Git Clone Repository Action

The Git Clone Repository action allows you to clone an existing git repository into a new directory. This action is a wrapper for the git command line. For more information on the use and options for this action, refer to the git clone command line documentation.

To clone a remote repository, you need to specify a working directory and the URL of the repository to clone from. The destination directory cannot already contain a repository, otherwise the command will fail.

Working directory - The directory to run the command line form.

Repository to clone - The URL of the repository to clone e.g. https://host.com[:port]/path/to/repo.git, ssh://[user@]host.com[:port]/path/to/repo.git or git://host.com[:port]/path/to/repo.git.

Clone is a local repository - When the repository to clone from is on a local machine, this flag bypasses the normal "Git aware" transport mechanism and clones the repository by making a copy of HEAD and everything under objects and refs directories.

No hard links - Force the cloning process from a repository on a local filesystem to copy the files under the `.git/objects` directory instead of using hardlinks. This may be desirable if you are trying to make a back-up of your repository.

Shared - When the repository to clone is on the local machine, instead of using hard links, automatically setup `.git/objects/info/alternates` to share the objects with the source repository. The resulting repository starts out without any object of its own. **NOTE**: this is a possibly dangerous operation; do **not** use it unless you understand what it does.
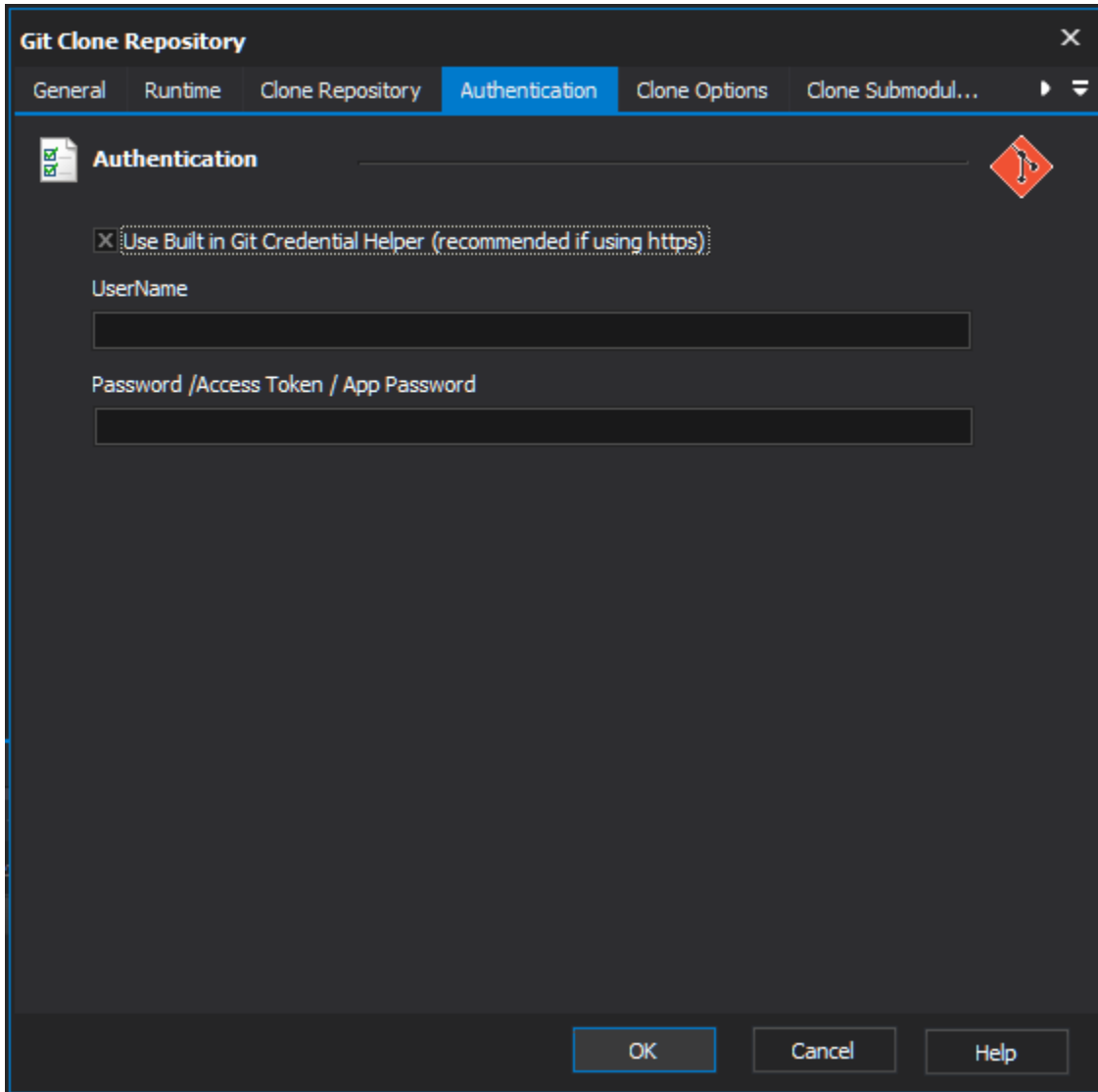
Name of the directory to clone into - This is the destination for the clone and is relative to the working directory. If this field is not provided then the last name in the URL will be used as the destination directory.

Reference repository - Path to a reference repository on the local machine to obtain objects from. This can allow fewer objects to be copied from the repository being cloned, reducing network and local storage costs.

Non-existing directories are skipped with a warning instead of aborting the clone - Skip folders not able in reference repository without aborting the clone.

Template - A directory containing templates will be copied to the .git directory. The template directory can include some directory structure, ignore patterns and hook files.
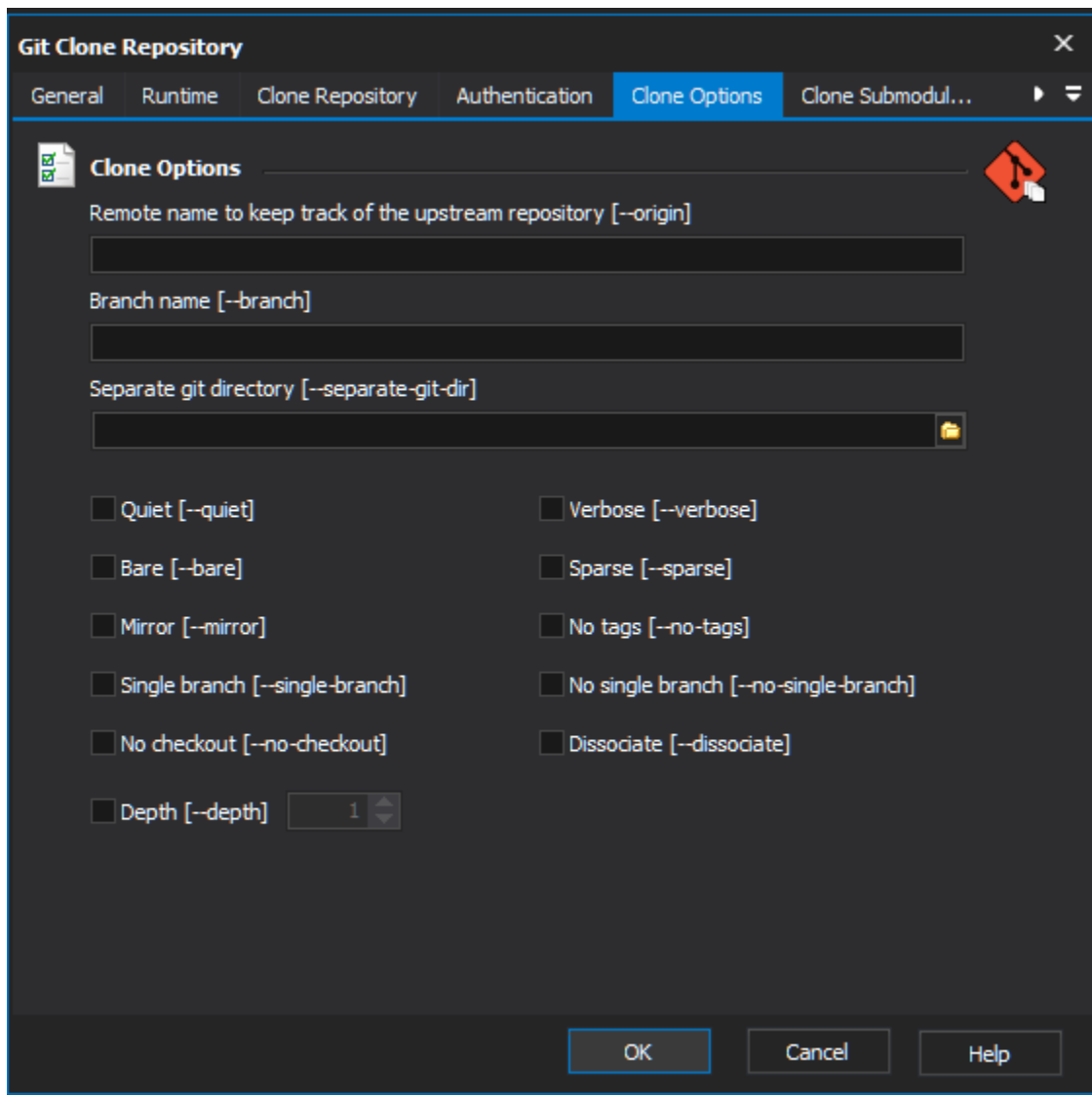
If your repository is private you will need to enter credentials on the Authentication tab.

We recommend that you use the built-in credentials helper to pass the credentials to the command line when connecting to the remote repository using an HTTPS.

There are many options that can be set to control the cloning process.

Remote name to keep track of the upstream repository - Name to use instead of origin for the upstream repository.

Branch name - Name of branch to point to (or checkout) in the cloned repository instead of pointing to the HEAD branch.

Separate git directory - A directory to place the cloned repository in which is separate to the working tree.

Quiet - Suppress output from Git.

Verbose - Verbose output from Git.

Bare - Rather than place the administrative files in the .git folder place then in the root destination directory. This implies that there is no checkout because there is nowhere to check out the working tree.

Sparse - Employ a sparse-checkout, with only files in the top-level directory initially being present.

Mirror - Set up a mirror of the source repository. This implies `--bare`, but rather than mapping only local branches of the source to local branches of the target, it also maps all refs (including remote-tracking branches, notes etc.).

No tags - Don't clone any tags or include tags in subsequent fetch and pull commands.

Single branch - Clone only the history leading to the tip of a single branch, either specified by the `--branch` option or the primary branch remote's `HEAD` points at.

No single branch - Use to override single branch default when running a shallow clone e.g. setting depth to 1.

Single branch - Clone only the history leading to the tip of a single branch, either specified by the `--branch` option or the primary branch remote's `HEAD` points at.

**No checkout** - Do not checkout the HEAD once the clone has completed.

Dissociate - Borrow the objects from reference repositories specified with the `--reference` option only to reduce network transfer, and stop borrowing from them after a clone is made by making necessary local copies of borrowed objects.

Depth - Limit the resulting repository to the specified number of revisions to create a shallow repository.

⚠️   Shallow repositories have a number of limitations that users should be aware of before using this option.

There are also some options which control how submodules are dealt with.
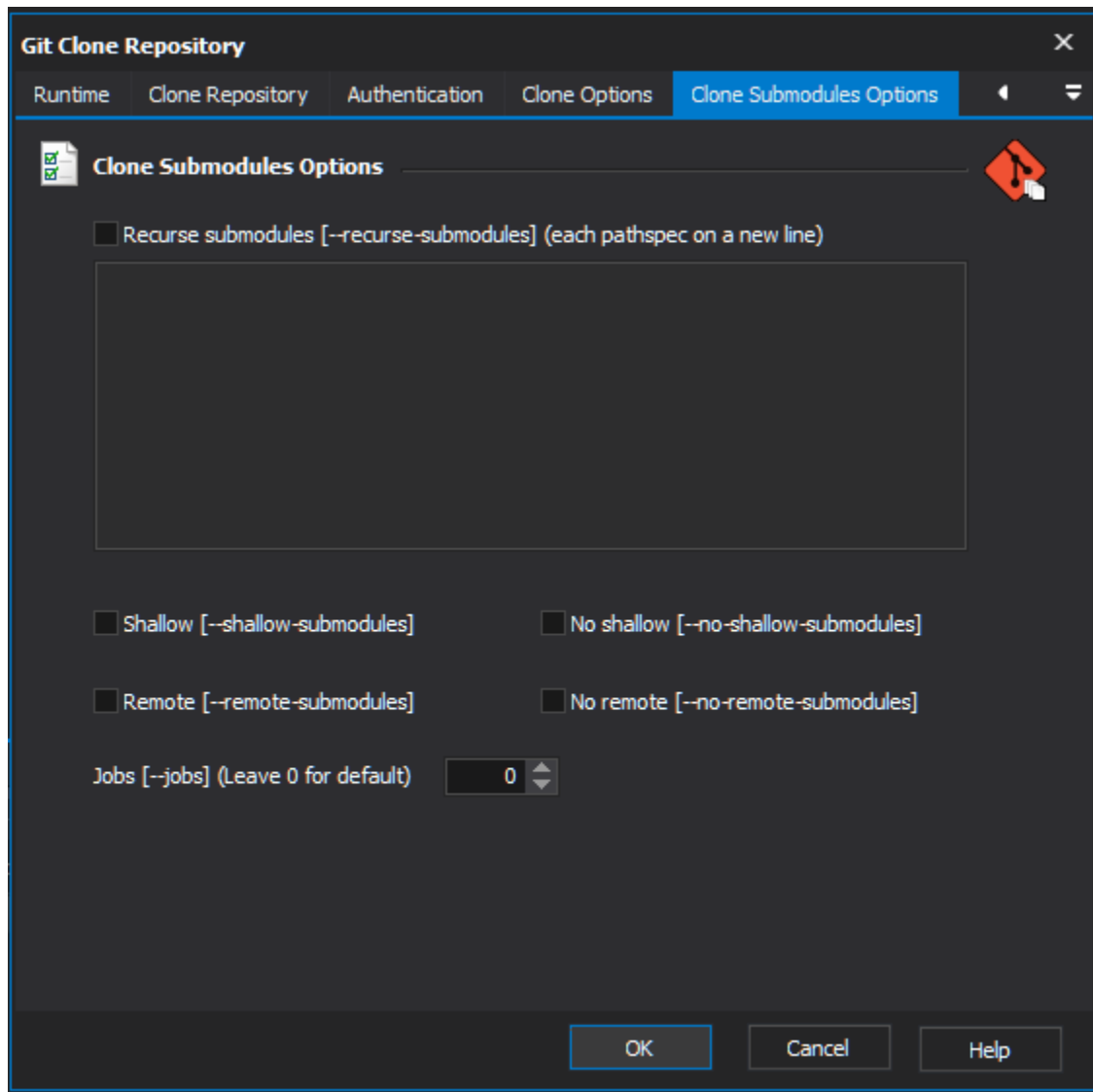
Recurse submodules - After the clone is created, initialize and clone submodules within based on the provided pathspec. If no pathspec is provided, all submodules are initialized and cloned. The submodules are initialized and cloned using their default settings.

Shallow - All submodules which are cloned will be shallow with a depth of 1.

No shallow - Ensure that full submodule history is cloned regardless of other settings.

Remote - All submodules which are cloned will use the status of the submodule's remote-tracking branch to update the submodule, rather than the main repository's recorded SHA-1 hash.

No remote - Ensure that the main repository's recorded SHA-1 hash to update the submodule regardless of other settings.

Jobs - The number of submodules to fetch at the same time.

For more information on cloning a git repository, see the git clone command line documentation.