Git

TeamCity supports Git out of the box. Git source control with Visual Studio Team Services is supported (see authentication notes below).

This page contains description of the Git-specific fields of the VCS root settings.
For common VCS Root properties, see this section.

> Git command line client needs to be installed on the agents if the agent-side checkout is used.

> Important Notes
> - Remote Run and Pre-Tested Commit are supported in the IntelliJ IDEA and Eclipse plugins; with the Visual Studio Addin use the Branch Remote Run Trigger.
> - Initial Git checkout may take significant time (sometimes hours), depending on the size of your project history, because the whole project history is downloaded during the initial checkout.

On this page:

General Settings

Option	Description
Fetch URL	The URL of the remote Git repository used for fetching data from the repository.
Push URL	The URL of the target remote Git repository used for pushing annotated tags created via VCS labeling build feature to the remote repository. If blank, the fetch URL is used.
Default branch	Configures default branch. Parameter references are supported here. Default value is `refs/heads/master` > You can configure Git-plugin to fetch all heads by adding a build configuration parameter `teamcity.git.fetchAllHeads=true`
Branch specification	Lists the patterns for branch names, required for feature branches support. The matched branches are monitored for changes in addition to the default branch. The syntax is similar to checkout rules: `+|-:branch_name`, where `branch_name` is specific to the VCS, i.e. `refs/heads/` in Git (with the optional * placeholder).
Use tags as branches	Allows monitoring / checking out git tags as branches making branch specification match tag names as well as branches (e.g. `+|-:refs/tags/<tag_name>`). By default, tags are ignored.

Username style	Defines a way TeamCity reports username for a VCS change. The username is reported based on the Author field of the Git commit and can include name, email or their combinations. Changing the username style will affect only newly collected changes. Old changes will continue to be stored with the style that was active at the time of collecting changes.
Submodules	Select whether you want to ignore the submodules, or treat them as a part of the source tree. Submodule repositories should either not require authentication or use the same protocol and accept the same authentication as configured in the VCS root.
Username for tags/merge	A custom username used for labeling.

Branch Matching Rules

- If the branch matches a line without patterns, the line is used.
- If the branch matches several lines with patterns, the best matching line is used.
- If there are several lines with equal matching, the one below takes precedence.

Everything that is matched by the wildcard will be shown as a branch name in the TeamCity interface. For example, `+:refs/heads/*` will match `refs/heads/feature1` branch, but in the TeamCity interface you'll see only `feature1` as a branch name.
The short name of the branch is determined as follows:

- if the line contains no brackets, then full line is used, if there are no patterns or part of line starting with the first pattern-matched character to the last pattern-matched character.
- if the line contains brackets, then part of the line within brackets is used.

When branches are specified here, and if your build configuration has a VCS trigger and a change is found in some branch, TeamCity will trigger a build in this branch.

Supported Git Protocols

The following protocols are supported for Git repository URL:

- ssh: (e.g. ssh://git.somwhere.org/repos/test.git, ssh://git@git.somwhereElse.org/repos/test.git, scp-like syntax: git@git.somwhere.org:repos/test.git)

  > **Be Careful**
  > The scp-like syntax requires a colon after the hostname, while the usual ssh url does not. This is a common source of errors.

- git: (e.g. git://git.kernel.org/pub/scm/git/git.git)
- http: (e.g. http://git.somewhere.org/projects/test.git)
- file: (e.g. file:///c:/projects/myproject/.git)

  > **Be Careful**
  > When you run TeamCity as a Windows service, it cannot access mapped network drives and repositories located on them.

Authentication Settings

Authentication Method	Description
Anonymous	Select this option to clone a repository with anonymous read access.
Password	Specify a valid username (if there is no username in the clone URL; the username specified here overrides the username from the URL) and a password to be used to clone the repository. For the agent-side checkout, it is supported only if git 1.7.3+ client is installed on the agent. See TW-18711 . For Git hosted from Team Foundation Server 2013, specify NTLM credentials here.

Private Key	Valid only for SSH protocol. A private key must be in the OpenSSH format. Select one of the options from the Private Key list and specify a valid username (if there is no username in the clone URL; the username specified here overrides the username from the URL). Available Private Key options: • Uploaded Key: uses the key(s) uploaded to the project. See SSH Keys Management for details. • Default Private key - Uses the keys available on the file system in the default locations used by common ssh tools: the mapping specified in `<USER_HOME>/.ssh/config` if the file exists or the private key file `<USER_HOME>/.ssh/id_rsa` (the files are required to be present on the server and also on the agent if the agent-side checkout is used). • Custom Private Key - Supported only for server-side checkout, see TW-18449. When this method is used, specify an absolute path to the private key in the Private Key Path field. If required, specify the passphrase to access your SSH key in the corresponding field.

For all the available options to connect to GitHub, please see the comment.

Authenticating to Visual Studio Team Services

If you use Git source control with Visual Studio Team Services, the following options are available to you:

Personal Access Tokens

To use access tokens, you need to create a personal access token in your Azure DevOps account, where you have to set some Code access scope in your repositories and use it when configuring a VCS root.

Option	Description
Username	Leave blank for TFVC, any value for Git, e.g. username
Password	Enter your personal access token created earlier

Required Access Scope

TFS subsystem	Scopes
TFVC	All scopes
Git	Code (read) / Code (read and write) for versioned settings
Work Items	Work items (read)
Commit Status	Code (status)

Alternate Authentication Credentials

To use the login/password pair authentication, you have to enable alternate credentials in your Azure DevOps account, where you can set a secondary username and password to use when configuring a VCS root.

Server Settings

These are the settings used in case of the server-side checkout.

Option	Description
Convert line-endings to CRLF	Convert line-endings of all text files to CRLF (works as setting `core.autocrlf=true` in a repository config). When not selected, no line-endings conversion is performed (works as setting `core.autocrlf=false`). Affects the server-side checkout only. A change to this property causes a clean checkout.
Custom clone directory on server	To interact with the remote git repository, the its bare clone is created on the TeamCity server machine. By default, the cloned repository is placed under `<TeamCity Data Directory>/system/caches/git` and <TeamCity Data Directory>`/system/caches/git/map`. The field specifies the mapping between repository url and its directory on the TeamCity server. Leave this field blank to use the default location.

Agent Settings

These are the settings used in case of the agent-side checkout.
Note that the agent-side checkout has limited support for SSH. The only supported authentication methods are "Default Private Key" and "Uploaded Private Key" .
If you plan to use the agent-side checkout, you need to have Git 1.6.4+ installed on the agents.

Option	Description
Path to git	Provide the path to a git executable to be used on the agent. When set to `%env.TEAMCITY_GIT_PATH%`, the automatically detected git will be used, see Git executable on the agent for details
Clean Policy/Clean Files Policy	Specify here when the "git clean" command is to run on the agent, and which files are to be removed.
Use mirrors	When enabled (default), TeamCity clones the repository under the agent's `system\git` directory and uses the mirror as an alternate repository when updating the checkout directory for the build. As a result, this speeds-up clean checkout (because only the working directory is cleaned), and saves disk space (as there is only one clone of the given git repository on an agent).


To configure a connection from a TeamCity server running behind a proxy to a remote Git repository, see this section.

Git executable on the agent

TeamCity needs Git command line client version 1.6.4+ on the agent in order to use the agent-side checkout.

The recommended approach is to ensure that the git client is available in PATH of the TeamCity agent and leave the "Path to git" setting in the VCS root blank.
If you only have the git command line on some machines, set "Path to git" setting in the VCS root to the %env.`TEAMCITY_GIT_PATH`% value.

Instead of adding Git to the agent's PATH, you can set the TEAMCITY_GIT_PATH environment variable (or `env.TEAMCITY_GIT_PATH` property in the agent's buildAgent.properties file) to the full path to the git executable.

If `TEAMCITY_GIT_PATH` is not defined, the Git agent plugin tries to detect the installed git on the launch of the agent. It first tries to run git from the following locations:

- for Windows - it tries to run `git.exe` at:
  - C:\Program Files\Git\bin
  - C:\Program Files (x86)\Git\bin
  - C:\cygwin\bin
- for *nix - it tries to run `git` at:
  - /usr/local/bin
  - /usr/bin
  - /opt/local/bin
  - /opt/bin

If git is not found in any of these locations, it tries to run the git accessible via the `PATH` environment variable.
If a compatible git (1.6.4+) is found, it is reported in the `TEAMCITY_GIT_PATH` environment variable. This variable can be used in the Path to git field in the VCS root settings. As a result, the configuration with such a VCS root will run only on the agents where git was detected or specified in the agent properties.

Configuring Git Garbage Collection on Server

TeamCity server maintains a clone for every Git repository it works with, so the process which collects changes in the large Git repository may cause memory problems on the TeamCity server if the Git garbage collection for the repository was not run for a long time.
TeamCity can automatically run git gc periodically when native Git client can be found on the server.

When TeamCity runs Git garbage collection, the details are logged into the teamcity-cleanup.log. If git garbage collection fails,

a corresponding warning is displayed. To fix the warning / meet automatic git gc requirements, perform the following:

1. Install a native Git client manually on the TeamCity server.
2. Specify the directory to the Git executable:
    a. either add it to the `Path` environment variable and restart the server,
    b. or set it in the `teamcity.server.git.executable.path` internal property without the server restart.

TeamCity executes Git garbage collection until the total time doesn't exceed 60 minutes quota; the quota can be changed using the `teamcity.server.git.gc.quota.minutes` internal property.
Git garbage collection is executed every night at 2 a.m., this can be changed by specifying the internal property with a cron expression like this: `"teamcity.git.cleanupCron=0 0 2 * * ?"` (restart the server for the property to take effect)

If git gc process is slow and cannot be finished within allotted time, check git-repack configuration in the default git configuration files. e.g. "--window-memory" can be increased to improve git gc performance.

Git LFS

TeamCity supports Git LFS for agent-side checkout. To use it, install git 1.8.5+ and Git LFS on the build agent machine. Git LFS should be enabled using the 'git lfs install' command (on Windows an elevated command prompt may be needed). More information is available in Git LFS documentation.

Internal Properties

For Git VCS it is possible to configure the following internal properties:

Property	Default	Description
teamcity.git.idle.timeout.seconds	1800	The idle timeout for communication with the remote repository. If no data were sent or received during this timeout, the plugin throws a timeout error to prevent hanging of the process forever.
teamcity.git.fetch.timeout	1800	(deprecated) Override of "teamcity.git.idle.timeout.seconds" for git fetch operation
teamcity.git.fetch.separate.process	true	Defines whether TeamCity runs git fetch in a separate process
teamcity.git.fetch.process.max.memory	512M	The value of the JVM -Xmx parameter for a separate fetch process. You also need to ensure the server machine has enough memory as the memory configured will be used in addition to the main server process and there can be several child processes doing git fetch and each using the configured amount of the memory. ⚠ For large repositories requiring heap memory greater than `-Xmx1024m` for Git fetch, switching to 64-bit Java may be needed.
teamcity.git.monitoring.expiration.timeout.hours	24	
teamcity.server.git.gc.enabled	false	Whether TeamCity should run `git gc` during the server cleanup (native git is used)
teamcity.server.git.executable.path	git	The path to the native git executable on the server
teamcity.server.git.gc.quota.minutes	60	Maximum amount of time to run `git gc`
teamcity.git.cleanupCron	0 0 2 * * ? *	Cron expression for the time of a cleanup in git-plugin, by default - daily at 2a.m.
teamcity.git.stream.file.threshold.mb	128	Threshold in megabytes after which JGit uses streams to inflate objects. Increase it if you have large binary files in the repository and see symptoms described in TW-14947

teamcity.git.buildPatchInSeparateProcess	true	Git-plugin builds patches in a separate process, set it to false to build patch in the server process. To build patch git-plugin has to read repository files into memory. To not run out of memory git-plugin reads only objects of size smaller than the threshold, for larger objects streams are used and they can be slow (TW-14947). With patch building in a separate process all objects are read into memory. Patch process uses the memory settings of the separate fetch process.
teamcity.git.mirror.expiration.timeout.days	7	The number of days after which an unused clone of the repository will be removed from the server machine. The repository is considered unused if there were no TeamCity operations on this repository, like checking for changes or getting the current version. These operations are quite frequent, so 7 days is a reasonably high value.
teamcity.git.commit.debug.info	false	Defines whether to log additional debug info on each found commit
teamcity.git.sshProxyType		Type of ssh proxy, supported values: http, socks4, socks5. Please keep in mind that socks4 proxy cannot resolve remote host names, so if you get an UnknownHostException, either switch to socks5 or add an entry for your git server into the hosts file on the TeamCity server machine.
teamcity.git.sshProxyHost		Ssh proxy host
teamcity.git.sshProxyPort		Ssh proxy port
teamcity.git.connectionRetryAttempts	3	Number of attempts to establish connection to the remote host for testing connection and getting a current repository state before admitting a failure
teamcity.git.connectionRetryIntervalSeconds	4	Interval in seconds between connection attempts

Agent configuration for Git:

Property	Default	Description
teamcity.git.use.native.ssh	false	When checkout on agent: whether TeamCity should use native SSH implementation.
teamcity.git.idle.timeout.seconds	1800	The idle timeout for the git fetch operation when the agent-side checkout is used. The fetch is terminated if there is no output from the fetch process during this time. Prior to 8.0.4 the default was 600.

Limitations

When using checkout on an agent, a limited subset of checkout rules is supported. Git-plugin translates some of the checkout rules to the sparse checkout patterns. Only the rules which do not remap files are supported:

```
+:some/dir
-:some/dir/subDir
```

An unsupported rule example is `+:some/dir=>some/otherDir`.

Known Issues

- java.lang.OutOfMemoryError while fetch repository. Usually occurs when there are large files in the repository. By default, TeamCity runs fetch in a separate process. To increase memory available to this process, change the teamcity.git.fetch.process.max.memory internal property (see description of this property above).
- Teamcity run as a Windows service cannot access a network mapped drives, so you cannot work with git repositories located on such drives. To make this work, run TeamCity using teamcity-server.bat.
- inflation using streams in JGit prevents `OutOfMemoryError`, but can be time-consuming (see the related thread at jgit-dev for details and the TW-14947 issue related to the problem). If you meet conditions similar to those described in the

issue, try to increase `teamcity.git.stream.file.threshold.mb`. Additionally, it is recommended to increase the overall amount of memory dedicated for TeamCity to prevent `OutOfMemoryError`.

Development Links

Git support is implemented as an open-source plugin. For development links, refer to the plugin's page.

See also:

- Administrator's Guide: Branch Remote Run Trigger