githooks(5) Manual Page

NAME

- H G

SYNOPSIS

$GIT_DIR/ /* (` . P `/*)

DESCRIPTION

H

. H .

B `$GIT_DIR/hooks`,

`core.hooksPath` (- (1)).

B G , $GIT_DIR

- . A

(*pre-receive*,*update*, *post-receive*, *post-update*, *push-to-checkout*)

$GIT_DIR.

H , - , .

S .

`git init` , . S

"TEMPLATE DIRECTORY" - (1) . W

" " G .

T .

HOOKS

applypatch-msg

T - (1). I ,
. E - `git am`
.

T ,
. I
.

T *applypatch-msg* , , *commit-msg* ,
.

pre-applypatch

T - (1). I ,
, .

I - ,
.

I
.

T *pre-applypatch* , , *pre-commit* ,
.

post-applypatch

T - (1). I ,
.

T , `git am`.

pre-commit

T - (1), `--no-verify` . I ,
. E -
`git commit` .

T *pre-commit* , , .

A `git commit` `GIT_EDITOR=:` .

T *pre-commit* , `hooks.allownonascii` - ASCII .

pre-merge-commit

T - (1), `--no-verify` . I , . E - `git merge` .

T *pre-merge-commit* , , *pre-commit* , .

T `GIT_EDITOR=:` .

I , (- (1)). A , , *pre-commit* , .

prepare-commit-msg

T - (1). , .

I . T . T , : `message` (`-m` `-F`); `template` (`-t`

`commit.template`); `merge` (
`.git/MERGE_MSG`); `squash` (`.git/SQUASH_MSG`);
`commit`, SHA-1 (`-c`, `-C` `--amend`).

I - , `git commit` .

T ,
`--no-verify` . A -
. I - .

T `prepare-commit-msg` G
.

commit-msg

T - (1) - (1), `--
no-verify` . I ,
. E - .

T ,
. I
.

T *commit-msg* , , "S - - " ,
.

post-commit

T - (1). I ,
.

T , `git
commit`.

pre-rebase

T -(1).
. T . T
. T
, .

post-checkout

T -(1). -(1).
. T : HEAD,
HEAD ( ),
(, =1) (
, =0). T `git switch` `git
checkout`,
.

I -(1), `--no-checkout` (`-n`) . T
- , HEAD
1. L `git worktree add` `--no-checkout` .

T , -
HEAD , .

post-merge

T - (1), `git pull`
. T ,
. T `git
merge` , .

T -
(. .: / ,
ACLS,). S / / . .

pre-push

T - (1). . T
, .

I
:

```
<local ref> SP <local sha1> SP <remote ref> SP <remote sha1> LF
```

F , `git push origin master:foreign`
:

```
refs/heads/master 67890 refs/heads/foreign 12345
```

, 40- SHA-1 . I
`<remote SHA-1>` 40 `0`. I , `<local ref>`
`(delete)` `<local SHA-1>` 40 `0`. I
(`HEAD~`,
SHA-1) .

I - , `git push` .
I
.

pre-receive

T - - (1) `git push`
( ) . J ,
- . I .

T . I ,
:

```
<old-value> SP <new-value> SP <ref-name> LF
```

`<old-value>` , `<new-value>`
`<ref-name>` . W
, `<old-value>` 40 `0`.

I - , . I , *update* .

B `git send-pack` , `echo` .

T `git push --push-option=...` `GIT_PUSH_OPTION_COUNT`, `GIT_PUSH_OPTION_0`, `GIT_PUSH_OPTION_1`, I , . I , , , `GIT_PUSH_OPTION_COUNT=0`.

S "Q E " - - (1) .

update

T - - (1) `git push` ( ) . J , . I .

T , :

- ,
- ,
- .

A . E - `git receive-pack` .

T *forced* . T , " - " .

I .. . H , , - , . T *post-receive* .

I ' , . S - (1). .

B `git send-pack` , `echo` .

T *update* , `hooks.allowunannotated` .

proc-receive

T - - (1). I - `receive.procReceiveRefs`, *receive-pack* , , `execute_commands()` . T *receive-pack.*

T . I , - *receive-pack* , - . I , *S* *receive-pack* *H* .

```
# Version and features negotiation.
S: PKT-LINE(version=1\0push-options atomic...)
S: flush-pkt
H: PKT-LINE(version=1\0push-options...)
H: flush-pkt
```

```
# Send commands from server to the hook.
S: PKT-LINE(<old-oid> <new-oid> <ref>)
S: ... ...
S: flush-pkt
# Send push-options only if the 'push-options' feature is enabled.
S: PKT-LINE(push-option)
S: ... ...
S: flush-pkt
```

```
# Receive result from the hook.
# OK, run this command successfully.
H: PKT-LINE(ok <ref>)
# NO, I reject it.
H: PKT-LINE(ng <ref> <reason>)
# Fall through, let 'receive-pack' to execute it.
H: PKT-LINE(ok <ref>)
H: PKT-LINE(option fall-through)
# OK, but has an alternate reference.  The alternate reference name
# and other status can be given in option directives.
H: PKT-LINE(ok <ref>)
H: PKT-LINE(option refname <refname>)
H: PKT-LINE(option old-oid <old-oid>)
H: PKT-LINE(option new-oid <new-oid>)
H: PKT-LINE(option forced-update)
H: ... ...
H: flush-pkt
```

E *proc-receive* -
- - , *proc-receive*
- - . F ,
" "
" " .

T . T
proc-receive
, .

post-receive

T - - (1) `git push` ( ) . I .

T . I , *pre-receive* .

T `git receive-pack`, .

T *post-update* .

B `git send-pack` , `echo` .

T *post-receive* , `post-receive-email` `contrib/hooks` G , .

T `git push --push-option=...` `GIT_PUSH_OPTION_COUNT`, `GIT_PUSH_OPTION_0`, `GIT_PUSH_OPTION_1`, I , . I , , , `GIT_PUSH_OPTION_COUNT=0`.

post-update

T - - (1) `git push` ( ) . I .

I , .

T , `git receive-pack`.

T *post-update* , , .. . T *post-receive* . Y .

W , *post-update* `git update-server-info` (. ., HTTP) . I G HTTP, .

B `git send-pack` , `echo` .

reference-transaction

T G . I , .

T , :

- " ": A .
- " ": T .
- " ": T , .

F , :

```
<old-value> SP <new-value> SP <ref-name> LF
```

T " " . I " " , - . T " " .

push-to-checkout

T - - (1) `git push` ( ) , `receive.denyCurrentBranch` `updateInstead`. S ; , . T .

T . I - (,). O , .

F , `git read-tree -u -m HEAD "$1"` `git fetch` `git push`, - `git read-tree -u -m` `git switch` `git checkout` .

pre-auto-gc

T `git gc --auto` (- (1)). I , - `git gc --auto` .

post-rewrite

T (- (1) `--amend` - (1); , - ( ) - - (1) - - !). I : `amend` `rebase`. F - .

T ,

```
<old-sha1> SP <new-sha1> [ SP <extra-info> ] LF
```

T *extra-info* - . I , SP . C , *extra-info.*

T (" . .< >" - (1)) , .

T - :

rebase

F *squash* *fixup* , . T *new-sha1.*

T .

sendemail-validate

T - - (1). I , - . E - `git send-email` - .

fsmonitor-watchman

T `core.fsmonitor` `.git/hooks/fsmonitor-watchman` `.git/hooks/fsmonitor-watchmanv2` .

V 1 , (1) , J 1, 1970.

V 2 , (2) . F . T NUL .

T . T . T NUL.

I OK . A - . W , .

G .

A " " `/`.

T . O , .

p4-changelist

T `git-p4 submit`.

T `p4-changelist` . I `--no-verify` . I , . E - .

T . I S .

R `git-p4 submit --help` .

p4-prepare-changelist

T `git-p4 submit`.

T `p4-prepare-changelist`

. I ,

. E

.

T , `-`

`-no-verify` . T `prepare-p4-only` .

R `git-p4 submit --`

p4-post-cha

T `p4 submit`.

T

T
- . "1" -
"0" .

O "1" . T
"1", "1" .

virtualFilesystem

"V F S " . T
. G

. G .

T . F . I
, (1).

T
. T
NUL. F (*dir1/a.txt*) (*dir1/*).

T . O ,
.