NAME

gitmodules - Defining submodule properties

SYNOPSIS

$GIT_WORK_TREE/.gitmodules

DESCRIPTION

The .gitmodules file, located in the top-level directory of a Git working tree, is a text file with a syntax matching the requirements of git-config(1).

The file contains one subsection per submodule, and the subsection value is the name of the submodule. The name is set to the path where the submodule has been added unless it was customized with the --name option of git submodule add. Each submodule section also contains the following required keys:

submodule.<name>.path
: Defines the path, relative to the top-level directory of the Git working tree, where the submodule is expected to be checked out. The path name must not end with a /. All submodule paths must be unique within the .gitmodules file.

submodule.<name>.url
: Defines a URL from which the submodule repository can be cloned. This may be either an absolute URL ready to be passed to git-clone(1) or (if it begins with ./ or ../) a location relative to the superproject’s origin repository.

In addition, there are a number of optional keys:

submodule.<name>.update
: Defines the default update procedure for the named submodule, i.e. how the submodule is updated by the git submodule update command in the superproject. This is only used by git submodule init to initialize the configuration variable of the same name. Allowed values here are checkout, rebase, merge or none. See description of update command in git-submodule(1) for their meaning. For security reasons, the !command form is not accepted here.

submodule.<name>.branch
: A remote branch name for tracking updates in the upstream submodule. If the option is not specified, it defaults to the remote HEAD. A special value of . is used to indicate that the name of the branch in the submodule should be the same name as the current branch in the current repository. See the --remote documentation in git-submodule(1) for details.

submodule.<name>.fetchRecurseSubmodules
: This option can be used to control recursive fetching of this submodule. If this option is also present in the submodule’s entry in .git/config of the superproject, the setting there will override the one found in .gitmodules. Both settings can be overridden on the command line by using the --[no-]recurse-submodules option to git fetch and git pull.

submodule.<name>.ignore
: Defines under what circumstances git status and the diff family show a submodule as modified. The following values are supported:

    all
    : The submodule will never be considered modified (but will nonetheless show up in the output of status and commit when it has been staged).

    dirty

```
               All changes to the submodule’s work tree will be ignored, only
               committed differences between the HEAD of the submodule and its
               recorded state in the superproject are taken into account.

           untracked
               Only untracked files in submodules will be ignored. Committed
               differences and modifications to tracked files will show up.

           none
               No modifications to submodules are ignored, all of committed
               differences, and modifications to tracked and untracked files
               are shown. This is the default option.

           If this option is also present in the submodule’s entry in
           .git/config of the superproject, the setting there will override
           the one found in .gitmodules.

           Both settings can be overridden on the command line by using the
           --ignore-submodules option. The git submodule commands are not
           affected by this setting.

       submodule.<name>.shallow
           When set to true, a clone of this submodule will be performed as a
           shallow clone (with a history depth of 1) unless the user
           explicitly asks for a non-shallow clone.

NOTES
       Git does not allow the .gitmodules file within a working tree to be a
       symbolic link, and will refuse to check out such a tree entry. This
       keeps behavior consistent when the file is accessed from the index or a
       tree versus from the filesystem, and helps Git reliably enforce
       security checks of the file contents.

EXAMPLES
       Consider the following .gitmodules file:

           [submodule "libfoo"]
                   path = include/foo
                   url = git://foo.com/git/lib.git

           [submodule "libbar"]
                   path = include/bar
                   url = git://bar.com/git/lib.git

       This defines two submodules, libfoo and libbar. These are expected to
       be checked out in the paths include/foo and include/bar, and for both
       submodules a URL is specified which can be used for cloning the
       submodules.

SEE ALSO
       git-submodule(1), gitsubmodules(7), git-config(1)

GIT
       Part of the git(1) suite
```