使用吉特

Git是一个快速分散的版本控制系统。Git 可用于 Windows、Linux 和 macOS。

您可以将 Gerrit 代码审查工具用于使用 Git 的项目。

设置版本控制系统

Qt设计工作室使用版本控制系统的命令行客户端来访问您的仓库。若要允许访问，请确保可以使用环境变量找到命令行客户端。或者，在**"编辑>首选项"**>"版本控制"中"版本控制系统特定"选项卡的"**命令**"字段中指定命令行客户端可执行文件**的路径。**PATH

如果需要身份验证才能访问存储库，请在**"用户名"和"密码"**字段中输入用户凭据。

在"超时"字段中输入版本控制操作的**超时。**

对于某些版本控制系统，您可以在日志**计数**字段中指定日志可以包含的最大行数。

设置版本控制系统后，使用命令行检查一切是否正常工作（例如，使用 status 命令）。如果没有出现任何问题，您应该准备好从Qt设计工作室使用该系统。

有关将 Git 用于 Windows 的更多信息，请参阅将 Git 用于 Windows。

设置常规选项

选择**"编辑>首选项"**>**"版本控制"**>**"常规"**以指定提交邮件的设置：

- **将提交邮件的换行**时间限制在以下位置将提交邮件的行长度限制为指定的字符数。
- **提交消息检查脚本**是可用于在提交之前对提交消息执行检查的脚本或程序。提交消息作为脚本的第一个参数传入。如果出现错误，脚本应输出有关标准错误的消息，并返回非零退出代码。
- **用户/别名配置文件**是以邮件映射格式列出作者姓名的文本文件。对于每个作者，您必须指定真实姓名和电子邮件地址，还可以选择指定别名和第二个电子邮件地址。例如：

```
Jon Doe <Jon.Doe@company.com> jdoe <jdoe@somemail.com>
Hans Mustermann <Hans.Mustermann@company.com> hm <info@company.com>
```

  在此字段中指定文件后，可以在"**昵称**"对话框中选择作者作为提交消息字段的值。

- **用户字段配置文件**是一个简单的文本文件，由指定将作者作为值的提交消息字段的行组成，例如：

```
Initial-patch-by:
Reported-by:
Rubber-stamped-by:
Signed-off-by:
Tested-by:
```

在此字段中指定文件后，您可以在提交更改时将作者添加为提交邮件字段的值。如果还指定了**用户/别名配置文件**，则可以在"**昵称**"对话框中选择作者。

- **SSH 提示命令**指定一个 ssh-askpass 命令，在使用 SSH 时，您可以使用该命令（在 Linux 上）提示用户输入密码。例如，或，具体取决于您使用的 ssh-askpass 实现。`ssh-askpassx11-ssh-askpass`
- 例如，**重置VCS缓存**将版本控制系统配置重置为Qt设计工作室从命令行更改后已知的状态。

将 Git 用于视窗

如果将 Git 配置为仅与 一起使用，并使用 SSH 授权，则 Git 会在环境指向的目录中查找 SSH 密钥。该变量始终由 设置。`git bashHOMEgit bash`

但是，该变量通常不在 Windows 命令提示符中设置。当您从 Windows 命令提示符运行 Git 时，它会在其安装目录中查找 SSH 密钥，因此授权将失败。

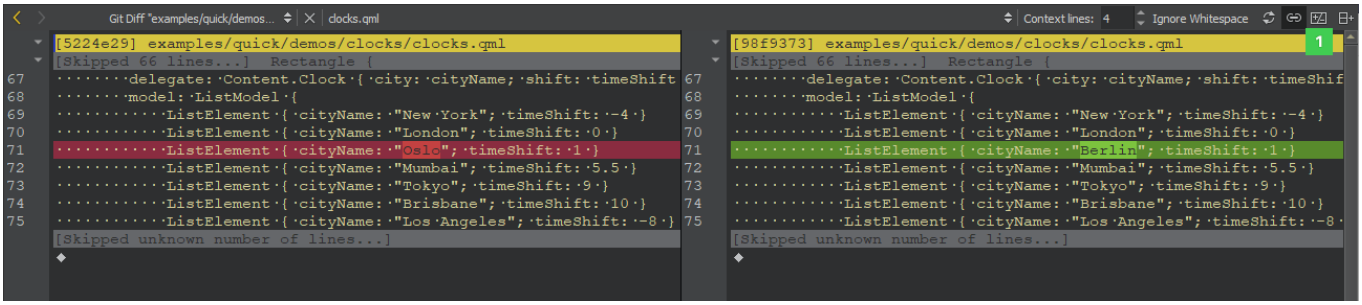
您可以从Qt设计工作室设置环境变量。选择"**编辑**>**首选项**"> Git >**版本控制**"，然后选中"**设置**"HOME"**环境变量**"复选框。 设置为当 Git 可执行文件运行时，并且授权的工作方式与 `.HOMEHOME%HOMEDRIVE%%HOMEPATH%git bash`

使用当前文件

若要使用当前文件，请选择"**工具**"> Git 中的命令>**当前文件**"。某些命令还可用于包含该文件的项目或本地存储库。

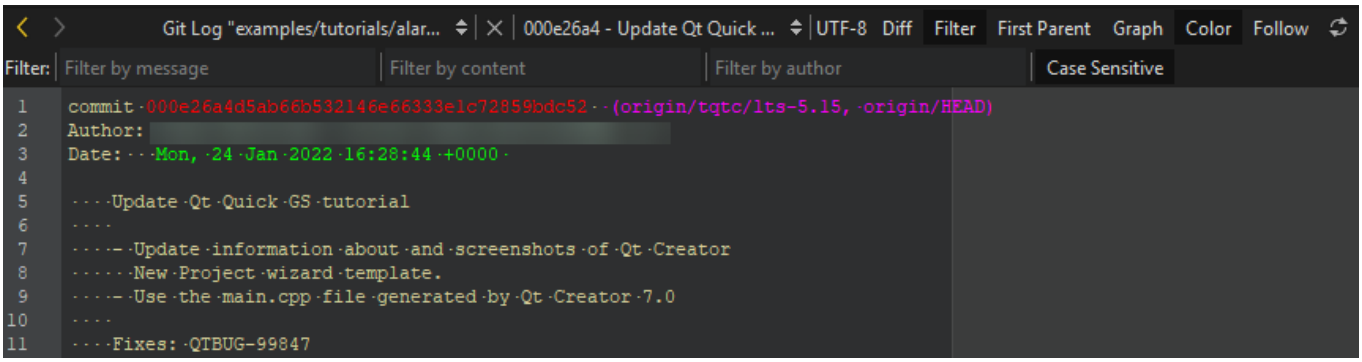
查看 Git 比较

You can *diff* the current file or project to compare it with the latest version stored in the repository and to display the differences. To display the diff in a read-only editor, select **Diff of**. If the file is accessible, you can double-click on a selected diff chunk and Qt Design Studio opens an editor displaying the file, scrolled to the line in question.



The diff is displayed side-by-side in a diff editor by default. To use the unified diff view instead, select the **Switch to Unified Diff Editor** (1) option from the toolbar. In both views, you can use context menu commands to apply, revert, stage, and unstage chunks or selected lines, as well as send chunks to a code pasting service.

Viewing Git Log



Right-clicking on a commit identifier brings up a context menu that lets you apply actions on the commit, such as view annotations or cherry-pick or revert a commit. For more information, see Working with Branches.

To toggle the diff view, select **Diff**. To use the patience algorithm for calculating the differences, select **Patience**. To only show text changes, select **Ignore Whitespace**.

To filter log entries by the text in the commit message, by strings that were added or removed, or by author, select **Filter**. Enter a search sting in the **Filter by message**, **Filter by content**, or **Filter by author** field. Select **Case Sensitive** to make filtering consider case.

To follow only the first parent on merge commits, select **First Parent**.

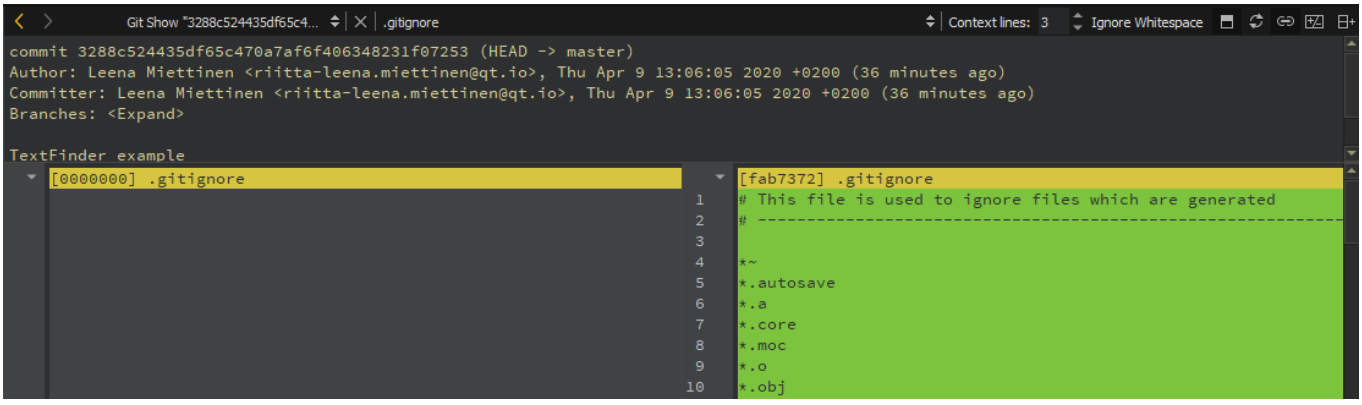
To toggle between textual and visual representation of the log, select **Graph**.

To toggle color coding of different parts of the log entries, select **Color**.

To show log also for previous names of the file, select **Follow**.

Select (**Reload**) to rescan the files.

To display a description of the change including the diff in the **Git Show** view, select **Describe Change** in the context menu.



Viewing Annotations

To view annotations, select **Blame**. The view displays the lines of the file prepended by the commit identifier they originate from. Clicking on the commit identifier shows a detailed description of the change.

To show the annotation of a previous version, right-click on the commit identifier and select **Blame Parent Revision**. This allows you to navigate through the history of the file and obtain previous versions of it.

The other actions in the context-menu enable you to apply actions to the commit, such as cherry-pick, checkout, or revert it.

Staging Changes

To mark a new or modified file for committing it to the repository, select **Stage File for Commit**. To undo this function, select **Unstage File from Commit**.

Resetting Changes

Git has an index that is used to stage changes. The index is committed on the next commit. Git allows you to revert back to the state of the last commit as well as to the state staged in the index.

To revert all changes and reset the current file to the state of the index, select **Undo Unstaged Changes**.

To return the current file to the state it was in right after the last commit, select **Undo Uncommitted Changes**. This reverts all changes, discarding the index.

Working with the Current Project

To work with the current project, select the commands in **Tools** > **Git** > **Current Project**. The **Diff Project** and **Log Project** commands, which are also available for the current file, are described above.

Cleaning Projects

To clean the working directory, select **Build Project** > **Clean**. All files that are not under version control are displayed in the **Clean Repository** dialog. Ignored files are deselected by default. Select the files to delete and click **Delete**.

Working with Local Repositories

To work with the local repository, select the commands in **Tools** > **Git** > **Local Repository**. The commands that are also available for the current file or project are described above.

Viewing Reference Log

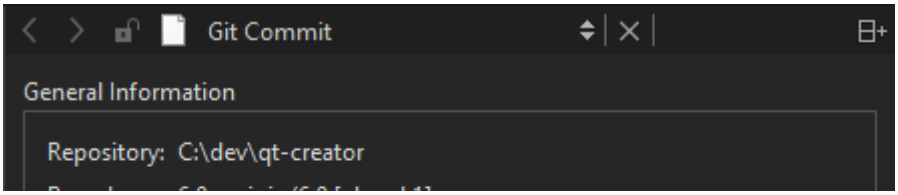
Reference logs record when the tips of branches and other references were updated in the local repository. To view the reference log, select **Reflog**.

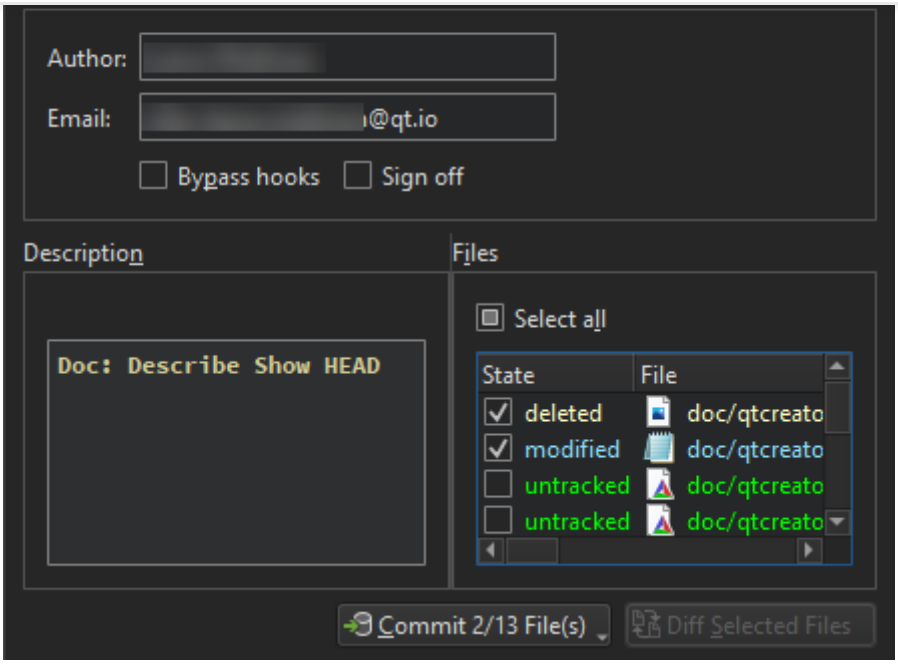
Viewing Git Status

To view the status of the repository in **Version Control**, select **Status**. The context menu contains additional actions, such as selecting and clearing all entries, copying text, and opening files.

Committing Changes to Git

To submit your changes to Git, select **Commit**. Qt Design Studio displays a commit page containing a text editor where you can enter your commit message and a checkable list of modified files to be included.





General Information displays the names of the repository and branch for the commit.

In **Commit Information**, you can edit information about the author of the commit. To bypass re-commit and commit message hooks, select **Bypass hooks**. If signoffs are used for your project, select **Sign off** to add a *signed-off-by* trailer by the author at the end of the commit log message.

In **Description**, edit the commit message.

In **Files**, select the files to include in the commit.

When you have finished filling out the commit page information, click on **Commit** to start committing.

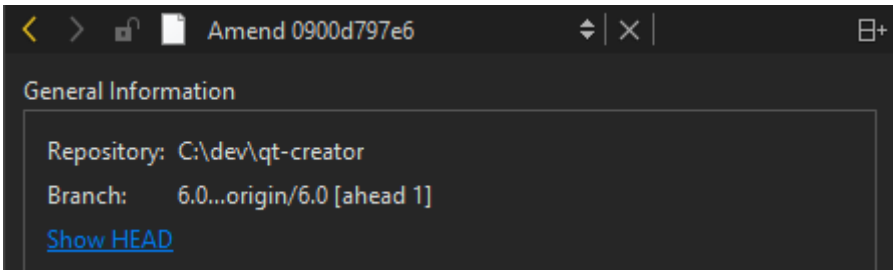
The **Diff Selected Files** button opens a diff view of the files selected in the file list. Select **Stage Chunk** in the context menu to stage a chunk or **Stage Selection** to stage the selected lines.

To unstage chunks or selected lines, select **Unstage Chunk** or **Unstage Selection** in the context menu. To revert the changes in a chunk, select **Revert Chunk**.

The commit page is just another editor, and therefore you return to it when you close the diff view. You can also switch to an open diff view by selecting it in the **Open Documents** view in the sidebar.

Amending Commits

To apply latest changes to the last commit, select **Tools** > **Git** > **Local Repository** > **Amend Last Commit**.



To view the commit in its current form, before amending, select **Show HEAD**.

To view a diff of the changes in the selected files, select **Diff Selected Files**.

Select **Commit** to amend the commit.

Resetting Local Repository

To reset changes, select **Reset**. This opens a dialog where you can select the commit to reset the working directory to. This is useful after applying patches for review, for example. You can choose between a **Soft** reset that does not touch the index file nor the working tree at all, a **Hard** reset that discards all changes to tracked files in the working tree, and a **Mixed** reset that resets HEAD and the index (nothing remains staged) without touching the working directory.

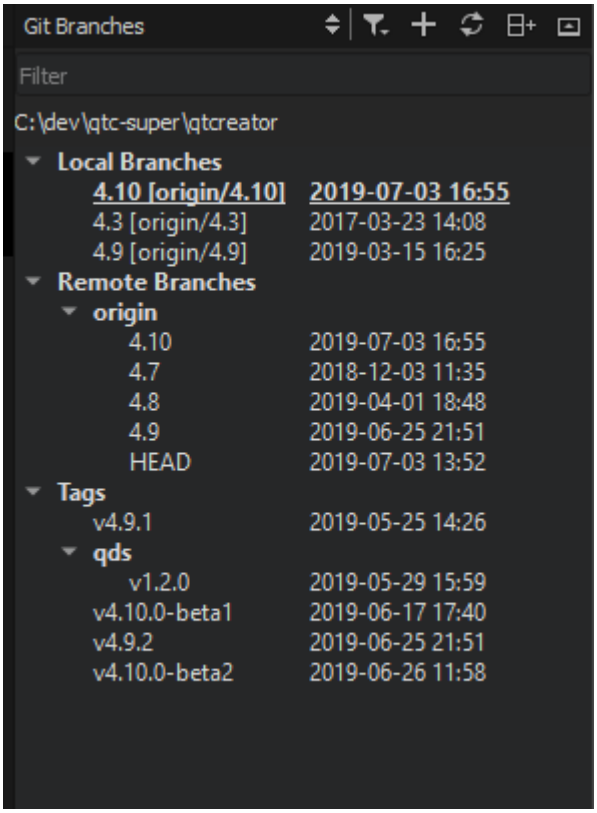
To recover removed files, select **Recover Deleted Files**.

To change a series of commits in the local repository, select **Interactive Rebase**. You can reorder or discard commits, squash them into a single commit, or edit the commit messages.

If you already pushed the local changes to a remote, Qt Design Studio refuses the interactive rebase because the local commits are missing. To start the interactive rebase from the change log, select **Branches** > **Log**. Select the change, and then select > **Interactive Rebase from Change** in the context menu.

Working with Branches

To work with Git branches, select **Branches**. The checked out branch is shown in bold and underlined in the list of branches in the **Git Branches** sidebar view.



Old entries and tags are filtered out of the list of branches by default. To include them, select (**Filter Tree**), and then select **Include Old Entries** and **Include Tags**.

To add a tag to a change in the change log, select **Branches** > **Log**. Select the change, and then select **Add Tag for Change** in the context menu.

If you checked out a specific commit, the list of branches displays a *Detached HEAD* entry.

For local and remote branches, the changes log can be shown by double clicking on the branch name.

The following operations are supported in the context-menu for a branch:

Menu Item	Description
Add	Create new tracking and non-tracking branches.
Remove	Remove a local branch. You cannot delete remote branches.
Rename	Rename a local branch or a tag. You cannot rename remote branches.
Checkout	Check out the selected branch and make it current. You can stash changes you have made to tracked files.
Diff	Show the differences between the selected and the current branch.
Log	Show the changes in a branch.
Reset	Reset the active branch to the selected branch. You can choose between a **Hard**, **Mixed**, and **Soft** reset. For more information, see Resetting Local Repository.
Merge	Join the development histories in two branches together. If the commit you are merging can be reached by following the first commit's history, there is no divergent work to merge together. To allow Git to move the branch pointer forward, select **Merge (Fast-Forward)**. If you do not want to fast-forward the branch, select **Merge (No Fast-Forward)**.
Rebase	Copy local commits to the updated upstream head.
Cherry Pick	Cherry pick the top commit from the selected branch.
Track	Set the current branch to track the selected one.
Push	Push the committed changes to the selected remote branch.

The following additional context-menu operations are available for **Remote Branches**. The context-menu can be opened on **Remote Branches** or on a specific remote repository.

Menu Item	Description
Fetch	Fetch all the branches and changes information from a specific remote repository, or from all remotes if applied to **Remote Branches**.
Manage Remotes	Open the **Remotes** dialog.

Configuring Merge Tools

Only graphical merge tools are supported. You can configure the merge tool to use on the command line. For example, to use the KDiff3 merge tool, enter the following command:

```
git config --global merge.tool kdiff3
```

Applying Patches

Patches are rewriting instructions that can be applied to a set of files. To apply a patch file that is open in Qt Design

Using Stashes

With Git, you can put your current set of changes onto a virtual shelf called a *stash*. Stashes are useful, for example, to put aside a set of changes to work on higher priority tasks or to pull in new chages from another repository.

To stash all local changes, select **Stash** > **Stash**. The working copy is reset to the state it had after the last commit. To save the current state of your unstaged files and reset the repository to its staged state, select **Stash Unstaged Files**.

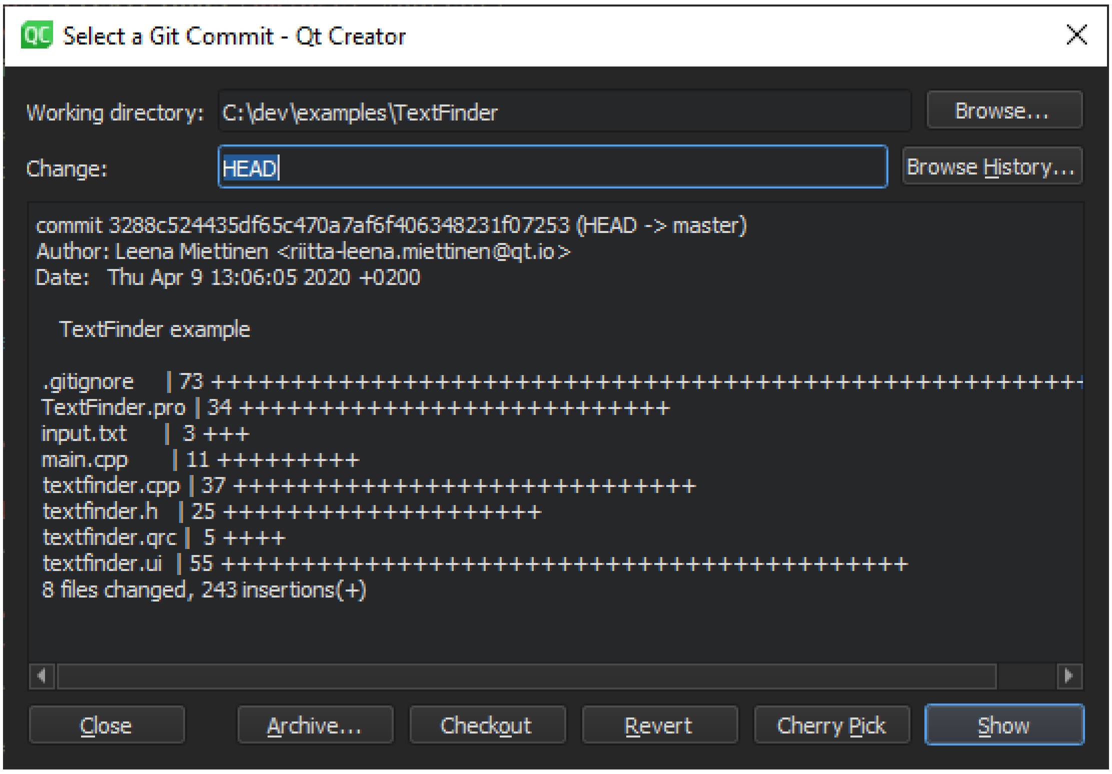
To display a dialog that shows all known stashes with options to restore, display or delete them, select **Stashes**.

To save a snapshot of your current work under a name for later reference, select **Take Snapshot**. The working copy is unchanged. For example, if you want to try something and find out later that it does not work, you can discard the changes and return to the state of the snapshot.

To remove a single stashed state from the stash list and apply it on top of the current working tree state, select **Stash Pop**.

Applying Actions to Commits

To browse a directory or the commit history and to apply actions on the commits, select **Tools** > **Git** > **Actions on Commits**.



You can apply the following actions on commits:

Menu Item	Description
Archive	Package the commit as a ZIP or tarball.
Checkout	Check out the change in a headless state.
Cherry Pick	Cherry-pick the selected change to the local repository.

Menu Item Description

Initializing Git Repositories

To start controlling a project directory that is currently not under version control, select **Tools** > **Git** > **Create Repository**. Qt Design Studio creates a new subdirectory named .git that contains all the necessary repository files. However, nothing in the project is tracked yet, so you will need to create an initial commit to start tracking the project files.

Working with Remote Repositories

To work with remote repositories, select the commands in **Tools** > **Git** > **Remote Repository**.

To fetch all the branches and change information from a remote repository, select **Fetch**.

To pull changes from the remote repository, select **Pull**. If there are locally modified files, you are prompted to stash the changes. Select **Edit** > **Preferences** > **Version Control** > **Git** and then select the **Pull with rebase** check box to perform a rebase operation while pulling.

To push committed changes to the remote repository, select **Push**. If the local branch does not have an upstream branch in the remote repository, Qt Design Studio prompts you to create it and set it as upstream.

Managing Remote Repositories

To manage remote repositories available in Git, select **Tools** > **Git** > **Remote Repository** > **Manage Remotes**. Double-click the names and URLs of the remote repositories to edit them.

The following operations are supported:

Menu Item	Description
Refresh	Refresh the list of remote repositories.
Add	Add a new remote repository.
Fetch	Fetch all the branches and change information from a remote repository.
Push	Push committed changes to the remote repository.
Remove	Remove a remote repository.

Using Git with Subversion

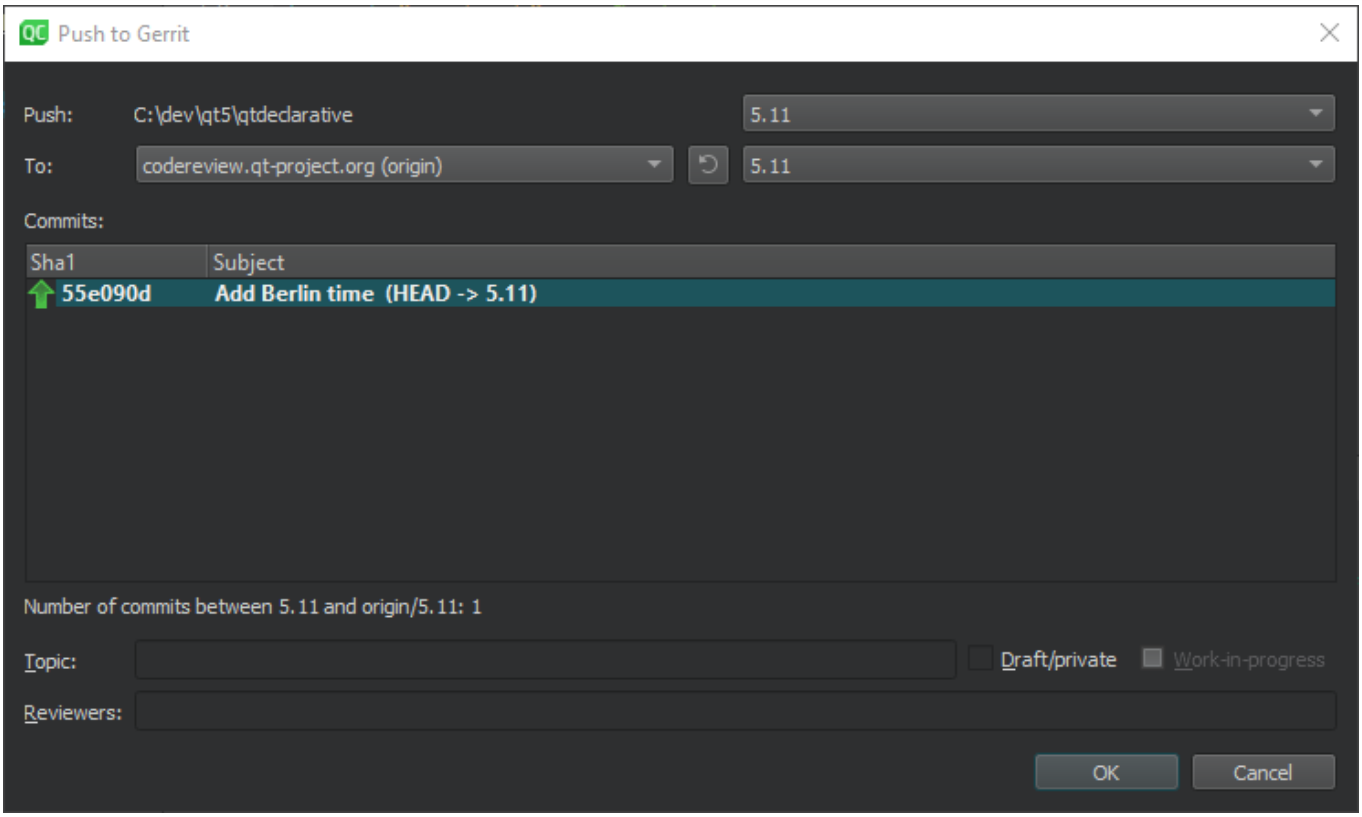
You can use Git as a client for a Subversion server. To fetch changes from a Subversion repository to a Git repository, select **Tools** > **Git** > **Remote Repository** > **Subversion** > **Fetch**.

To view the Git Subversion log, select **Log**.

To publish the commits from the current branch to the Subversion server, select **DCommit**. This will create a revision in Subversion for every local Git commit. Afterwards, the branch is rebased or reset (depending on whether or not there is a diff between Subversion and head).

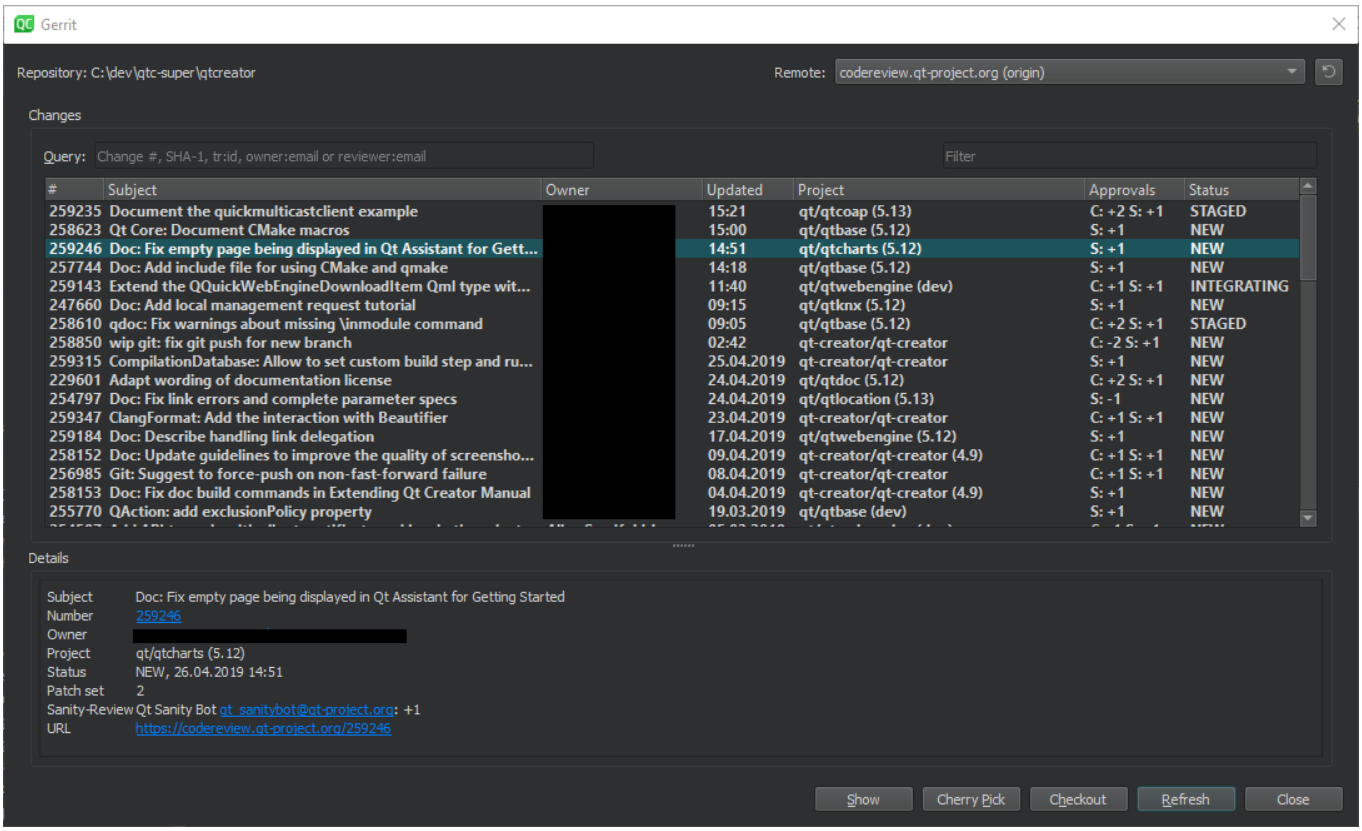
Reviewing Code with Gerrit

If your Git project uses Gerrit for code reviews, you can view your changes in Qt Design Studio.



Select the **Draft/private** check box to push changes that are only visible to you and the reviewers. If you are using Gerrit 2.15 or later, you can select the **Work-in-progress** check box to push changes that do not generate email notifications.

To view the same information about each change as in the Gerrit web interface, select **Tools** > **Git** > **Remote Repository** > **Gerrit**.



To view details of the selected change, select **Show**.

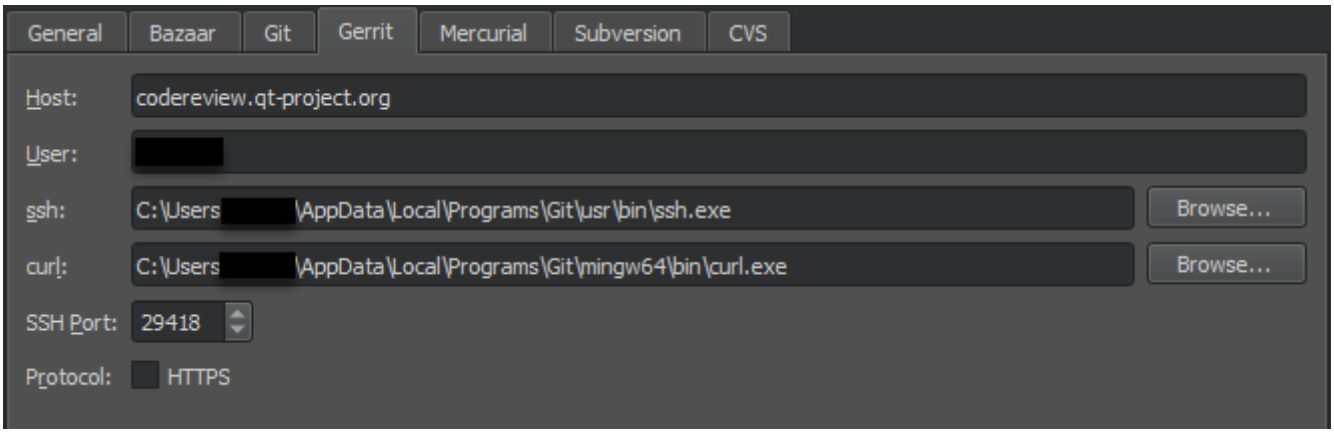
directory to, and then select **OK**.

To check out the change in a headless state, select **Checkout**.

To refresh the list of changes, select **Refresh**.

The **Remote** field lists the remotes of the current repository that are detected as Gerrit servers. Select **Edit** > **Preferences** > **Version Control** > **Gerrit** to specify a fallback connection to a Gerrit server over SSH. The Gerrit REST interface and the curl tool are used for HTTP connections.

Select the **HTTPS** check box to prepend to the Gerrit URL if Gerrit does not provide it.`https`



Working with Git Tools

To start a graphical interface to Git, select **Tools** > **Git** > **Git Tools** > **Git Gui**.

> **Note:** On macOS, the default Git installation does not contain Git Gui. To use Git Gui, install it separately. To start Git Gui from Qt Design Studio, select **Preferences** > **Version Control** > **Git**, and set the path to the environment that contains Git Gui in the **Prepend to PATH** field.

To start the commit viewer for Git, select **Tools** > **Git** > **Git Tools** > **Gitk**. You can also start the tool to view commits in the current document or in the folder that contains the current document. To specify arguments for running Gitk, select **Edit** > **Preferences** > **Version Control** > **Git**.

To use some other application for viewing Git history, such as GitX or QGit viewer, select **Edit** > **Preferences** > **Version Control** > **Git** and specify the path to the application executable in the **Command** field. To start the application, select **Tools** > **Git** > **Git Tools** > **Repository Browser**.

To start , select **Tools** > **Git** > **Git Tools** > **Git Bash**.`git bash`

To resolve merge conflicts, select **Tools** > **Git** > **Git Tools** > **Merge Tool**. This menu item is visible only when you have merge conflicts to resolve.

‹ Developer Topics

Converting Qt 5 Projects into Qt 6 Projects ›

Qt DOCUMENTATION

Contact Us

Company

About Us
Investors
Newsroom
Careers
Office Locations

Licensing

Terms & Conditions
Open Source
FAQ

Support

Support Services
Professional Services
Partners
Training

For Customers

Support Center
Downloads
Qt Login
Contact Us
Customer Success

Community

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In