



# StarTeam 16.2

Git Command Line Utility Help

**Micro Focus**  
**The Lawn**  
**22-30 Old Bath Road**  
**Newbury, Berkshire RG14 1QN**  
**UK**  
<http://www.microfocus.com>

**Copyright © Micro Focus 2017. All rights reserved.**

**MICRO FOCUS, the Micro Focus logo and StarTeam are trademarks or registered trademarks of Micro Focus IP Development Limited or its subsidiaries or affiliated companies in the United States, United Kingdom and other countries.**

**All other marks are the property of their respective owners.**

**2017-11-02**

# Contents

<b>About the StarTeam Git Command Line Utility</b>	<b>4</b>
<b>Prerequisites</b>	<b>5</b>
<b>Known Issues</b>	<b>6</b>
<b>Supported Platforms</b>	<b>7</b>
<b>Installation and Licensing</b>	<b>8</b>
User Credentials	8
<b>Architecture</b>	<b>9</b>
<b>Typical Workflow</b>	<b>10</b>
<b>Supported Git Clients</b>	<b>11</b>
GIT GUI	11
SourceTree	11
Eclipse	12
<b>Commands</b>	<b>13</b>
Clone Command	13
Fetch Command	14
Pull Command	16
Push Command	17

# About the StarTeam Git Command Line Utility

The StarTeam Git Command Line Utility brings centralized source code control and security to teams using Git. Git users can connect to the StarTeam Server using the command line utility to push and pull changes with their local Git repository. Teams are able to share Projects in StarTeam such that developers can work synchronously either in Git or StarTeam clients.

With the StarTeam Git Command Line Utility, developers can:

- Use the `clone` command to clone a StarTeam View from a StarTeam Server and populate the master branch in a local Git repository with the tip content.
  - ⚠ **Important:** The `clone` command initializes the developers local Git workspace and allows them to issue further commands to interact with the StarTeam View content. User credentials specified during the `clone` command will be cached and re-used during subsequent `fetch`, `pull`, and `push` commands from the same local repository unless a different username or password is specified during those commands. However, caching of credentials is only available on Microsoft Windows platforms and all commands issued from Linux must specify a username and password.
- Use the `fetch` and `pull` commands to update a local Git repository with changes from StarTeam. When you `pull` changes, each StarTeam *Change Package* becomes a commit in a local Git repository.
- Use the `push` command to push changes from a local Git repository into StarTeam Server. Each Git commit becomes a StarTeam Change Package, including the user ID, timestamp, and comment.

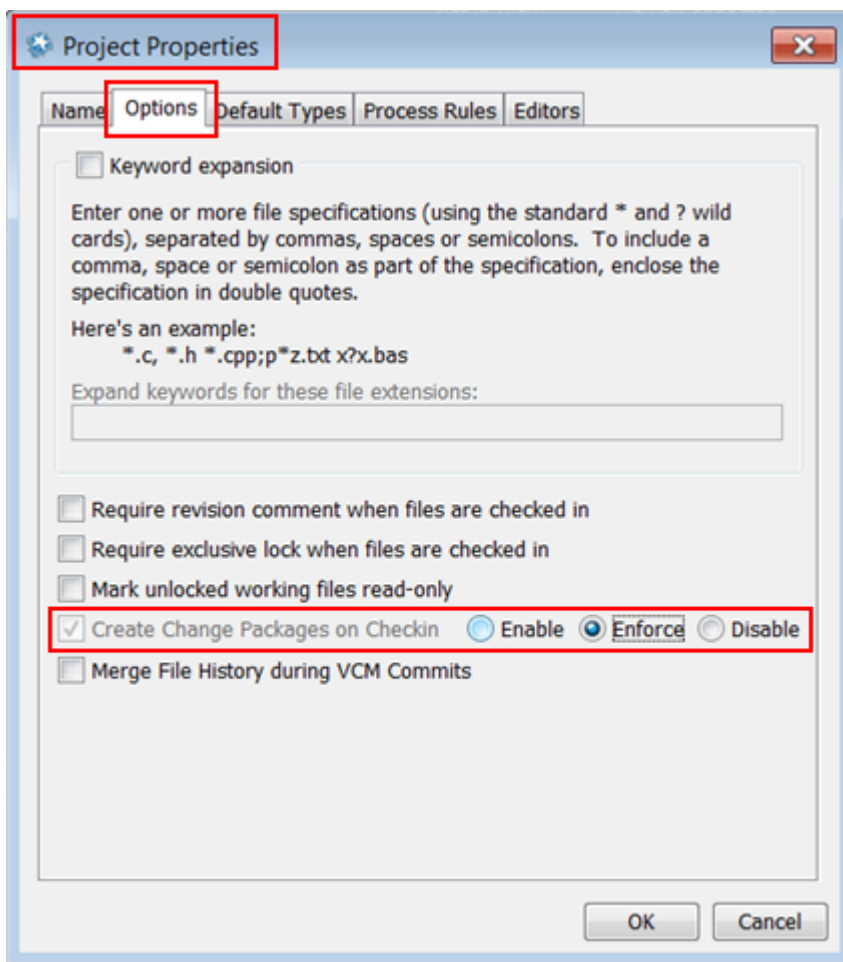
## Benefits

- Developers are able to use their preferred Git client tools on a local Git repository while connecting with the centralized StarTeam source code control system using the StarTeam Git Command Line Utility.
- Enterprises retain what StarTeam offers: rich change management, ALM traceability, Issue/Process enforcement, security, and visibility.

# Prerequisites

The following are required to use the StarTeam Git Command Line Utility:

- StarTeam Server 16.2 +.
- The StarTeam project being cloned requires the **Create Change Packages on Checkin** property to be set to **Enforce**. This property ensures that any StarTeam client accessing this project performs all file/folder updates contained within a *Change Package*. Once this property is set, only StarTeam 16.2 + clients are capable of updating the project. The StarTeam Git Command Line Utility `clone` command will attempt to automatically set this property, but if the user does not have appropriate access rights to set the property, they will be instructed to contact their StarTeam Server Administrator to set the project property from the StarTeam Cross-Platform Client:



# Known Issues

<b>GUI Tools</b>	Some of the suggested GUI tools do not provide full output when an error occurs during a command. For full output, please refer to the StarTeam Git Command Line Utility LOG file located at %APPDATA%\Borland\StarTeam\git-starteam-connector-*.log.
<b>Submodule Synchronization</b>	The StarTeam Git Command Line Utility does not support synchronizing <i>submodules</i> . A <i>submodule</i> is a repository embedded in your main repository.
<b>Empty Folders</b>	Git does not allow empty folders. Any empty folders in StarTeam will not be created in the Git repository.
<b>Git Commit IDs and StarTeam Change Package IDs</b>	The StarTeam Git Command Line Utility maintains an internal mapping between <code>Git Commit IDs</code> and <code>StarTeam Change Package IDs</code> . Commands like <code>filter-branch</code> , <code>commit-ammend</code> , <code>squash</code> , and any other similar commands which re-write the commit history in the Git repository, could potentially affect the working of the synchronization between StarTeam and Git.
<b>StarFlow Extensions Project</b>	The <code>StarFlow Extensions</code> project cannot be cloned.
<b>Merge Conflicts</b>	If there are <code>merge</code> conflicts during a <code>pull</code> operation that need to be resolved, the error message will specify the name of only the first file that was in conflict.
<b>Linux</b>	Symbolic links are not supported in this release of the StarTeam Git Command Line Utility.

# Supported Platforms

The following operating systems are supported for this release:

- Microsoft Windows 10.
- Microsoft Windows 8.
- Microsoft Windows 7.
- Red Hat Enterprise Linux 7.3.
- SUSE 11.3.



**Note:** For UNIX operating systems, if the Git config option `core.filemode` is set to `true`, the StarTeam Git Command Line Utility will honor the executable bit of files during `clone`, `pull`, or `push`. For any file on StarTeam Server with `Executable` property enabled, the corresponding file in the Git working tree will have the executable bit set, and vice versa.

If Git config option `core.filemode` is set to `false`, the executable bit is not honored and will not be set.

# Installation and Licensing

## Installation

The StarTeam Git Command Line Utility files are installed as part of the StarTeam SDK.

### Microsoft Windows

Navigate to `C:\Program Files\Micro Focus\StarTeam SDK <version #>\lib`.

### Linux

1. Run the `git-st-setup` file under `<StarTeam SDK install folder>/bin`.
2. Update your `Path` environment variable to include `<StarTeam SDK install folder>/lib`

For running the utility and command syntax, see [Commands](#).



**Note:** Because the StarTeam Git Command Line Utility capability is bundled within the StarTeam SDK, any installation (and uninstallation) instructions can be found in the *StarTeam Installation Guide*. For example, to uninstall the StarTeam SDK (and thereby uninstall the StarTeam Git Command Line Utility), use the **SDK Runtime Uninstall** option on the Windows **Start** menu. If uninstalling the StarTeam SDK, please be sure other StarTeam applications are not using the SDK.

## Licensing

Users of the StarTeam Git Command Line Utility must provide their valid StarTeam credentials when submitting commands. As such, each user must have a valid StarTeam license.

## User Credentials

Newer versions of Git (for example, Git for Microsoft Windows 2.13.3) support Git *Credentials Manager*, which provides a way to securely store credentials on Microsoft Windows . If the Git *Credentials Manager* was enabled during the installation of Git, StarTeam Git Command Line Utility will use it to store the credentials of the StarTeam user in Microsoft Windows *Credential Store*. The user will be prompted to enter their StarTeam credentials once during `clone` and will not be prompted to enter on every subsequent `pull/push` command (if `username` and `password` are not specified as part of the command).

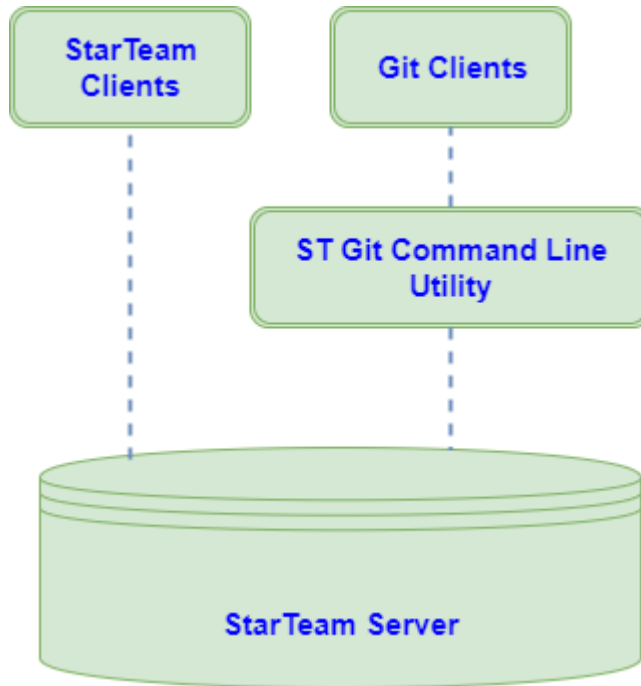


**Note:** The credentials can be edited/removed from the **Control Panel > Credential Manager**.

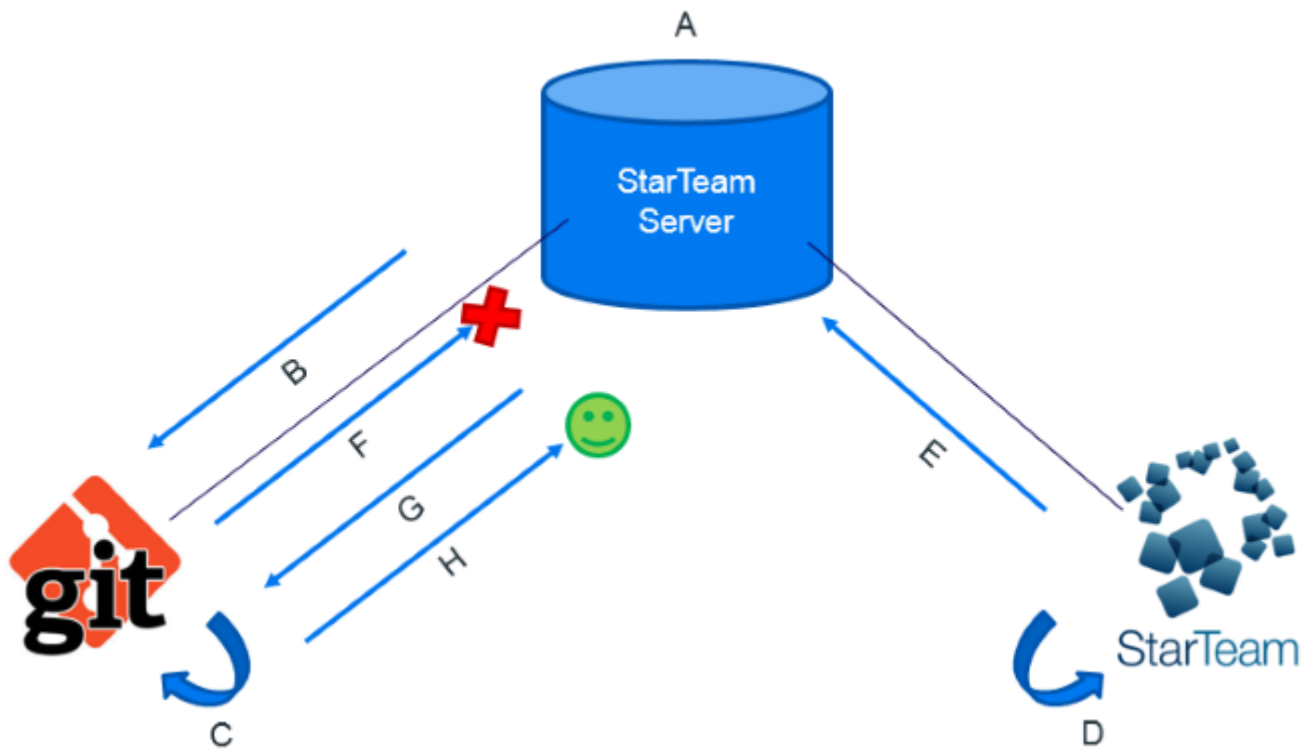


# Architecture

StarTeam is an enterprise SCCM (Software Configuration and Change Management) repository that securely stores code and artifacts from teams using multiple native clients as well as Git.



# Typical Workflow



- A** A StarTeam Server View is under the central control of StarTeam Server.
- B** A Git developer clones the View into their local Git repository.
- C** The Git developer branches, commits, and works as normal with their Git repository.
- D** Another developer is using a StarTeam client. They update their local StarTeam controlled work area from the same View.
- E** The second developer works on the code and pushes their changes to StarTeam Server.
- F** The Git developer tries to push their changes. The push fails as there are changes in the View that they need to merge.
- G** The Git developer pulls the changes and merges them into their local Git repository.
- H** The Git developer can now successfully push their changes to StarTeam Server.

# Supported Git Clients

The following Git clients can be configured with the StarTeam Git Command Line Utility.

<b>GIT GUI</b>	<a href="https://git-scm.com/docs/git-gui">https://git-scm.com/docs/git-gui</a>
<b>SourceTree</b>	<a href="https://www.sourcetreeapp.com/">https://www.sourcetreeapp.com/</a>
<b>Eclipse</b>	<a href="https://www.eclipse.org/downloads/">https://www.eclipse.org/downloads/</a>

The following sections contain configuration information for these tools.

## GIT GUI

Git repositories cloned using StarTeam Git Command Line Utility will automatically be configured to be accessed from GIT GUI. The following configuration options are set in the Git config file (`.git\config`). The initial clone must be performed using the command line.

Once cloned, users can use the GIT GUI menu options to **Pull**, **Fetch** or **Push to StarTeam Server**. Password and other options can be entered in the input dialog box.

### Customizing the Commands

The following configuration options are saved by default in the `.git\config` file and can be customized to add more default command options.

```
[guitool "StarTeam/Pull"]
  cmd = git st pull
  argprompt = yes
[guitool "StarTeam/Push"]
  cmd = git st push
  argprompt = yes
[guitool "StarTeam/Fetch"]
  cmd = git st fetch
  argprompt = yes
```

**Pull**, **Push**, and **Fetch** options can be found under the **Tools > StarTeam** menu of GIT GUI. Arguments can be entered in the input dialog box.

For example, click **Tools > Add** to open the **Add Tool** dialog box.

1. In the **Name** field, enter `StarTeam\pull`.
2. In the **Command** field, enter `git st pull --auto-cache-agent`.

## SourceTree

You can use Git commands in SourceTree 2.x for the StarTeam Git Command Line Utility after the following has been performed.

1. Perform an initial clone using the StarTeam Git Command Line Utility.
2. Add individual custom actions in SourceTree to `pull`, `fetch`, or `push` commands from StarTeam Server.

## Adding the Commands

1. In SourceTree, click **Tools > Options**.
2. Click the **Custom Actions** tab.
3. Click **Add**.
4. In the **Menu Caption** field, enter `StarTeam Push`.
5. In the **Script to Run** field, enter `git`.
6. In the **Parameters** field, enter `st push`.

Repeat the same steps for `Pull` and `Fetch`.

## Further Configuration

Password and other options can be entered in the command prompt which is opened when a menu action is performed.

Click **Tools > Custom Actions > StarTeam Pull**, for example.

Custom action configuration can also be edited in the SourceTree configuration file

```
%  
LOCALAPPDATA%\Atlassian\SourceTree\customactions.xml
```

# Eclipse

You can use Git commands in Eclipse Neon (4.6) platform for the StarTeam Git Command Line Utility after the following has been performed:

1. Perform an initial clone using the StarTeam Git Command Line Utility.
2. Configure the Eclipse **External Tools** menus to `pull`, `fetch` or `push` to StarTeam Server.

## Configuration

To configure external tools to perform `pull`, `push`, or `fetch` commands from StarTeam Server:

1. Click **Run > External Tools > External Tools Configuration**.
2. In the **Location** field, enter the location of your `Git.exe`. For example, `C:\Program Files (x86)\Git\bin\git.exe`.
3. In **Working Directory**, enter the Workspace location of the project.
4. In the **Arguments** field, enter `st push -pi ? lbl ?`
5. To configure the `pull` and `fetch` commands, repeat steps 1-3 above, but for step 4 use `st pull` and `st fetch`, respectively.



**Note:** Eclipse's external tool configurations are saved under `<Eclipse workspace location> \.metadata\.plugins\org.eclipse.debug.core\.launches\`.

## Running the Commands

Go to **Run > External Tools > StarTeam Pull/Fetch/Push**. The output can be seen in Eclipse's console window.


# Commands

The StarTeam Git Command Line Utility provides commands for a Git developer to interact with the StarTeam Server.

To use the utility, open your command prompt and navigate to the `C:\Program Files\Micro Focus\StarTeamSDK <version #>\lib` and type in:


**If Git command line is installed** `git st <command> <options> <arguments>`


**If Git command line is not installed on the system** `sh git-st <command> <options> <arguments>`

 **Note:** If the StarTeam SDK `lib` folder is on the `PATH` then these commands can be issued from any directory in the command prompt.

## Clone Command

The `Clone` command clones a StarTeam view into a new Git repository.

 **Important:** The `Clone` command is the first step in setting up a local Git repository for further `Push` and `Pull` commands with StarTeam.

 **Important:** The `Clone` command will recreate the folders and files from StarTeam in the local Git repository using the default working directories from the StarTeam View.

### Arguments

**<starteam connection string>** Connection string format:  
`[starteam://]<hostname>:<port>/<project name>[/<view name>][/<folder name>]`

**<directory>** Git working directory.

### Options:

**--auto-cache-agent** *Autolocate* the Cache Agent to use while checking out the source code.

**--bare (-b)** Clone to a bare repository.

**--eol (cr|lf|crlf|off|platform)** Automatically convert the *end-of-line* markers while checking out files from StarTeam Server. The default is off, no end-of-line conversion is performed.



### Note:

- Regardless of the **EOL** setting in the file's properties in the StarTeam Cross-Platform Client (Fixed `CR\LF\CRLF`, Client defined), the `-eol cr`, `lf`, and `crlf` value will always override that setting.
- When `-eol platform` option is used then pulled/fetched files will have the platform-specific EOL setting:
  - Microsoft Windows: `CRLF`.
  - Linux: `LF`.


- Mac: - CR.


<b>--help (-h)</b>	Command usage text.
<b>--password (-p)</b> <b>&lt;password&gt;</b>	StarTeam Password. If the password is not specified and has not been cached during previous commands then you will be prompted for a password.
<b>--quiet (-q)</b>	Progress is not reported.
<b>--use-cache-agent</b> <b>&lt;host:port&gt;</b>	Use Cache Agent <code>&lt;host:port&gt;</code> while checking out the source code.
<b>--username (-u)</b> <b>&lt;username&gt;</b>	StarTeam user that will perform this operation. If the StarTeam <b>User name</b> is not specified during the <code>fetch/pull/push</code> , then the user that was specified during the <code>clone</code> will automatically be used.
<b>--verbose (-v)</b>	Verbose mode.

### Example 1: Cloning the Full View Hierarchy

This example shows you how to `clone` to a new Git repository where:

- *Project* = SampleProject
- *View* = SampleView
- *Server* = 10.150.1.12
- *Port* = 49201
- *Working Directory* = C:\gitdemo\Sample


 **Important:** Use a colon `:` as the delimiter during `clone`

 **Note:** This example of specifying a full view hierarchy can also be used in the `fetch`, `pull` or `push` commands as well.

```
C:\gitdemo\Sample>git st clone "10.150.1.12:49201/SampleProject/SampleView:
1.x Release:1.2 Release"
```

### Example 2: Cloning from a Sub Folder

```
C:\gitdemo\Sample>git st clone "10.150.1.12:49201/SampleProject/
SampleView/src/core/java"
```

 **Important:** If View names within a StarTeam project are unique, then the Project Name/View Name format may be used in the command without the need to specify the full View hierarchy.


## Fetch Command

The `Fetch` command fetches the latest changes from a StarTeam View into `FETCH_HEAD` of the local Git repository.

### Arguments

<b>&lt;starteam connection string&gt;</b>	Connection string format: [starteam://]<hostname>:<port>/<project name>[/<view name>][/<folder name>]
<b>&lt;directory&gt;</b>	Git working directory.

## Options


<b>--auto-cache-agent</b>	<i>Autolocate</i> the Cache Agent to use while checking out the source code.
<b>--deep (-d)</b>	Performs a deep <i>fetch/pull/push</i> , creating <i>commits</i> in the Git repository for each StarTeam Change Package since the last <i>fetch</i> . If the <i>deep</i> option is not specified, then the collection of StarTeam Change Packages will be grouped into a single <i>commit</i> in Git.
<b>--eol (cr lf crlf off platform)</b>	Automatically convert the <i>end-of-line</i> markers while checking out files from StarTeam Server. The default is off, no end-of-line conversion is performed.
	 <b>Note:</b>
	<ul style="list-style-type: none"><li>• Regardless of the <b>EOL</b> setting in the file's properties in the StarTeam Cross-Platform Client (<i>Fixed CR\LF\CRLF, Client defined</i>), the <i>-eol cr, lf, and crlf</i> value will always override that setting.</li><li>• When <i>-eol platform</i> option is used then <i>pulled/fetched</i> files will have the platform-specific EOL setting:<ul style="list-style-type: none"><li>• Microsoft Windows: CRLF.</li><li>• Linux: - LF.</li><li>• Mac: - CR.</li></ul></li></ul>
<b>--force (-f)</b>	Performs a force <i>Fetch/Pull</i> by downloading the latest Change Set.
<b>--help (-h)</b>	Command usage text.
<b>--password (-p)</b> <b>&lt;password&gt;</b>	StarTeam Password. If the password is not specified and has not been cached during previous commands then you will be prompted for a password.
<b>--quiet (-q)</b>	Progress is not reported.
<b>--shallow</b>	Creates a single <i>commit</i> for all StarTeam Change Packages in the <i>fetch</i> .
<b>--use-cache-agent</b> <b>&lt;host:port&gt;</b>	Use Cache Agent <i>&lt;host:port&gt;</i> while checking out the source code.
<b>--username (-u)</b> <b>&lt;username&gt;</b>	StarTeam user that will perform this operation. If the StarTeam <b>User name</b> is not specified during the <i>fetch/pull/push</i> , then the user that was specified during the <i>clone</i> will automatically be used.
<b>--verbose (-v)</b>	Verbose mode.

## Example

Fetches the latest changes from StarTeam Server into *FETCH\_HEAD* ref. Run the following command to perform a *fetch*.

```
C:\gitdemo\Sample>git st fetch
```

Followed by a prompt for password.

 **Important:** As the subsequent *fetch, pull, and push* commands are issued against the same Project/View/Folder hierarchy as the *clone* command, if you choose to provide the Project/View/Folder during these commands then they must match the structure from *clone*. We recommend that you only pass the Project/View/Folder path in a *fetch, pull, or push* if you need to specify a new server address or port.


# Pull Command

The `pull` command performs a `Fetch` from a StarTeam Server View and then performs a `merge/rebase` into the local `Master branch`.

## Arguments

<code>&lt;starteam connection string&gt;</code>	Connection string format: <code>[starteam://]&lt;hostname&gt;:&lt;port&gt;/&lt;project name&gt;[/&lt;view name&gt;][/&lt;folder name&gt;]</code>
<code>&lt;directory&gt;</code>	Git working directory.

## Options

<code>--auto-cache-agent</code>	<i>Autolocate</i> the Cache Agent to use while checking out the source code.
<code>--deep (-d)</code>	Performs a deep <code>fetch/pull/push</code> , creating <code>commits</code> in the Git repository for each StarTeam Change Package since the last <code>fetch</code> . If the <code>deep</code> option is not specified, then the collection of StarTeam Change Packages will be grouped into a single <code>commit</code> in Git.
<code>--eol (cr lf crlf off platform)</code>	Automatically convert the <i>end-of-line</i> markers while checking out files from StarTeam Server. The default is off, no end-of-line conversion is performed.   <b>Note:</b> <ul style="list-style-type: none"><li>Regardless of the <b>EOL</b> setting in the file's properties in the StarTeam Cross-Platform Client (Fixed <code>CRLF\CRLF</code>, Client defined), the <code>-eol cr</code>, <code>lf</code>, and <code>crlf</code> value will always override that setting.</li><li>When <code>-eol platform</code> option is used then pulled/fetched files will have the platform-specific EOL setting:<ul style="list-style-type: none"><li>Microsoft Windows: <code>CRLF</code>.</li><li>Linux: - <code>LF</code>.</li><li>Mac: - <code>CR</code>.</li></ul></li></ul>
<code>--force (-f)</code>	Performs a force <code>Fetch/Pull</code> by downloading the latest Change Set.
<code>--help (-h)</code>	Command usage text.
<code>--password (-p)</code> <code>&lt;password&gt;</code>	StarTeam Password. If the password is not specified and has not been cached during previous commands then you will be prompted for a password.
<code>--quiet (-q)</code>	Progress is not reported.
<code>--rebase (-r)</code>	Incorporate changes by rebasing rather than merging.
<code>--shallow</code>	Creates a single <code>commit</code> for all StarTeam Change Packages in the <i>fetch</i> .
<code>--strategy (-s)</code> <code>&lt;strategy&gt;</code>	The merge strategy to use. Valid Git merge strategies such as <code>ours</code> , <code>theirs</code> and <code>resolve</code> are supported.
<code>--use-cache-agent</code> <code>&lt;host:port&gt;</code>	Use Cache Agent <code>&lt;host:port&gt;</code> while checking out the source code.



<b>--username (-u)</b> <b>&lt;username&gt;</b>	StarTeam user that will perform this operation. If the StarTeam <b>User name</b> is not specified during the <code>fetch/pull/push</code> , then the user that was specified during the <code>clone</code> will automatically be used.
<b>--verbose (-v)</b>	Verbose mode.

### Example

This example `pulls` the latest changes from StarTeam Server to a configured Git repository.

```
C:\gitdemo\Sample>git st pull
```

Followed by a prompt for password.

## Push Command

The `Push` command pushes changes from the local Git repository's master to the configured StarTeam Server view.

### Arguments

<b>&lt;starteam connection string&gt;</b>	Connection string format: [starteam://]<hostname>:<port>/<project name>[/<view name>][/<folder name>]
<b>&lt;directory&gt;</b>	Git working directory.

### Options

<b>--attach-label (-lbl) &lt;label name&gt;</b>	Specifies a StarTeam Label to be applied to the <code>pushed</code> files. The Label can be either a View or Revision Label, but it must already exist on the Server. The value can be a placeholder (?) in which case, the user will be prompted to enter the Label <b>Name</b> .
<b>--autoignore</b>	Automatically select the commit paths in Git repository tree to ignore when <code>commit</code> has more than one parent.
<b>--changepackage-name &lt;changepackage name&gt;</b>	Over-ride the name of the Change Package.
<b>--changerequest (-cr) &lt;crID&gt;</b>	Choose StarTeam Change Request as Process Item during <code>push</code> to StarTeam Server.
<b>--clear-active-pi</b>	Clear the chosen Active Process Item upon a successful <code>push</code> .
<b>--complete</b>	Mark the chosen Process Item as complete upon a successful <code>push</code> .
<b>--create &lt;project connection url&gt;</b>	Create a new project on the remote StarTeam Server. URL should be of the format <hostname>:<port>/<project name>.
<b>--deep (-d)</b>	Performs a <code>deep push</code> that creates Change Packages in the StarTeam Server for each <code>commit</code> in the Git repository since the last <code>push</code> .
<b>--help (-h)</b>	Command usage text.
<b>--ignore (-i) id1,id2</b>	Ignore <code>commit</code> IDs when performing <code>deep push</code> if a <code>commit</code> has more than one parent.
<b>--lock (-l)</b>	Obtain a StarTeam lock on files/folders while <code>pushing</code> to StarTeam Server.

<b>--metadata</b>	Include Git commit metadata in the Change Set when performing a <code>deep push</code> .
<b>--no-tags</b>	Tags associated with a <code>commit</code> are pushed to StarTeam Server as Labels by default. This option is used to disable the feature.
<b>--non-exclusive-lock (-x)</b>	Obtain a non-exclusive lock on StarTeam Files/Folders being <code>pushed</code> .
<b>--password (-p) &lt;password&gt;</b>	StarTeam Password. If the password is not specified and has not been cached during previous commands then you will be prompted for a password.
<b>--processitem (-pi) &lt;type:pild&gt;</b>	Choose a StarTeam Process Item to use during <code>push</code> . Any valid Process Item type can be used. The value can also be a placeholder, ( ? ), in which case, the user will be prompted to enter the Process Item <code>Type</code> and <code>ID</code> .
<b>--quiet (-q)</b>	Progress is not reported.
<b>--requirement (-r) &lt;reqID&gt;</b>	Choose StarTeam Requirement to use during <code>push</code> to StarTeam Server.
<b>--set-active-pi</b>	Set the chosen Process Item as <code>Active</code> upon a successful <code>push</code> .
<b>--shallow</b>	Performs a <code>shallow push</code> , creating just one Change Package in the StarTeam Server for all the <code>commits</code> in the Git repository since last <code>push</code> .
<b>--task (-t) &lt;tskID&gt;</b>	Choose a Task as Process Item during <code>push</code> to StarTeam Server.
<b>--unlock</b>	Unlock Files/Folders while <code>pushing</code> to StarTeam Server.
<b>--username (-u) &lt;username&gt;</b>	StarTeam user that will perform this operation. If the StarTeam <b>User name</b> is not specified during the <code>fetch/pull/push</code> , then the user that was specified during the <code>clone</code> will automatically be used.
<b>--verbose (-v)</b>	Verbose mode.

## Example

To `Push` the latest changes from Git repository to StarTeam Server using the following command:

```
C:\gitdemo\Sample>git st push
```

To create a new Project on StarTeam Server from an existing Git repository, use the `--create` option.

The following example would create a new Project called `TestProject` on StarTeam Server and push the source code from the Git repository using the `shallow` option. The command has to be run from within Git working directory.

```
C:\gitdemo\ExampleGit> git st push --create starteam://10.150.1.12:49201/
TestProject --shallow
```

## Specifying Process Items

Process Items can be specified in one of the following ways:

- The following `push` command will use `ChangeRequest 6144` and set it as the Active Process Item:

```
C:\Test\gitRepo> git st push -cr 6144 --set-active-pi
```

Subsequent pushes will use `ChangeRequest 6144` as process item:

```
C:\Test\gitRepo> git st push
```

- The following command will mark the active Change Request as `Fixed` and clear the Active Process Item on the workstation.

```
C:\Test\gitRepo> git st push --complete --clear-active-pi
```

- Associate push changes with ChangeRequest 5561:  
C:\gitdemo\Sample>git st push -cr 5561
- Associate push changes with Task 5001:  
C:\gitdemo\Sample>git st push -t 5001
- Associate push changes with Requirement 531:  
C:\gitdemo\Sample>git st push -r 531
- Associate push changes with a custom type item Defect and id=531:  
C:\gitdemo\Sample>git st push -pi Defect:531
- Associate push changes with ChangeRequest 5561 and mark it as complete:  
C:\gitdemo\Sample>git st push -cr 5561 --complete

For *Out-of-View Process Items*, specify the Project name and View name in front of the complete folder path. Separate the view path with a colon (:). For example, `-cr MyProject/RootView:ChildView/SourceCode/37` specifies Change Request 37 in the SourceCode Folder of the ChildView View in the MyProject Project.

During execution, the process first assumes that the Process Item is in the current View, and it checks the current View to determine whether the full path corresponds to a folder path within that View. If the Process Item is not found in the current View, it is treated as an Out-of-View Process Item, and the search for the Process Item begins from the Project and View.

To associate push changes with Change Request 5561 in a different view, 11.0 release, and mark it as complete:

```
C:\gitdemo\Sample>git st push -cr Sample:Sample/11.0 Release/5561 --complete
```

Alternatively, StarTeam Process Items to use can be specified as part of a Git commit message. The subsequent push to StarTeam Server will recognize the Process Item to use from the commit messages.

Examples:

- C:\gitdemo\Sample> git add
- Specify using ChangeRequest 5651:  
C:\gitdemo\Sample> git commit -m "Fix for ChangeRequest:5651"
- Specify using Task 5651:  
C:\gitdemo\Sample> git commit -m "Fix for Task:5651"
- Specify using Requirement 5651:  
C:\gitdemo\Sample> git commit -m "Fix for Requirement:5651"
- Push the above Git commit using the process item in the commit message:  
C:\gitdemo\Sample> git st push

# Index

## A

about 4  
architecture 9

## C

clone command 13  
commands 13

## E

Eclipse 12

## F

fetch command 14

## G

GIT GUI 11

## I

installation 8

## K

known issues 6

## L

licensing 8  
Linux limitation 4

## P

prerequisites 5  
pull command 16  
push command 17

## S

SourceTree 11  
support Git clients 11  
supported platforms 7

## U

user credentials 8

## W

workflow 10