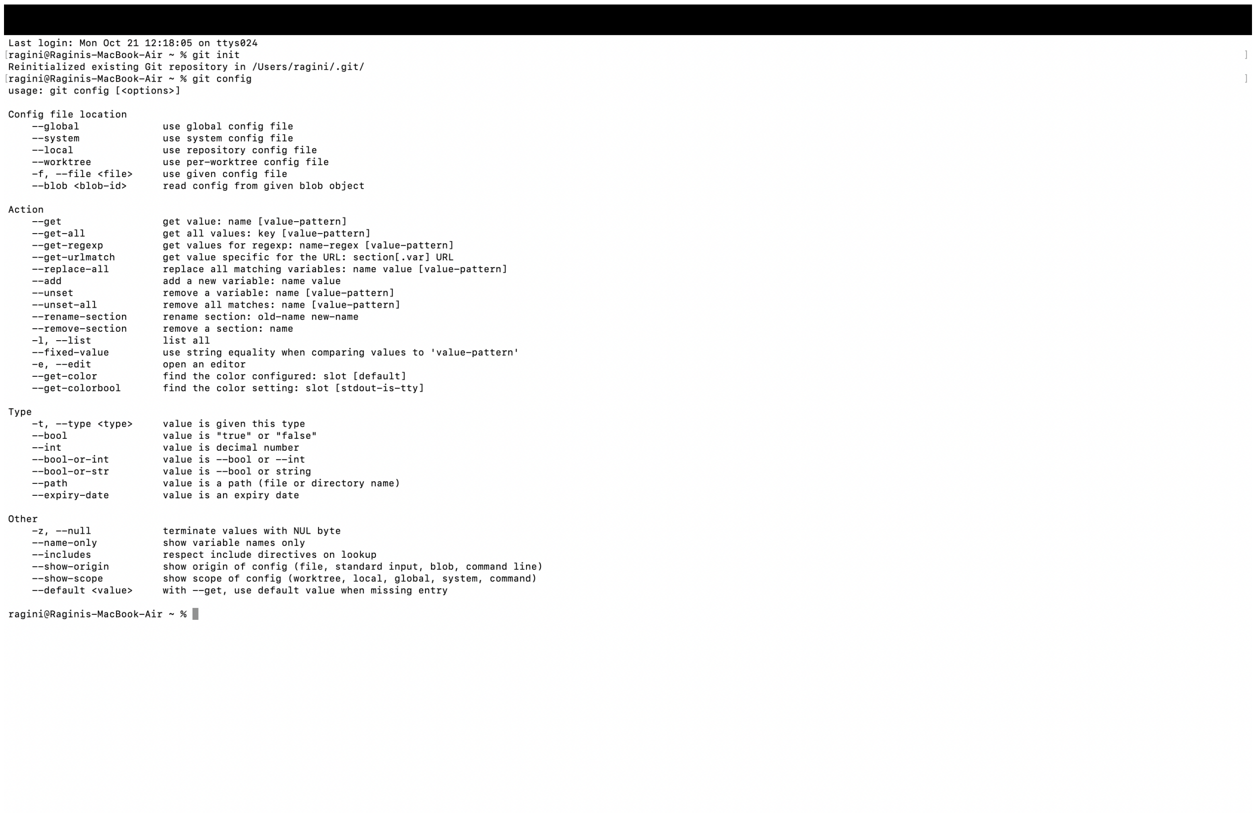
1. Git init :-
   This command initializes a new Git repository in the current directory. It creates a hidden .git folder that contains all the necessary metadata for version control.



[redacted]

```
Last login: Mon Oct 21 12:18:05 on ttys024
ragini@Raginis-MacBook-Air ~ % git init
Reinitialized existing Git repository in /Users/ragini/.git/
ragini@Raginis-MacBook-Air ~ % 
```

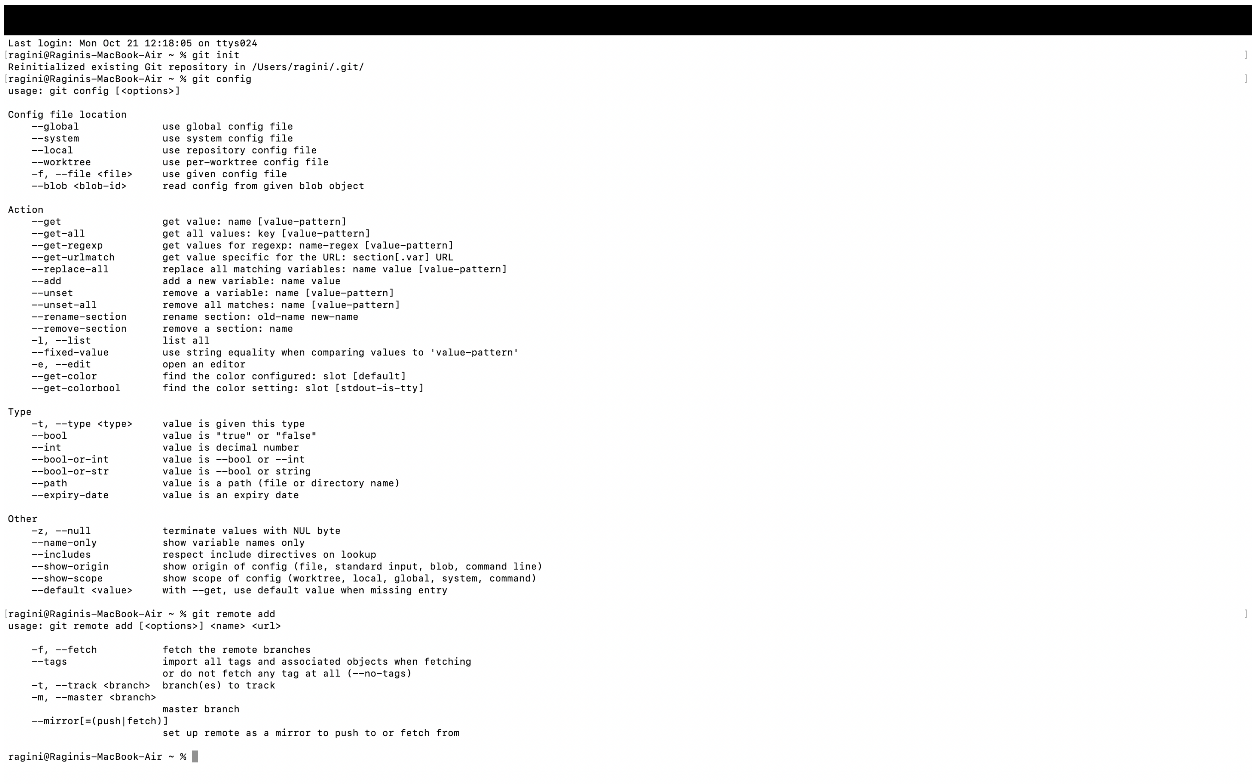
2. Git config :-
   This command is used to set configuration values on a global or local project level. It's crucial for setting up your identity for commits.



[redacted]

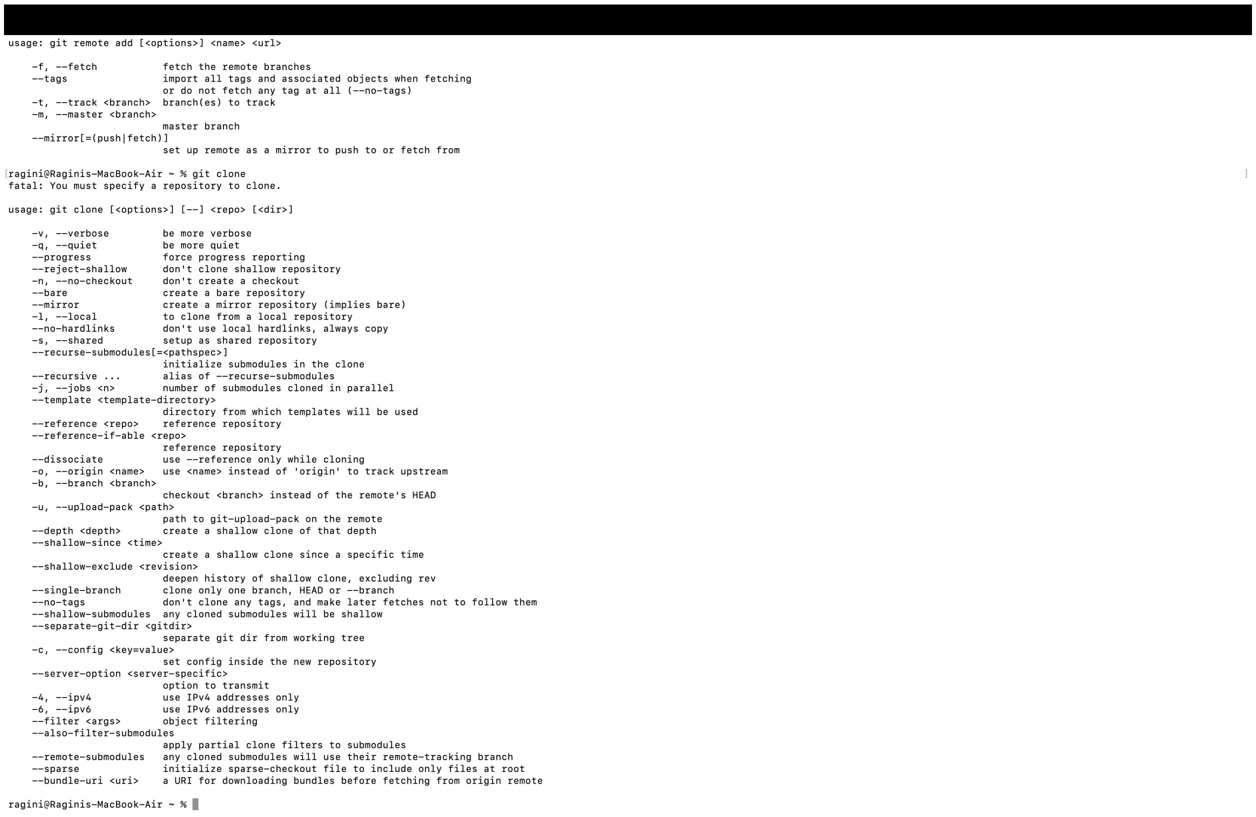
```
Last login: Mon Oct 21 12:18:05 on ttys024
ragini@Raginis-MacBook-Air ~ % git init
Reinitialized existing Git repository in /Users/ragini/.git/
ragini@Raginis-MacBook-Air ~ % git config
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-pattern]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-pattern]
    --unset-all           remove all matches: name [value-pattern]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    --fixed-value         use string equality when comparing values to 'value-pattern'
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    -t, --type <type>     value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --bool-or-str         value is --bool or string
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, command line)
    --show-scope          show scope of config (worktree, local, global, system, command)
    --default <value>     with --get, use default value when missing entry

ragini@Raginis-MacBook-Air ~ %
```

3. Git remote add :-
   This command is used to add a remote repository to your local Git repository. It's typically used when you want to connect your local repository to a remote one, like on GitHub.



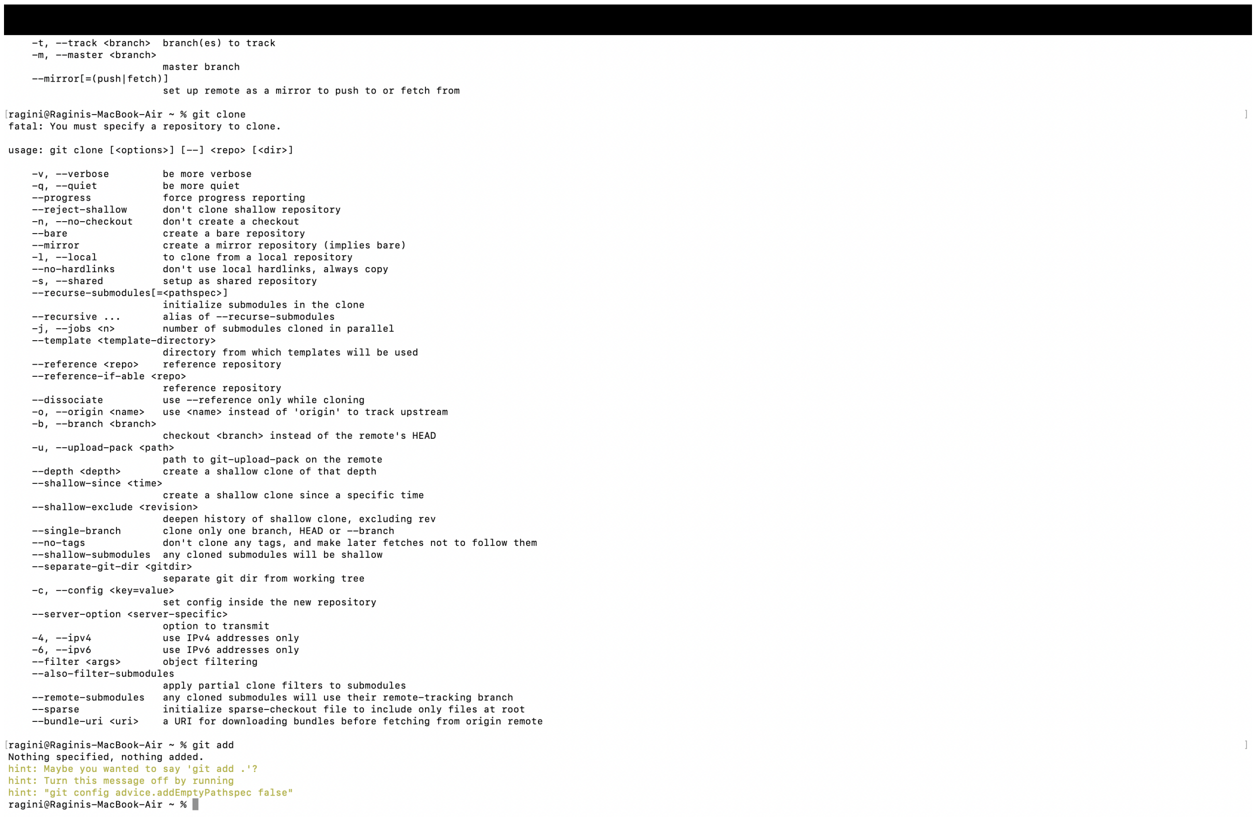
```
Last login: Mon Oct 21 12:18:05 on ttys024
ragini@Raginis-MacBook-Air ~ % git init
Reinitialized existing Git repository in /Users/ragini/.git/
ragini@Raginis-MacBook-Air ~ % git config
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-pattern]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-pattern]
    --unset-all           remove all matches: name [value-pattern]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    --fixed-value         use string equality when comparing values to 'value-pattern'
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    -t, --type <type>     value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --bool-or-str         value is --bool or string
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, command line)
    --show-scope          show scope of config (worktree, local, global, system, command)
    --default <value>     with --get, use default value when missing entry

ragini@Raginis-MacBook-Air ~ % git remote add
usage: git remote add [<options>] <name> <url>

    -f, --fetch           fetch the remote branches
    --tags                import all tags and associated objects when fetching
                          or do not fetch any tag at all (--no-tags)
    -t, --track <branch>  branch(es) to track
    -m, --master <branch>
                          master branch
    --mirror[=(push|fetch)]
                          set up remote as a mirror to push to or fetch from

ragini@Raginis-MacBook-Air ~ % 
```

4. Git clone :-
   This command is used to create a copy of an existing Git repository. It's commonly used to get a local copy of a remote repository.



```
usage: git remote add [<options>] <name> <url>

    -f, --fetch           fetch the remote branches
    --tags                import all tags and associated objects when fetching
                          or do not fetch any tag at all (--no-tags)
    -t, --track <branch>  branch(es) to track
    -m, --master <branch>
                          master branch
    --mirror[=(push|fetch)]
                          set up remote as a mirror to push to or fetch from

ragini@Raginis-MacBook-Air ~ % git clone
fatal: You must specify a repository to clone.

usage: git clone [<options>] [--] <repo> [<dir>]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --progress            force progress reporting
    --reject-shallow      don't clone shallow repository
    -n, --no-checkout     don't create a checkout
    --bare                create a bare repository
    --mirror              create a mirror repository (implies bare)
    -l, --local           to clone from a local repository
    --no-hardlinks        don't use local hardlinks, always copy
    -s, --shared          setup as shared repository
    --recurse-submodules[=<pathspec>]
                          initialize submodules in the clone
    --recursive ...       alias of --recurse-submodules
    -j, --jobs <n>        number of submodules cloned in parallel
    --template <template-directory>
                          directory from which templates will be used
    --reference <repo>    reference repository
    --reference-if-able <repo>
                          reference repository
    --dissociate          use --reference only while cloning
    -o, --origin <name>   use <name> instead of 'origin' to track upstream
    -b, --branch <branch>
                          checkout <branch> instead of the remote's HEAD
    -u, --upload-pack <path>
                          path to git-upload-pack on the remote
    --depth <depth>       create a shallow clone of that depth
    --shallow-since <time>
                          create a shallow clone since a specific time
    --shallow-exclude <revision>
                          deepen history of shallow clone, excluding rev
    --single-branch       clone only one branch, HEAD or --branch
    --no-tags             don't clone any tags, and make later fetches not to follow them
    --shallow-submodules  any cloned submodules will be shallow
    --separate-git-dir <gitdir>
                          separate git dir from working tree
    -c, --config <key=value>
                          set config inside the new repository
    --server-option <server-specific>
                          option to transmit
    -4, --ipv4            use IPv4 addresses only
    -6, --ipv6            use IPv6 addresses only
    --filter <args>       object filtering
    --also-filter-submodules
                          apply partial clone filters to submodules
    --remote-submodules   any cloned submodules will use their remote-tracking branch
    --sparse              initialize sparse-checkout file to include only files at root
    --bundle-uri <uri>    a URI for downloading bundles before fetching from origin remote

ragini@Raginis-MacBook-Air ~ %
```

5. Git add :-
   This command adds new or changed files in your working directory to the Git staging area. It's an essential command that you'll use in your daily Git workflow.



```
    -t, --track <branch>  branch(es) to track
    -m, --master <branch>
                          master branch
    --mirror[=(push|fetch)]
                          set up remote as a mirror to push to or fetch from

ragini@Raginis-MacBook-Air ~ % git clone
fatal: You must specify a repository to clone.

usage: git clone [<options>] [--] <repo> [<dir>]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --progress            force progress reporting
    --reject-shallow      don't clone shallow repository
    -n, --no-checkout     don't create a checkout
    --bare                create a bare repository
    --mirror              create a mirror repository (implies bare)
    -l, --local           to clone from a local repository
    --no-hardlinks        don't use local hardlinks, always copy
    -s, --shared          setup as shared repository
    --recurse-submodules[=<pathspec>]
                          initialize submodules in the clone
    --recursive ...       alias of --recurse-submodules
    -j, --jobs <n>        number of submodules cloned in parallel
    --template <template-directory>
                          directory from which templates will be used
    --reference <repo>    reference repository
    --reference-if-able <repo>
                          reference repository
    --dissociate          use --reference only while cloning
    -o, --origin <name>   use <name> instead of 'origin' to track upstream
    -b, --branch <branch>
                          checkout <branch> instead of the remote's HEAD
    -u, --upload-pack <path>
                          path to git-upload-pack on the remote
    --depth <depth>       create a shallow clone of that depth
    --shallow-since <time>
                          create a shallow clone since a specific time
    --shallow-exclude <revision>
                          deepen history of shallow clone, excluding rev
    --single-branch       clone only one branch, HEAD or --branch
    --no-tags             don't clone any tags, and make later fetches not to follow them
    --shallow-submodules  any cloned submodules will be shallow
    --separate-git-dir <gitdir>
                          separate git dir from working tree
    -c, --config <key=value>
                          set config inside the new repository
    --server-option <server-specific>
                          option to transmit
    -4, --ipv4            use IPv4 addresses only
    -6, --ipv6            use IPv6 addresses only
    --filter <args>       object filtering
    --also-filter-submodules
                          apply partial clone filters to submodules
    --remote-submodules   any cloned submodules will use their remote-tracking branch
    --sparse              initialize sparse-checkout file to include only files at root
    --bundle-uri <uri>    a URI for downloading bundles before fetching from origin remote

ragini@Raginis-MacBook-Air ~ % git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"
ragini@Raginis-MacBook-Air ~ %
```