I'm not a robot

Click here to download the latest (2.50.1) x64 version of Git for Windows. This is the most recent maintained build. It was released on 2025-07-08. Other Git for Windows downloads Standalone Installer Git for Windows/x64 Setup. Git for Windows/ARM64 Setup. Portable ("thumbdrive edition") Git for Windows/x64 Portable. Git for Windows/ARM64 Portable. Using winget tool Install winget tool if you don't already have it, then type this command in command prompt or Powershell. winget install --id Git.Git -e --source winget The current source code release is version 2.50.1. If you want the newer version, you can build it from the source code. Now What? Now that you have downloaded Git, it's time to start using it. So, what is Git in a nutshell?This is an important section to absorb, because if you understand what Git is and the fundamentals of how it works, then using Git effectively will probably be much easier for you.As you learn Git, try to clear your mind of the things you may know about other VCSs, such as CVS, Subversion or Perforcedoing so will help you avoid subtle confusion when using the tool.Even though Gits user interface is fairly similar to these other VCSs, Git stores and thinks about information in a very different way, and understanding these differences will help you avoid becoming confused while using it.The major difference between Git and any other VCS (Subversion and friends included) is the way Git thinks about its data.Conceptually, most other systems store information as a list of file-based changes.These other systems (CVS, Subversion, Perforce, and so on) think of the information they store as a set of files and the changes made to each file over time (this is commonly described as delta-based version control).Figure 4. Storing data as changes to a base version of each fileGit doesnt think of or store its data this way.Instead, Git thinks of its data more like a series of snapshots of a miniature filesystem.With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.To be efficient, if files have not changed, Git doesnt store the file again, just a link to the previous identical file it has already stored.Git thinks about its data more like a stream of snapshots.Figure 5. Storing data as snapshots of the project over timeThis is an important distinction between Git and nearly all other VCSs.It makes Git reconsider almost every aspect of version control that most other systems copied from the previous generation.This makes Git more like a mini filesystem with some incredibly powerful tools built on top of it, rather than simply a VCS.Well explore some of the benefits you gain by thinking of your data this way when we cover Git branching in Git Branching.Most operations in Git need only local files and resources to operategenerally no information is needed from another computer on your network.If youre used to a CVCS where most operations have that network latency overhead, this aspect of Git will make you think that the gods of speed have blessed Git with unworldly powers.Because you have the entire history of the project right there on your local disk, most operations seem almost instantaneous.For example, to browse the history of the project, Git doesnt need to go out to the server to get the history and display it for youit simply reads it directly from your local database.This means you see the project history almost instantly.If you want to see the changes introduced between the current version of a file and the file a month ago, Git can look up the file a month ago and do a local difference calculation, instead of having to either ask a remote server to do it or pull an older version of the file from the remote server to do it locally.This also means that there is very little you cant do if youre offline or off VPN.If you get on an airplane or a train and want to do a little work, you can commit happily (to your local copy, remember?) until you get to a network connection to upload.If you go home and cant get your VPN client working properly, you can still work.In many other systems, doing so is either impossible or painful.In Perforce, for example, you cant do much when you arent connected to the server; in Subversion and CVS, you can edit files, but you cant commit changes to your database (because your database is offline).This may not seem like a huge deal, but you may be surprised what a big difference it can make.Everything in Git is checksummed before it is stored and is then referred to by that checksum.This means its impossible to change the contents of any file or directory without Git knowing about it.This functionality is built into Git at the lowest levels and is integral to its philosophy.You cant lose information in transit or get file corruption without Git being able to detect it.The mechanism that Git uses for this checksumming is called a SHA-1 hash.This is a 40-character string composed of hexadecimal characters (09 and af) and calculated based on the contents of a file or directory structure in Git.A SHA-1 hash looks something like this:24b9da6552252987aa493b52f8696cd6d3b00373You will see these hash values all over the place in Git because it uses them so much.In fact, Git stores everything in its database not by file name but by the hash value of its contents.When you do actions in Git, nearly all of them only add data to the Git database.It is hard to get the system to do anything that is not undoable or to make it erase data in any way.As with any VCS, you can lose or mess up changes you havent committed yet, but after you commit a snapshot into Git, it is very difficult to lose, especially if you regularly push your database to another repository.This makes using Git a joy because we know we can experiment without the danger of severely screwing things up.For a more in-depth look at how Git stores its data and how you can recover data that seems lost, see Undoing Things.Pay attention nowhere is the main thing to remember about Git if you want the rest of your learning process to go smoothly.Git has three main states that your files can reside in: modified, staged, and committed:Modified means that you have changed the file but have not committed it to your database yet.Staged means that you have marked a modified file in its current version to go into your next commit snapshot.Committed means that the data is safely stored in your local database.This leads us to the three main sections of a Git project: the working tree, the staging area, and the Git directory.Figure 6. Working tree, staging area, and Git directoryThe working tree is a single checkout of one version of the project.These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit.Its technical name in Git parlance is the index, but the phrase staging area works just as well.The Git directory is where Git stores the metadata and object database for your project.This is the most important part of Git, and it is what is copied when you clone a repository from another computer.The basic Git workflow goes something like this:You modify files in your working tree.You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.If a particular version of a file is in the Git directory, its considered committed.If it has been modified and was added to the staging area, it is staged.And if it was changed since it was checked out but has not been staged, it is modified.In Git Basics, youll learn more about these states and how you can either take advantage of them or skip the staged part entirely. Git comes with built-in GUI tools (git-gui, gitk), but there are several third-party tools for users looking for a platform-specific experience. View GUI Clients Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects. View Logos If you already have Git installed, you can get the latest development version via Git itself: git clone You can also always browse the current contents of the git repository using the web interface. If --list is given, or if there are no non-option arguments, existingbranches are listed; the current branch will be highlighted in green andmarked with an asterisk. Any branches checked out in linked worktrees willbe highlighted in cyan and marked with a plus sign. Option -r causes theremote-tracking branches to be listed,and option -a shows both local and remote branches.If a is given, it is used as a shell wildcard to restrict the output tomatching branches. If multiple patterns are given, a branch is shown ifit matches any of the patterns.Note that when providing a, you must use --list; otherwise the command may be interpretedas branch creation.With --contains, shows only the branches that contain the named commit(in other words, the branches whose tip commits are descendants of thenamed commit), --no-contains inverts it. With --merged, only branchesmerged into the named commit (i.e. the branches whose tip commits arereachable from the named commit) will be listed. With --no-merged onlybranches not merged into the named commit will be listed. If the argument is missing it defaults to HEAD (i.e. the tip of the currentbranch).The commands second form creates a new branch head named which points to the current HEAD, or if given. As aspecial case, for , you may use ... as ashortcut for the merge base of and if there is exactlyone merge base. You can leave out at most one of and ,in which case it defaults to HEAD.Note that this will create the new branch, but it will not switch theworking tree to it; use git switch to switch to thenew branch.When a local branch is started off a remote-tracking branch, Git sets up thebranch (specifically the branch..remote and branch..mergeconfiguration entries) so that git pull will appropriately merge fromthe remote-tracking branch. This behavior may be changed via the globalbranch.autoSetupMerge configuration flag. That setting can beoverridden by using the --track and --no-track options, andchanged later using git branch --set-upstream-to.With a -m or -M option, will be renamed to .If had a corresponding reflog, it is renamed to match, and a reflog entry is created to remember the branchrenaming. If exists, -M must be used to force the renameto happen.The -c and -C options have the exact same semantics as -m and-M, except instead of the branch being renamed, it will be copied to anew name, along with its config and reflog.With a -d or -D option, will be deleted. You mayspecify more than one branch for deletion. If the branch currentlyhas a reflog then the reflog will also be deleted.Use -r together with -d to delete remote-tracking branches. Note, that itonly makes sense to delete remote-tracking branches if they no longer existin the remote repository or if git fetch was configured not to fetchthem again. See also the prune subcommand of git-remote[1] for away to clean up all obsolete remote-tracking branches. Git Note Git 2.8.0 Git Git 2.8 Linux Git Fedora RPM RHEL CentOS dnf$ sudo dnf install git-all Debian Ubuntu apt$ sudo apt install git-all Mac Git Xcode Command Line ToolsMavericks 10.9 Terminal 'git' Figure 7. Git macOS Installer. GitHub for macOS Git GitHub for macOS Git Git Git autotoolscurlzlibopensslexpat libiconv dnf Fedora apt Debian Git $ sudo dnf install dh-autoreconf curl-devel expat-devel gettext-devel \ openssl-devel perl-devel zlib-devel$ sudo apt-get install dh-autoreconf libcurl4-gnutls-dev libexpat1-dev \ gettext libz-dev libssl-devdochtmlinfo$ sudo dnf install asciidoc xmlto docbook2X$ sudo apt-get install asciidoc xmlto docbook2xNote RHEL RHEL CentOS Scientific Linux EPEL docbook2X Debian Debian/Ubuntu/Ubuntu-derivatives install-info $ sudo apt-get install install-info RPM Fedora/RHEL/RHEL getopt Debian $ sudo dnf install getopt Fedora/RHEL/RHEL$ sudo ln -s /usr/bin/db2x_docbook2texi /usr/bin/docbook2x-texi$ tar -zxf git-2.8.0.tar.gz$ cd git-2.8.0$ make configure$ ./configure --prefix=/usr$ make all doc info$ sudo make install install-doc install-html install-info Git Git $ git clone git://git.kernel.org/pub/scm/git/git.git The official and comprehensive man pages that are included in the Git package itself. Quick reference guides: GitHub Cheat Sheet | Visual Git Cheat Sheet Book Videos Length: 08:15 Length: 05:06 Length: 05:59 Length: 04:26 See all videos External Links The External Links section is a curated, ever-evolving collection of tutorials, books, videos, and other Git resources. 2nd Edition (2014) The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versions of the book are available on Amazon.com. The version found here has been updated with corrections and additions from hundreds of contributors. If you see an error or have a suggestion, patches and issues are welcome in its GitHub repository. GIT_MERGE_VERBOSITY A number controlling the amount of output shown bythe recursive merge strategy. Overrides merge.verbosity.See git-merge[1]GIT_PAGER This environment variable overrides $PAGER. If it is setto an empty string or to the value "cat", Git will not launcha pager. See also the core.pager option ingit-config[1].GIT_PROGRESS_DELAY A number controlling how many seconds to delay before showingoptional progress indicators. Defaults to 2.GIT_EDITOR This environment variable overrides $EDITOR and $VISUAL.It is used by several Git commands when, on interactive mode,an editor is to be launched. See also git-var[1]and the core.editor option in git-config[1].GIT_SEQUENCE_EDITOR This environment variable overrides the configured Git editorwhen editing the todo list of an interactive rebase. See alsogit-rebase[1] and the sequence.editor option ingit-config[1].GIT_SSH GIT_SSH_COMMAND If either of these environment variables is set then git fetchand git push will use the specified command instead of sshwhen they need to connect to a remote system.The command-line parameters passed to the configured command aredetermined by the ssh variant. See ssh.variant option ingit-config[1] for details.$GIT_SSH_COMMAND takes precedence over $GIT_SSH, and is interpretedby the shell, which allows additional arguments to be included.$GIT_SSH on the other hand must be just the path to a program(which can be a wrapper shell script, if additional arguments areneeded).Usually it is easier to configure any desired options through yourpersonal .ssh/config file. Please consult your ssh documentationfor further details.GIT_SSH_VARIANT If this environment variable is set, it overrides Gits autodetectionwhether GIT_SSH/GIT_SSH_COMMAND/core.sshCommand refer to OpenSSH,plink or tortoiseplink. This variable overrides the config settingssh.variant that serves the same purpose.GIT_SSL_NO_VERIFY Setting and exporting this environment variable to any valuetells Git not to verify the SSL certificate when fetching orpushing over HTTPS.GIT_ATTR_SOURCE Sets the treeish that gitattributes will be read from.GIT_ASKPASS If this environment variable is set, then Git commands which need toacquire passwords or passphrases (e.g. for HTTP or IMAP authentication)will call this program with a suitable prompt as command-line argumentand read the password from its STDOUT. See also the core.askPassoption in git-config[1].GIT_TERMINAL_PROMPT If this Boolean environment variable is set to false, git will not prompton the terminal (e.g., when asking for HTTP authentication).GIT_CONFIG_GLOBAL GIT_CONFIG_SYSTEM Take the configuration from the given files instead from global orsystem-level configuration files. If GIT_CONFIG_SYSTEM is set, thesystem config file defined at build time (usually /etc/gitconfig)will not be read. Likewise, if GIT_CONFIG_GLOBAL is set, neither$HOME/.gitconfig nor $XDG_CONFIG_HOME/git/config will be read. Canbe set to /dev/null to skip reading configuration files of therespective level.GIT_CONFIG_NOSYSTEM Whether to skip reading settings from the system-wide$(prefix)/etc/gitconfig file. This Boolean environment variable canbe used along with $HOME and $XDG_CONFIG_HOME to create apredictable environment for a picky script, or you can set itto true to temporarily avoid using a buggy /etc/gitconfig file whilewaiting for someone with sufficient permissions to fix it.GIT_FLUSH If this Boolean environment variable is set to true, then commands suchas git blame (in incremental mode), git rev-list, git log,git check-attr and git check-ignore willforce a flush of the output stream after each record have beenflushed. If thisvariable is set to false, the output of these commands will be doneusing completely buffered I/O. If this environment variable isnot set, Git will choose buffered or record-oriented flushingbased on whether stdout appears to be redirected to a file or not.GIT_TRACE Enables general trace messages, e.g. alias expansion, built-incommand execution and external command execution.If this variable is set to "1", "2" or "true" (comparisonis case insensitive), trace messages will be printed tostderr.If the variable is set to an integer value greater than 2and lower than 10 (strictly) then Git will interpret thisvalue as an open file descriptor and will try to write thetrace messages into this file descriptor.Alternatively, if the variable is set to an absolute path(starting with a / character), Git will interpret thisas a file path and will try to append the trace messagesto it.Unsetting the variable, or setting it to empty, "0" or"false" (case insensitive) disables trace messages.GIT_TRACE_FSMONITOR Enables trace messages for the filesystem monitor extension.See GIT_TRACE for available trace output options.GIT_TRACE_PACK_ACCESS Enables trace messages for all accesses to any packs. For eachaccess, the pack file name and an offset in the pack isrecorded. This may be helpful for troubleshooting somepack-related performance problems.See GIT_TRACE for available trace output options.GIT_TRACE_PACKET Enables trace messages for all packets coming in or out of agiven program. This can help with debugging object negotiationor other protocol issues. Tracing is turned off at a packetstarting with "PACK" (but see GIT_TRACE_PACKFILE below).See GIT_TRACE for available trace output options.GIT_TRACE_PACKFILE Enables tracing of packfiles sent or received by agiven program. Unlike other trace output, this trace isverbatim: no headers, and no quoting of binary data. You almostcertainly want to direct into a file (e.g.,GIT_TRACE_PACKFILE=/tmp/my.pack) rather than displaying it onthe terminal or mixing it with other trace output.Note that this is currently only implemented for the client sideof clones and fetches.GIT_TRACE_PERFORMANCE Enables performance related trace messages, e.g. total executiontime of each Git command.See GIT_TRACE for available trace output options.GIT_TRACE_REFS Enables trace messages for operations on the ref database.See GIT_TRACE for available trace output options.GIT_TRACE_SETUP Enables trace messages printing the .git, working tree and currentworking directory after Git has completed its setup phase.See GIT_TRACE for available trace output options.GIT_TRACE_SHALLOW Enables trace messages that can help debugging fetching /cloning of shallow repositories.See GIT_TRACE for available trace output options.GIT_TRACE_CURL Enables a curl full trace dump of all incoming and outgoing data,including descriptive information, of the git transport protocol.This is similar to doing curl --trace-ascii on the command line.See GIT_TRACE for available trace output options.GIT_TRACE_CURL_NO_DATA When a curl trace is enabled (see GIT_TRACE_CURL above), do not dumpdata (that is, only dump info lines and headers).GIT_TRACE2 Enables more detailed trace messages from the "trace2" library.Output from GIT_TRACE2 is a simple text-based format for humanreadability.If this variable is set to "1", "2" or "true" (comparisonis case insensitive), trace messages will be printed tostderr.If the variable is set to an integer value greater than 2and lower than 10 (strictly) then Git will interpret thisvalue as an open file descriptor and will try to write thetrace messages into this file descriptor.Alternatively, if the variable is set to an absolute path(starting with a / character), Git will interpret thisas a file path and will try to append the trace messagesto it. If the path already exists and is a directory, thetrace messages will be written to files (one per process)in that directory, named according to the last componentof the SID and an optional counter (to avoid filenamecollisions).In addition, if the variable is set toaf_unix:[:], Git will tryto open the path as a Unix Domain Socket. The socket typecan be either stream or dgram.Unsetting the variable, or setting it to empty, "0" or"false" (case insensitive) disables trace messages.See Trace2 documentationfor full details.GIT_TRACE2_EVENT This setting writes a JSON-based format that is suited for machineinterpretation.See GIT_TRACE2 for available trace output options andTrace2 documentation for full details.GIT_TRACE2_PERF In addition to the text-based messages available in GIT_TRACE2, thissetting writes a column-based format for understanding nestingregions.See GIT_TRACE2 for available trace output options andTrace2 documentation for full details.GIT_TRACE_REDACT By default, when tracing is activated, Git redacts the values ofcookies, the "Authorization:" header, the "Proxy-Authorization:"header and packfile URIs. Set this Boolean environment variable to false to prevent thisredaction.GIT_NO_REPLACE_OBJECTS Setting and exporting this environment variable tells Git toignore replacement refs and do not replace Git objects.GIT_LITERAL_PATHSPECS Setting this Boolean environment variable to true will cause Git to treat allpathspecs literally, rather than as glob patterns. For example,running GIT_LITERAL_PATHSPECS=1 git log -- *.c' will searchfor commits that touch the path *.c, not any paths that theglob *.c matches. You might want this if you are feedingliteral paths to Git (e.g., paths previously given to you bygit ls-tree, --raw diff output, etc).GIT_GLOB_PATHSPECS Setting this Boolean environment variable to true will cause Git to treat allpathspecs as glob patterns (aka "glob" magic).GIT_NOGLOB_PATHSPECS Setting this Boolean environment variable to true will cause Git to treat allpathspecs as literal (aka "literal" magic).GIT_ICASE_PATHSPECS Setting this Boolean environment variable to true will cause Git to treat allpathspecs as case-insensitive.GIT_NO_LAZY_FETCH Setting this Boolean environment variable to true tells Gitnot to lazily fetch missing objects from the promisor remoteon demand.GIT_REFLOG_ACTION When a ref is updated, reflog entries are created to keeptrack of the reason why the ref was updated (which istypically the name of the high-level command that updatedthe ref), in addition to the old and new values of the ref.A scripted Porcelain command can use set_reflog_actionhelper function in git-sh-setup to set its name to thisvariable when it is invoked as the top level command by theend user, to be recorded in the body of the reflog.GIT_REF_PARANOIA If this Boolean environment variable is set to false, ignore broken or badly named refs when iteratingover lists of refs. Normally Git will try to include any suchrefs, which may cause some operations to fail. This is usuallypreferable, as potentially destructive operations (e.g.,git-prune[1]) are better off aborting rather thanignoring broken refs (and thus considering the history theypoint to as not worth saving). The default value is 1 (i.e.,be paranoid about detecting and aborting all operations). Youshould not normally need to set this to 0, but it may beuseful when trying to salvage data from a corrupted repository.GIT_COMMIT_GRAPH_PARANOIA When loading a commit object from the commit-graph, Git performs anexistence check on the object in the object database. This is done toavoid issues with stale commit-graphs that contain references toalready-deleted commits, but comes with a performance penalty.The default is "false", which disables the aforementioned behavior.Setting this to "true" enables the existence check so that stale commitswill never be returned from the commit-graph at the cost of performance.GIT_ALLOW_PROTOCOL If set to a colon-separated list of protocols, behave as ifprotocol.allow is set to never, and each of the listedprotocols has protocol..allow set to always(overriding any existing configuration). See the description ofprotocol.allow in git-config[1] for more details.GIT_PROTOCOL_FROM_USER Set this Boolean environment variable to false to prevent protocols used by fetch/push/clone which areconfigured to the user state. This is useful to restrict recursivesubmodule initialization from an untrusted repository or for programswhich feed potentially-untrusted URLS to git commands. Seegit-config[1] for more details.GIT_PROTOCOL For internal use only. Used in handshaking the wire protocol.Contains a colon : separated list of keys with optional values[=]. Presence of unknown keys and values must beignored.Note that servers may need to be configured to allow this variable topass over some transports. It will be propagated automatically whenaccessing local repositories (i.e., file:// or a filesystem path), aswell as over the git:// protocol. For git-over-http, it should workautomatically in most configurations, but see the discussion ingit-http-backend[1]. For git-over-ssh, the ssh server may needto be configured to allow clients to pass this variable (e.g., by usingAcceptEnv GIT_PROTOCOL with OpenSSH).This configuration is optional. If the variable is not propagated, thenclients will fall back to the original "v0" protocol (but may miss auton some performance improvements or features). This variable currentlyonly affects clones and fetches; it is not yet used for pushes (but maybe in the future).GIT_OPTIONAL_LOCKS If this Boolean environment variable is set to false, Git will complete any requested operation withoutperforming any optional sub-operations that require taking a lock.For example, this will prevent git status from refreshing theindex as a side effect. This is useful for processes running inthe background which do not want to cause lock contention withother operations on the repository. Defaults to 1.GIT_REDIRECT_STDIN GIT_REDIRECT_STDOUT GIT_REDIRECT_STDERR Windows-only: allow redirecting the standard input/output/errorhandles to paths specified by the environment variables. This isparticularly useful in multi-threaded applications where thecanonical way to pass standard handles via CreateProcess() isnot an option because it would require the handles to be markedinheritable (and consequently every spawned process wouldinherit them, possibly blocking regular Git operations). Theprimary intended use case is to use named pipes for communication(e.g. \\.\pipe\my-git-stdin-123).Two special values are supported: off will simply close thecorresponding standard handle, and if GIT_REDIRECT_STDERR is2>&1, standard error will be redirected to the same handle asstandard output.GIT_PRINT_SHA1_ELLIPSIS (deprecated) If set to yes, print an ellipsis following an(abbreviated) SHA-1 value. This affects indications ofdetached HEADs (git-checkout[1]) and the rawdiff output (git-diff[1]). Printing anellipsis in the cases mentioned is no longer consideredadequate and support for it is likely to be removed in theforeseeable future (along with the variable).GIT_ADVICE If set to 0, then disable all advice messages. These messages areintended to provide hints to human users that may help them get out ofproblematic situations or take advantage of new features. Users candisable individual messages using the advice.* config keys. Thesemessages may be disruptive to tools that execute Git processes, so thisvariable is available to disable the messages. (The --no-adviceglobal option is also available, but old Git versions may fail whenthis option is not understood. The environment variable will be ignoredby Git versions that do not understand it.) git [-v | --version] [-h | --help] [-C] [-c =] [--exec-path[=]] [--html-path] [--man-path] [--info-path] [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch] [--no-optional-locks] [--no-advice] [--bare] [--git-dir=] [--work-tree=] [--namespace=] [--config-env==] [] -v --version git Git git version ... git-version[1] --version --help --version -h --help --all -a git --help ... git help ... git-help[1] -C git -C -C -C -C "" --git-dir --work-tree -C git --git-dir=a.git --work-tree=b -C c statusgit --git-dir=c/a.git --work-tree=c/b status -c = git config . git -c foo.bar ... = foo.bar true [foo]bar git -c foo.bar= ... foo.bar git config --type=bool false --config-env== -c = -c Git /proc/self/cmdline /proc/self/environ Linux http.extraHeader url.< URL >.insteadOf --exec-path[=] Git GIT_EXEC_PATH git --html-path Git HTML --man-path Git man page manpath man(1) --info-path Git -p --paginate lessgit $PAGER pager. -P --no-pager Git --git-dir= ".git" GIT_DIR GIT_DIR ".git" ".git" Git --work-tree= GIT_WORK_TREE Git Git git -C --work-tree= GIT_WORK_TREE core.worktree git-config[1] core.worktree --namespace= Git gitnamespaces[7] GIT_NAMESPACE --bare GIT_DIR --no-replace-objects Do not use replacement refs to replace Git objects. This is equivalent to exporting the GIT_NO_REPLACE_OBJECTS environment variable with any value. See git-replace[1] for more information. --no-lazy-fetch git cat-file -e GIT_NO_LAZY_FETCH 1 --no-optional-locks GIT_OPTIONAL_LOCKS 0 --no-advice --literal-pathspecs GIT_LITERAL_PATHSPECS 1 --glob-pathspecs "glob" GIT_GLOB_PATHSPECS 1 ":(literal)" "glob" --noglob-pathspecs "literal" GIT_NOGLOB_PATHSPECS 1 ":(glob)" --icase-pathspecs "icase" GIT_ICASE_PATHSPECS 1 --list-cmds=[,] /buildinsparseopt parse-options mainlibexec others $PATH git- list- command-list.txt Git nohelpersalias config completion.commands --attr-source= gitattributes gitattributes[5] GIT_ATTR_SOURCE Git WarningMissing zh_HANS-CN/{build_dir}/cmds-mainporcelain.adocSee original version for this content.WarningMissing zh_HANS-CN/{build_dir}/cmds-ancillarymanipulators.adocSee original version for this content.WarningMissing zh_HANS-CN/{build_dir}/cmds-ancillaryinterrogators.adocSee original version for this content. SCMWarningMissing zh_HANS-CN/{build_dir}/cmds-foreignscminterface.adocSee original version for this content.git resetgit restore git revertgit-revert[1] git-restore[1] git-reset[1] git reset git restore WarningMissing zh_HANS-CN/{build_dir}/cmds-plumbingmanipulators.adocSee original version for this content.WarningMissing zh_HANS-CN/{build_dir}/cmds-plumbinginterrogators.adocSee original version for this content.WarningMissing zh_HANS-CN/{build_dir}/cmds-synchingrepositories.adocSee original version for this content.WarningMissing zh_HANS-CN/{build_dir}/cmds-synchelpers.adocSee original version for this content.WarningMissing zh_HANS-CN/{build_dir}/cmds-purehelpers.adocSee original version for this content. WarningMissing zh_HANS-CN/{build_dir}/cmds-guide.adocSee original version for this content. git-help[1] --user-formatsWarningMissing zh_HANS-CN/{build_dir}/cmds-userinterfaces.adocSee original version for this content. git git-help[1] --developer-interfaces WarningMissing zh_HANS-CN/{build_dir}/cmds-developerinterfaces.adocSee original version for this content.Git ## '#' ';' #; [core]; filemode = false; [user]name = "Junio C Hamano"email = "gitster@pobox.com" git-config[1] blobtreecommit tag GIT_INDEX_FILE Git HEAD refs/tags/ refs/heads/ gitrevisions[7] Various Git commands pay attention to environment variables and change their behavior. The environment variables marked as "Boolean" take their values the same way as Boolean valued configuration variables, i.e., "true", "yes", "on" and positive numbers are taken as "yes", while "false", "no", "off", and "0" are taken as "no". HOME Specifies the path to the users home directory. On Windows, if unset, Git will set a process environment variable equal to: $HOMEDRIVE$HOMEPATH if both $HOMEDRIVE and $HOMEPATH exist; otherwise $USERPROFILE if $USERPROFILE exists. Git Git SCMS Git GIT_INDEX_FILE $GIT_DIR/index GIT_INDEX_VERSION 2 3 git-update-index[1] GIT_OBJECT_DIRECTORY sha1 $GIT_DIR/objects GIT_ALTERNATE_OBJECT_DIRECTORIES Git ":" Windows ";" Git Git " C "path-with-\"-and-:-in-it":vanilla-path path-with-"-and-:-in-it vanilla-path GIT_DIR GIT_DIR .git --git-dir GIT_WORK_TREE --work-tree core.worktree GIT_NAMESPACE Git gitnamespaces[7] --namespace GIT_CEILING_DIRECTORIES Git GIT_DIR Git Git GIT_CEILING_DIRECTORIES=/maybe/symlink::/very/slow/non/symlink GIT_DISCOVERY_ACROSS_FILESYSTEM ".git " Git "true" Git GIT_CEILING_DIRECTORIES GIT_DIR GIT_COMMON_DIR $GIT_DIR HEAD index $GIT_DIR gitrepository-layout[5] git-worktree[1] GIT_INDEX_FILEGIT_OBJECT_DIRECTORY GIT_DEFAULT_HASH "sha1" git-init[1] --object-format GIT_DEFAULT_REF_FORMAT "files" git-init[1] --ref-format GIT_AUTHOR_NAME user.name author.name GIT_AUTHOR_EMAIL user.email author.email GIT_AUTHOR_DATE git-commit[1] GIT_COMMITTER_NAME user.name committer.name GIT_COMMITTER_EMAIL user.email committer.email GIT_COMMITTER_DATE reflogs git-commit[1] EMAIL GIT_DIFF_OPTS "--unified=??" "-u??" Git diff "-U" "--unified" GIT_EXTERNAL_DIFF Git GIT_EXTERNAL_DIFF Git GIT_EXTERNAL_DIFF 7 path old-file old-hex old-mode new-file new-hex new-mode -file GIT_EXTERNAL_DIFF -hex 40 SHA-1 -mode "git-diff-files" new-file/dev/null old-file old-file GIT_EXTERNAL_DIFF GIT_EXTERNAL_DIFF GIT_EXTERNAL_DIFF GIT_EXTERNAL_DIFF GIT_DIFF_PATH_COUNTER GIT_DIFF_PATH_TOTAL GIT_EXTERNAL_DIFF_TRUST_EXIT_CODE true GIT_EXTERNAL_DIFF 0 1 diff(1) false 0 Git GIT_DIFF_PATH_COUNTER 1 1 GIT_DIFF_PATH_TOTAL GIT_MERGE_VERBOSITY merge.verbosity git-merge[1] GIT_PAGER $PAGER "cat"Git git-config[1] core.pager GIT_PROGRESS_DELAY 2 GIT_EDITOR $EDITOR $VISUAL Git git-var[1] git-config[1] core.editor GIT_SEQUENCE_EDITOR Git git-rebase[1] git-config[1] sequence.editor GIT_SSH GIT_SSH_COMMAND git fetch git push ssh ssh git-config[1] ssh.variant $GIT_SSH_COMMAND $GIT_SSH shell $GIT_SSH shell .ssh/config ssh GIT_SSH_VARIANT Git GIT_SSH/GIT_SSH_COMMAND/core.sshCommand OpenSSHplink tortoiseplink ssh.variant GIT_SSL_NO_VERIFY Git HTTPS SSL GIT_ATTR_SOURCE gitattributes GIT_ASKPASS Git HTTP IMAP STDOUT git-config[1] core.askPass GIT_TERMINAL_PROMPT falsegit HTTP GIT_CONFIG_GLOBAL GIT_CONFIG_SYSTEM GIT_CONFIG_SYSTEM /etc/gitconfig GIT_CONFIG_GLOBAL $HOME/.gitconfig $XDG_CONFIG_HOME/git/config /dev/null GIT_CONFIG_NOSYSTEM $(prefix)/etc/gitconfig $HOME $XDG_CONFIG_HOME true /etc/gitconfig GIT_FLUSH true git blamegit rev-listgit loggit check-attr git check-ignore false I/O I/O Git GIT_TRACE "1""2" "true" 2 10 Git / Git "0" "false" GIT_TRACE_FSMONITOR GIT_TRACE GIT_TRACE_PACK_ACCESS GIT_TRACE GIT_TRACE_PACKET "PACK" GIT_TRACE_PACKFILE GIT_TRACE GIT_TRACE_PACKFILE=/tmp/my.pack GIT_TRACE_PERFORMANCE Git GIT_TRACE GIT_TRACE_REFS GIT_TRACE GIT_TRACE_SETUP Git .git GIT_TRACE GIT_TRACE_SHALLOW / GIT_TRACE GIT_TRACE_CURL git curl curl --trace-ascii GIT_TRACE GIT_TRACE_CURL_NO_DATA curl GIT_TRACE_CURL GIT_TRACE2 "trace2" GIT_TRACE2 "1""2" "true" 2 10 Git / Git SID af_unix:[:]Git Unix stream dgram"0" "false" GIT_TRACE2_EVENT JSON GIT_TRACE2 Trace2 GIT_TRACE2_PERF GIT_TRACE2 GIT_TRACE2 Trace2 GIT_TRACE_REDACT Git cookie"Authorization:""Proxy-Authorization:" packfile URI false GIT_NO_REPLACE_OBJECTS Setting and exporting this environment variable tells Git to ignore replacement refs and do not replace Git objects. GIT_LITERAL_PATHSPECS true Git GIT_LITERAL_PATHSPECS=1 git log -- '*.c' *.c *.c Git git ls-tree--raw GIT_GLOB_PATHSPECS true Git "glob" GIT_NOGLOB_PATHSPECS trueGit "literal" GIT_ICASE_PATHSPECS true Git GIT_NO_LAZY_FETCH Setting this Boolean environment variable to true tells Git not to lazily fetch missing objects from the promisor remote on demand. GIT_REFLOG_ACTION git-sh-setup set_reflog_action GIT_REF_PARANOIA falseGit git-prune[1] 1 0 GIT_COMMIT_GRAPH_PARANOIA Git "false" "true" GIT_ALLOW_PROTOCOL protocol.allow never protocol..allow always git-config[1] protocol.allow GIT_PROTOCOL_FROM_USER false fetch/push/clone user git URLS git-config[1] GIT_PROTOCOL For internal use only. Used in handshaking the wire protocol. Contains a colon : separated list of keys with optional values [=]. Presence of unknown keys and values must be ignored. file:// git:// git-over-http git-http-backend[1] git-over-ssh ssh OpenSSH AcceptEnv GIT_PROTOCOL "v0" GIT_OPTIONAL_LOCKS falseGit git status 1 GIT_REDIRECT_STDIN GIT_REDIRECT_STDOUT GIT_REDIRECT_STDERR Windows// CreateProcess() Git \\.\pipe\my-git-stdin-123 GIT_REDIRECT_STDERR 2>&1 GIT_PRINT_SHA1_ELLIPSIS yes SHA-1 HEAD git-checkout[1]git-diff[1] GIT_ADVICE 0 advice.* Git (--no-advice Git Git Git gitcore-tutorial[7]Git ".git " .git "index" blobstree blobs commits "changeset" "version" SHA-1 40 tag "pack files" SHA-1 refs/head/ "head" SHA-1 SHA-1 refs/tags/ HEAD "stages" Git shell git clone git log .git .git Git Git git-config[1] safe.directory zip tarball .git git clone --no-local Git upload-pack upload-pack upload-pack git-daemon[1]ssh ID git-upload-pack[1]

Most fun places in florida for young adults. Fun places in florida for adults. Fun place florida. Fun places to stay florida. Fun resorts for young adults. Fun things to do in florida for young adults. Fun in florida. Affordable places to live in florida for young adults. Best places to go in florida for young adults.

- http://iksanhp.com/upload_files/file/13558273235.pdf
- pediatric nurse practitioner job descriptions for resume
- geminoraci
- fada
- https://www.ogsb.org/asset/ckeditor/kcfinder/upload/files/58391d8b-eabd-4e40-8b66-c1c40ba90711.pdf
- hutuvebi
- https://stalwart-inst.com/alpha/ckfinder/userfiles/files/26393862059.pdf
- http://waypl.pl/upload/File/926750fb-8314-4958-a39f-88dbc5fb8f3e.pdf