

Qt 创建者手册 > [使用叮当工具](#)

使用叮当工具

Qt Creator集成了以下Clang工具，通过使用静态分析来查找C，C++和客观C源代码中的问题：

- › **Clang-Tidy**，为典型的编程错误（如样式违规或接口误用）提供诊断和修复。
- › **克莱兹**，这有助于克隆理解Qt语义。它显示与Qt相关的编译器警告，从不必要的内存分配到滥用API，并提供用于修复某些问题的重构操作。


注意： 叮叮当静态分析器检查是叮叮当的一部分。要使用检查，您必须为叮当工具创建自定义配置，并为叮叮当当工具启用它们。

Clang工具是与Qt Creator一起交付和安装的，因此您不需要单独设置它们。

除了运行工具以收集诊断信息外，还可以选择从使用该选项导出的 **YAML** 文件中加载诊断信息。-export fixes

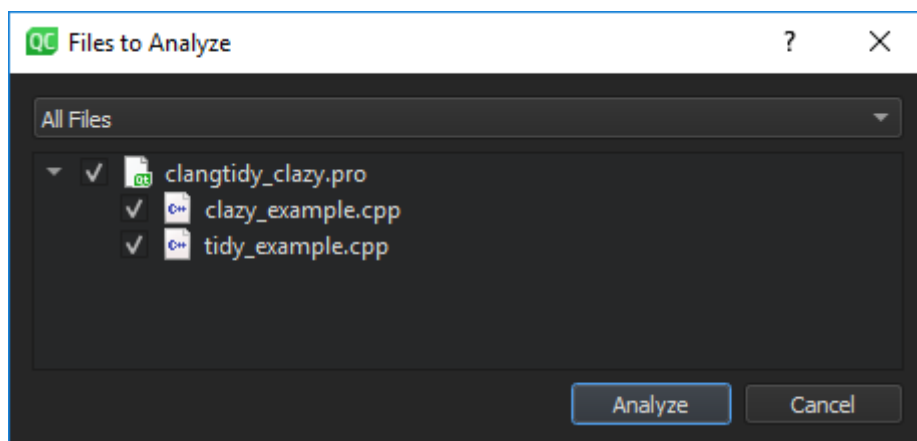
运行叮当工具

要运行 Clang 工具来分析当前打开的文件，请执行以下操作：

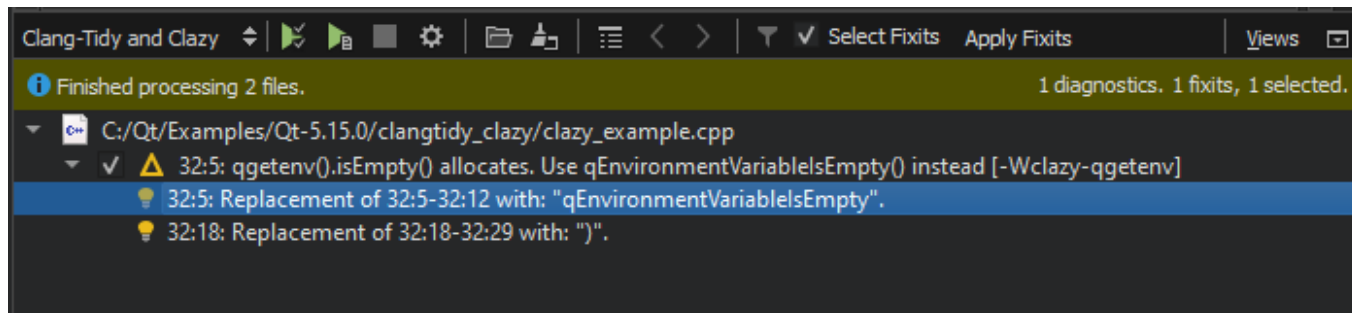
- › 选择编辑器工具栏上的  (**分析文件**) 按钮。
- › 选择“**工具**”>C++>“**分析当前文件**”。

要运行 Clang 工具来分析打开的项目，请执行以下操作：

1. 选择“**分析**”>“**咯咯-整洁**”和“**克拉齐**”。



发现的问题显示在“叮当当”和“克拉齐”视图中：



注意：如果在模式选择器中选择“**调试**”以打开“**调试**”模式，然后选择“Clang-Tidy”和“Clazy”，则必须选择“**（开始）**”按钮以打开“**要分析的文件**”对话框。

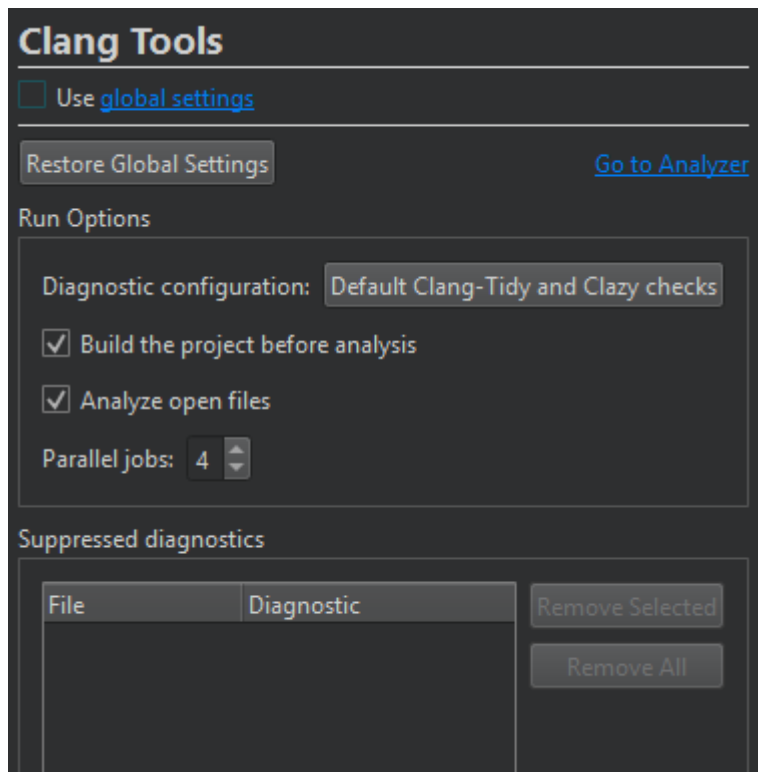
双击某个问题可移动到该问题在代码编辑器中出现的位置。

如果某个问题存在 fixit，则可以选中该问题旁边的复选框以安排其修复。选中“**选择修复程序**”复选框以选择所有修复程序。通过将鼠标指针悬停在复选框旁边的图标上，可以查看问题的状态。

若要查看有关用🔦图标标记的问题的详细信息，请将鼠标指针悬停在该行上。

您可以通过在上下文菜单中选择“禁用此检查”或“**禁用这些检查**”来全局或为特定项目禁用特定类型的检查。

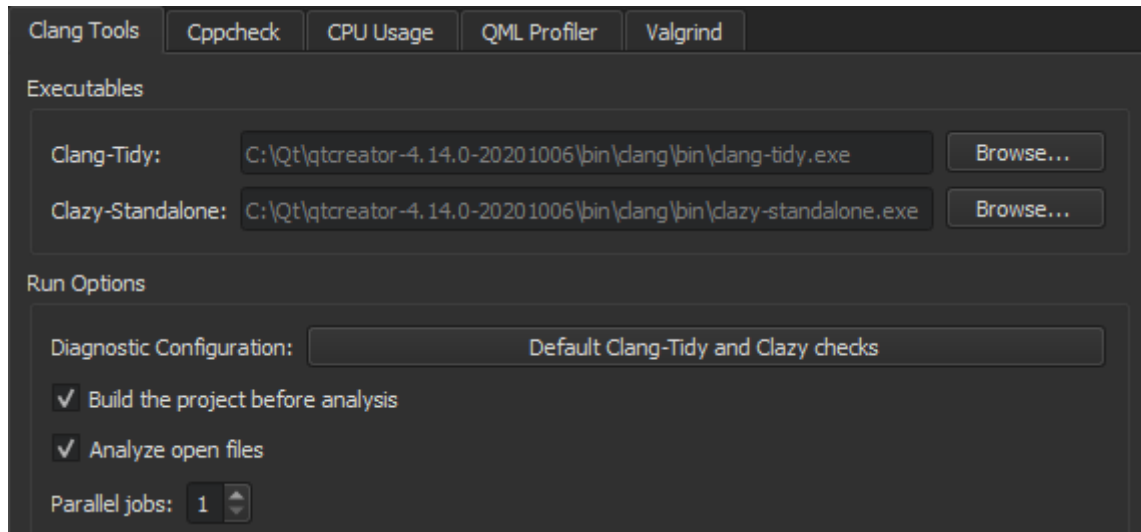
选择该⚙️按钮可为当前项目自定义 Clang 诊断。



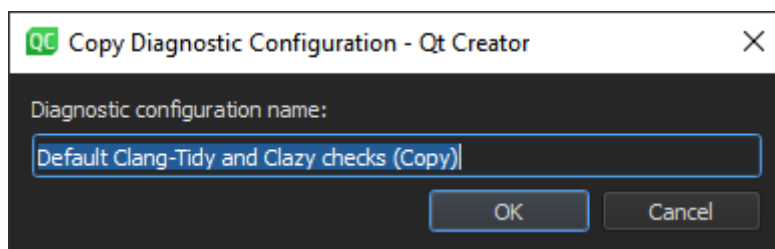
To restore the global settings, select **Restore Global Settings**. To view and modify the global settings, select the link in **Use global settings**. To open the Clang static analyzer, select **Go to Analyzer**.

To configure Clang diagnostics globally for Clang tools:

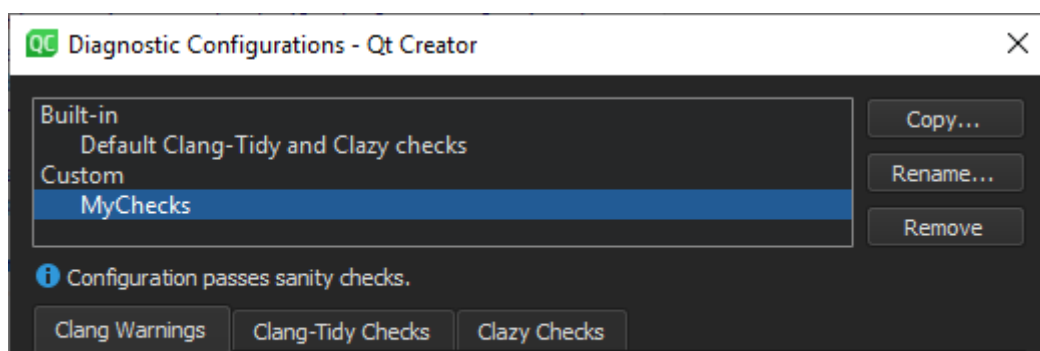
1. Select **Edit > Preferences > Analyzer > Clang Tools**.

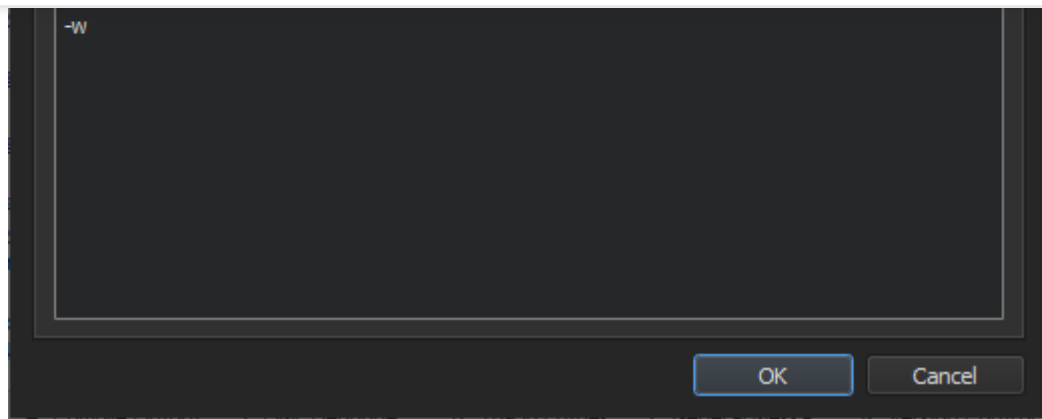


2. In the **Clang-Tidy** and **Clazy-Standalone** fields, set the paths to the executables to use.
3. To build the project before running the Clang tools, select the **Build the project before analysis** check box. The Clang tools do not require the project to be built before analysis, but they might display misleading warnings about files missing that are generated during the build. For big projects, not building the project might save some time.
4. To disable automatic analysis of open documents, deselect the **Analyze open files** check box.
5. In the **Parallel jobs** field, select the number of jobs to run in parallel to make the analysis faster on multi-core processors.
6. In the **Diagnostic Configuration** group, select **Manage** to create or edit a custom configuration.
7. Select **Copy** to create a custom Clang configuration.

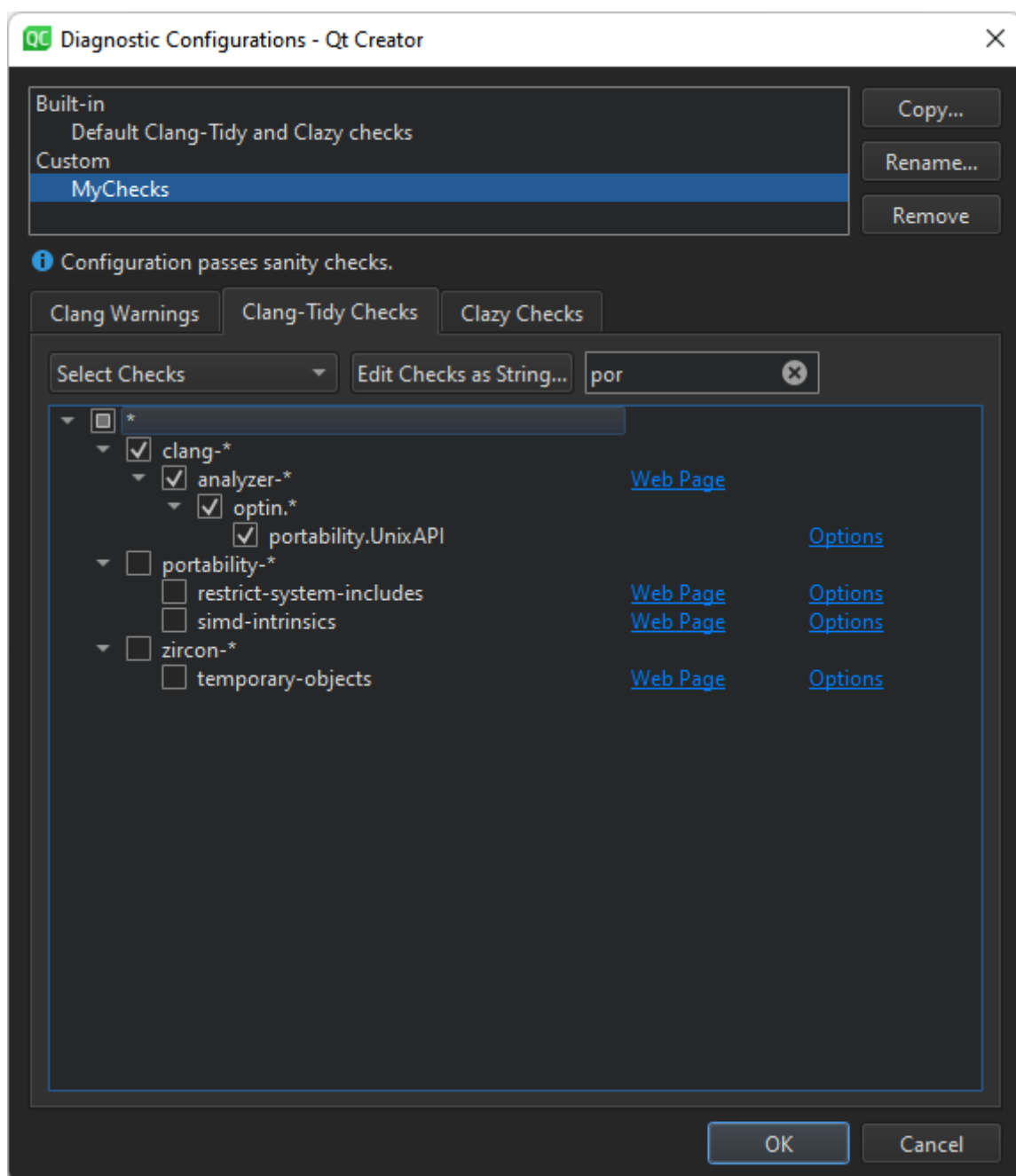


8. In the **Diagnostic configuration name** field, give the configuration a name, and then select **OK**.
9. In the **Clang Warnings** tab, select the **Use diagnostic flags from the build system** check box to forward diagnostic flags, such as warning flags, from the build system to the Clang code model for displaying annotations in the code editor.



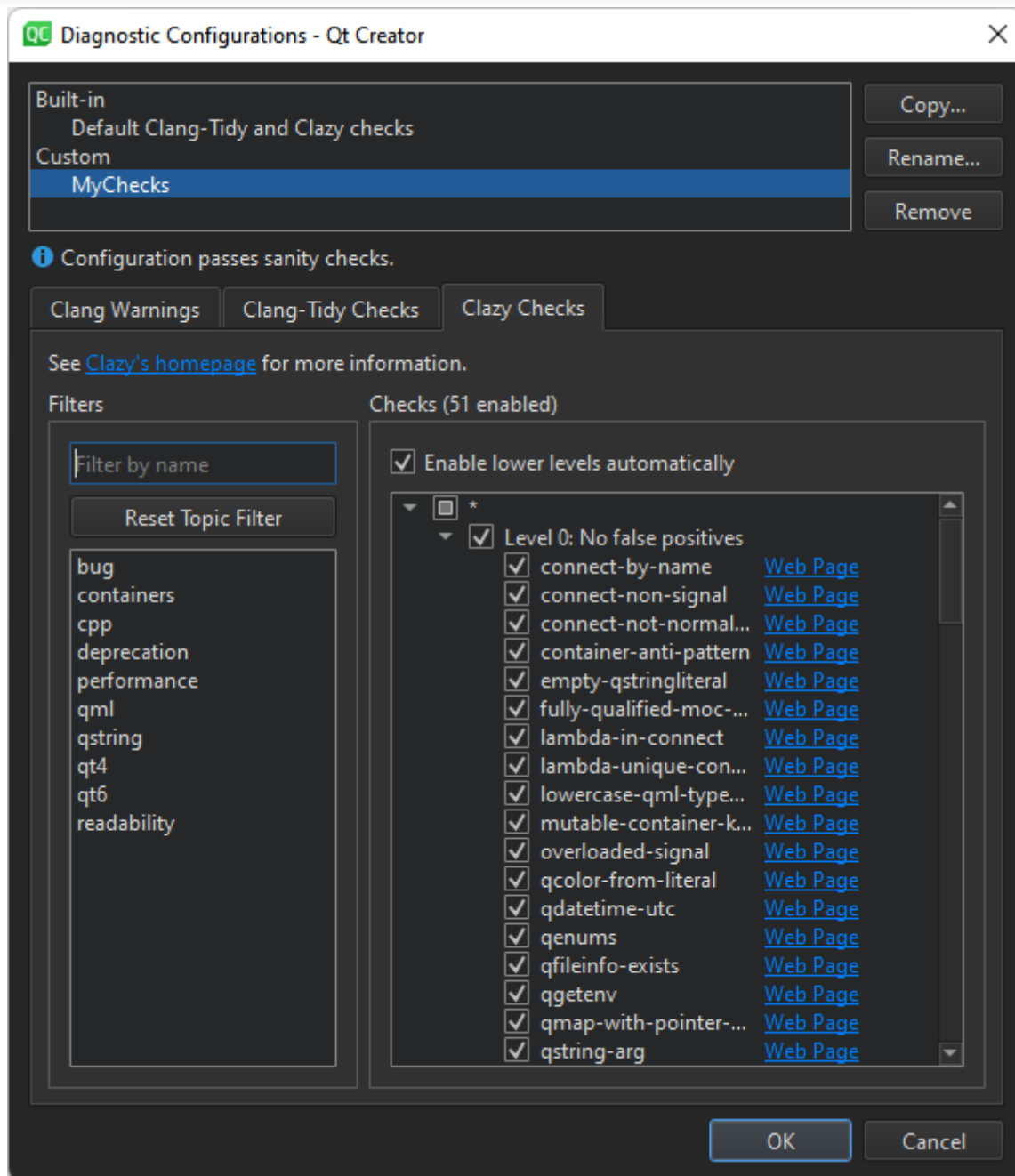


10. In the **Clang-Tidy Checks** tab, select **Select Checks** to select the checks to perform. To filter the checks, enter a string in the **Filter by name** field.



For more information about the available checkers, see [Clang Static Analyzer documentation](#).

11. To edit the selected check as plain text, select **Edit Checks as String**.



13. In the **Filters** field, select topics to view only checks related to those areas in the **Checks** field. To filter the checks in the selected areas, enter a string in the **Filter by name** field.

14. To view all checks again, select **Reset Filter**.

15. To view more information about the checks online, select the **Web Page** links next to them.

To suppress diagnostics, select **Suppress This Diagnostic** in the context menu. To view the suppression list for a project and to remove diagnostics from it, select **Projects > Project Settings > Clang Tools**.

Selecting Clazy Check Levels

The Clazy checks are divided into levels from 0 to 3. The checks at level 0 are very stable and provide hardly any false positives, while the checks at level 3 can be considered experimental. You can select the checks to perform at each level. To include the checks from the lower levels automatically, select the **Enable lower levels automatically** check box.

Creating Clang-Tidy Configuration Files

option takes precedence. The effective configuration can be inspected using `.-dump-config`

Qt Creator creates the configuration for you based on the checks you select. To store the checks in file format, you can create a `.clang-tidy` file, as follows:

1. Select **Edit Checks as String** and copy the contents of the field.
2. Pipe the output of into a file named `.`. For example: `clang-tidy -dump-config clang-tidy -checks=-*,bugprone-*,cppcoreguidelines-avoid-* -dump-config > .clang-tidy`
3. Move the `.clang-tidy` file to the parent directory of the sources.

To add more checks using Qt Creator later on, copy the checks from your `.clang-tidy` file into the **Edit Checks as String** field, select additional checks, and copy-paste the contents of the field to the `.clang-tidy` file.

[◀ Running Valgrind Tools on External Applications](#)

[Detecting Memory Leaks with Heob >](#)

© 2022 The Qt Company Ltd. Documentation contributions included herein are the copyrights of their respective owners. The documentation provided herein is licensed under the terms of the [GNU Free Documentation License version 1.3](#) as published by the Free Software Foundation. Qt and respective logos are trademarks of The Qt Company Ltd in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.



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



- Contribute to Qt
- Forum
- Wiki
- Downloads
- Marketplace

© 2022 The Qt Company

[Feedback](#) [Sign In](#)