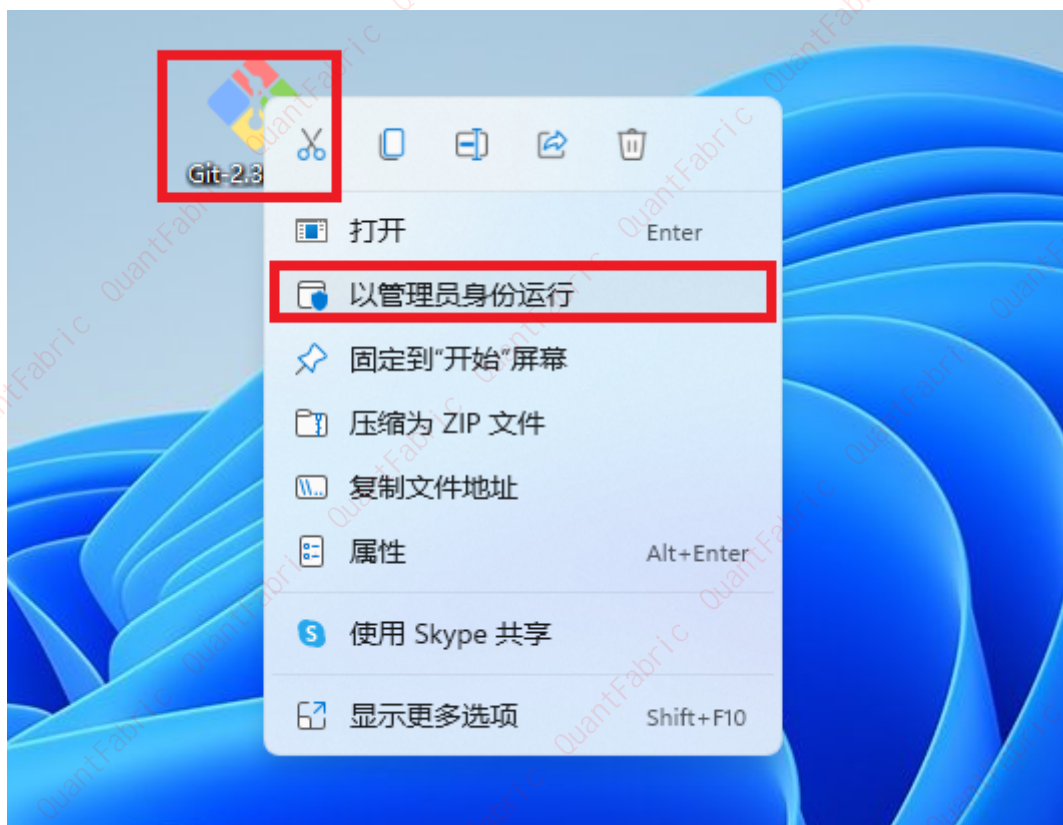


QuantFabric——开发环境搭建

一、GitBash

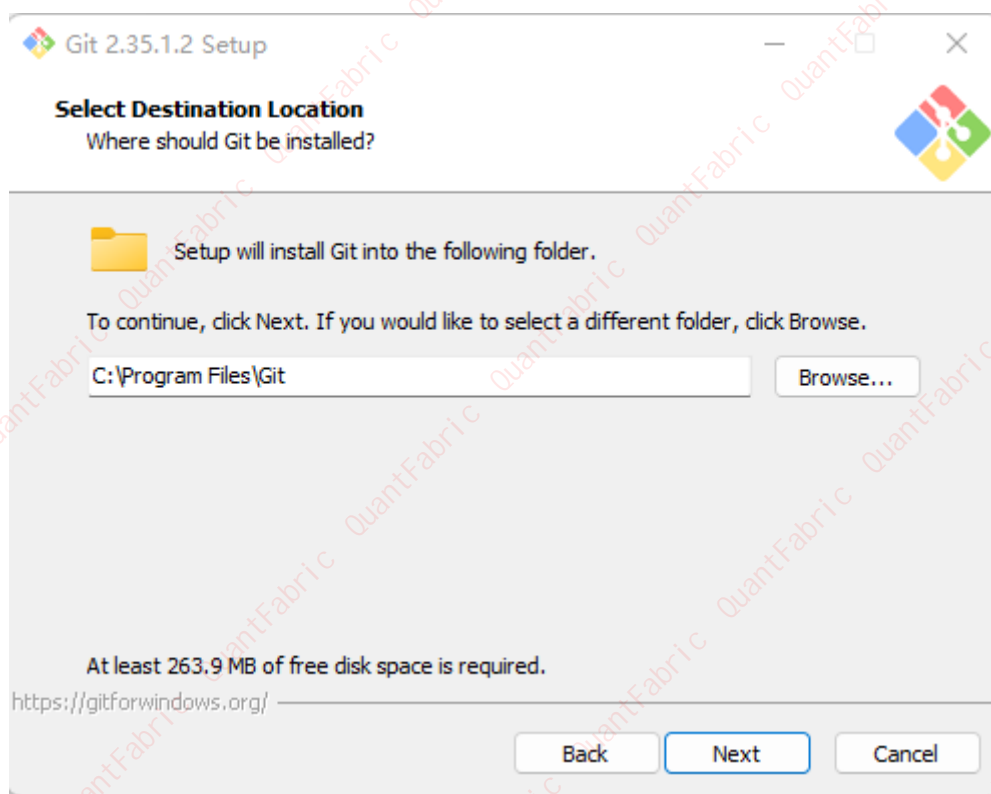
- Git客户端提供了GUI客户端模块和GitBash模块。
- Git客户端下载地址：[git-scm](https://git-scm.com)
- git-bash安装流程如下：
 - 鼠标右键点击exe安装程序，使用管理员安装程序。



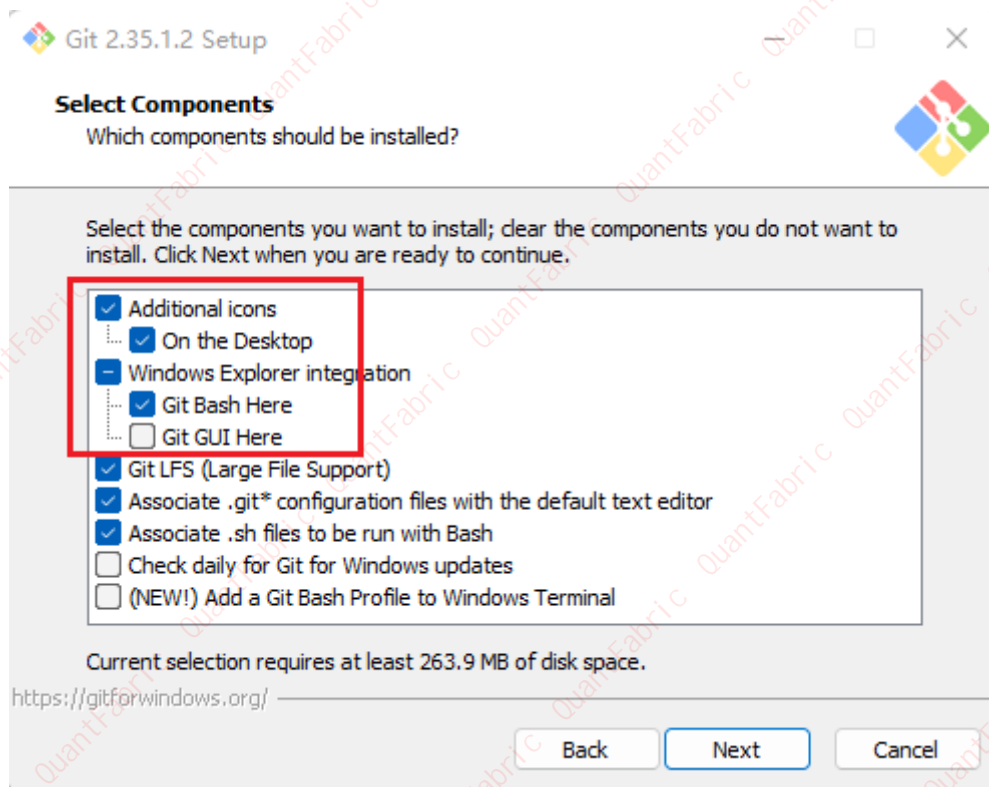
- 选择下一步：



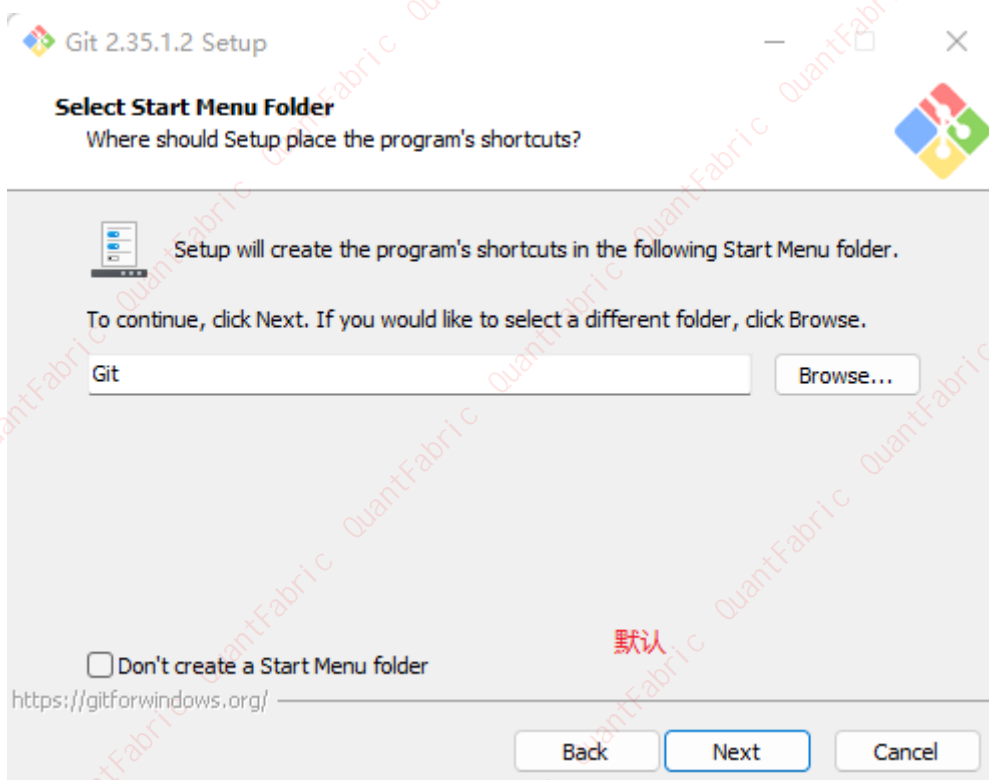
- 选择安装路径:



- 选择安装组件:



- 选择开始菜单设置:



- 设置gitbash编辑器:

Choosing the default editor used by Git

Which editor would you like Git to use?

Use Vim (the ubiquitous text editor) as Git's default editor

The [Vim editor](#), while powerful, [can be hard to use](#). Its user interface is unintuitive and its key bindings are awkward.

Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.

Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.

<https://gitforwindows.org/>

Back

Next

Cancel

- git版本库初始化分支设置:

Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

☒ Let Git decide

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

☐ Override the default branch name for new repositories

NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

main

This setting does not affect existing repositories.

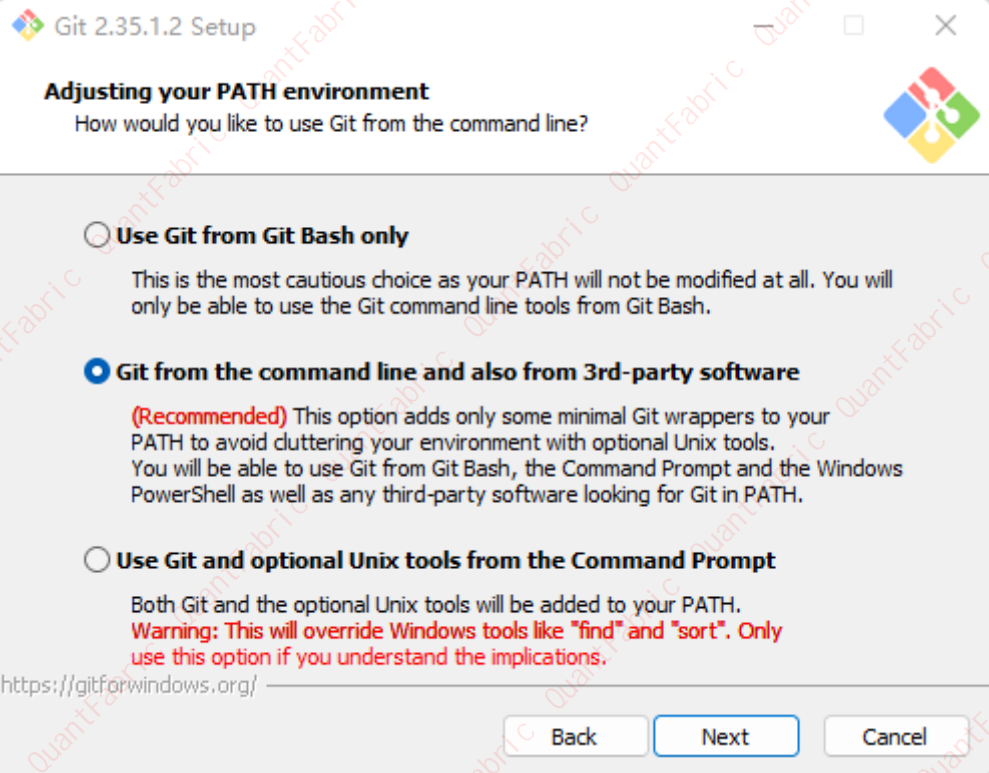
<https://gitforwindows.org/>

Back

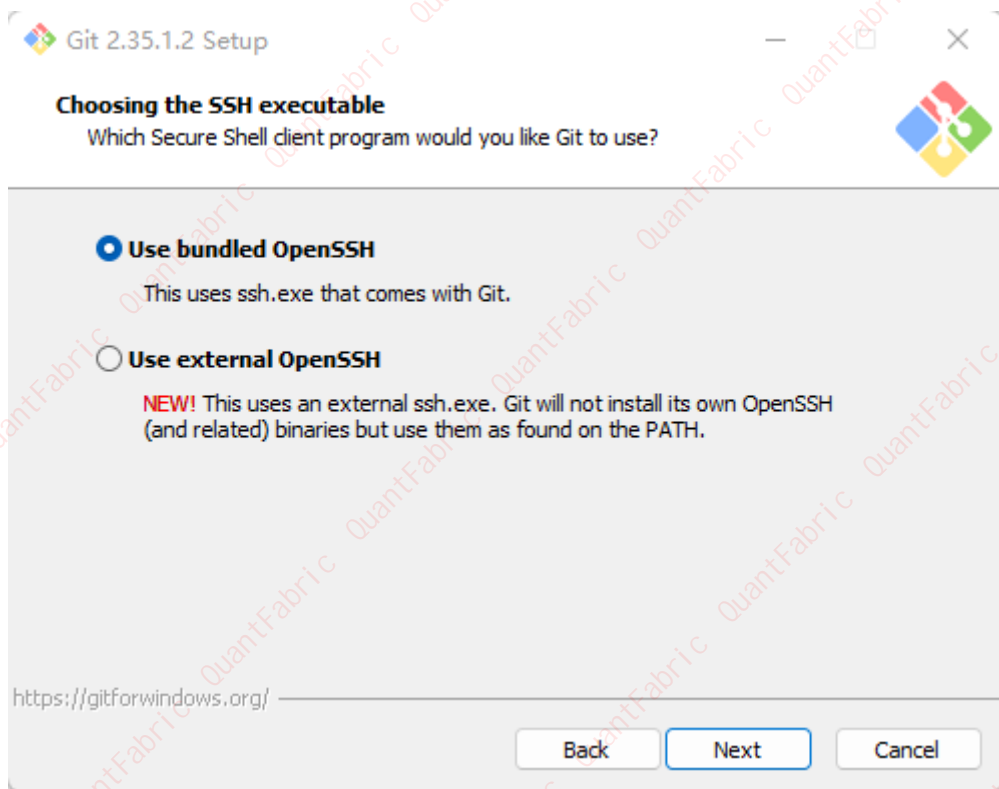
Next

Cancel

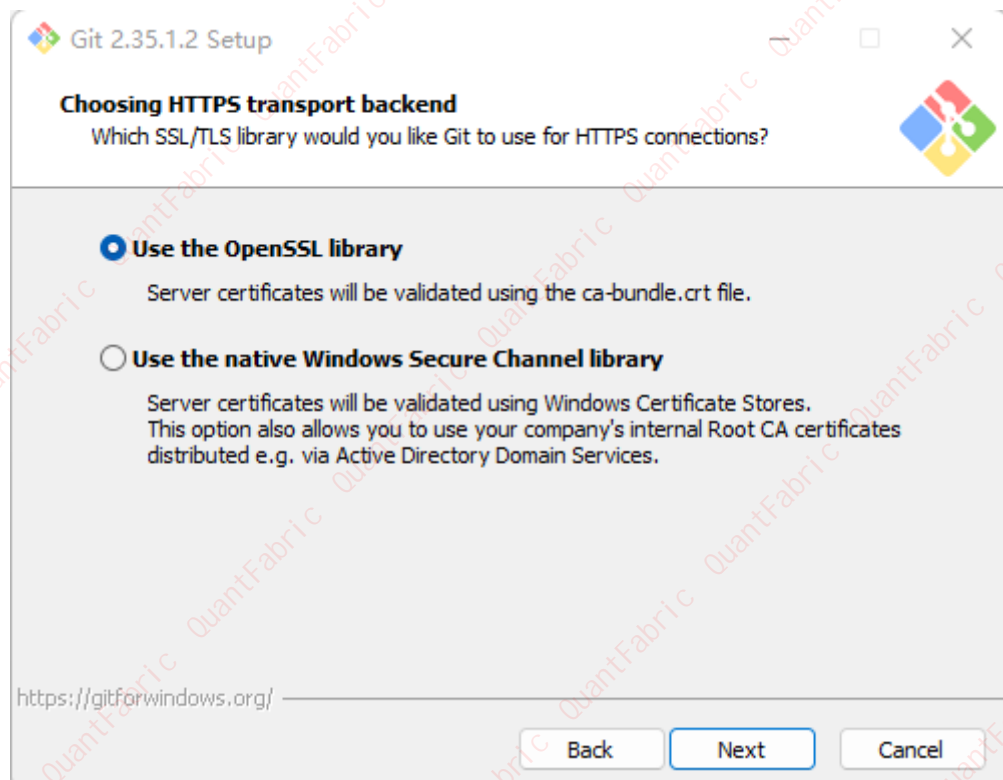
- git环境变量设置:



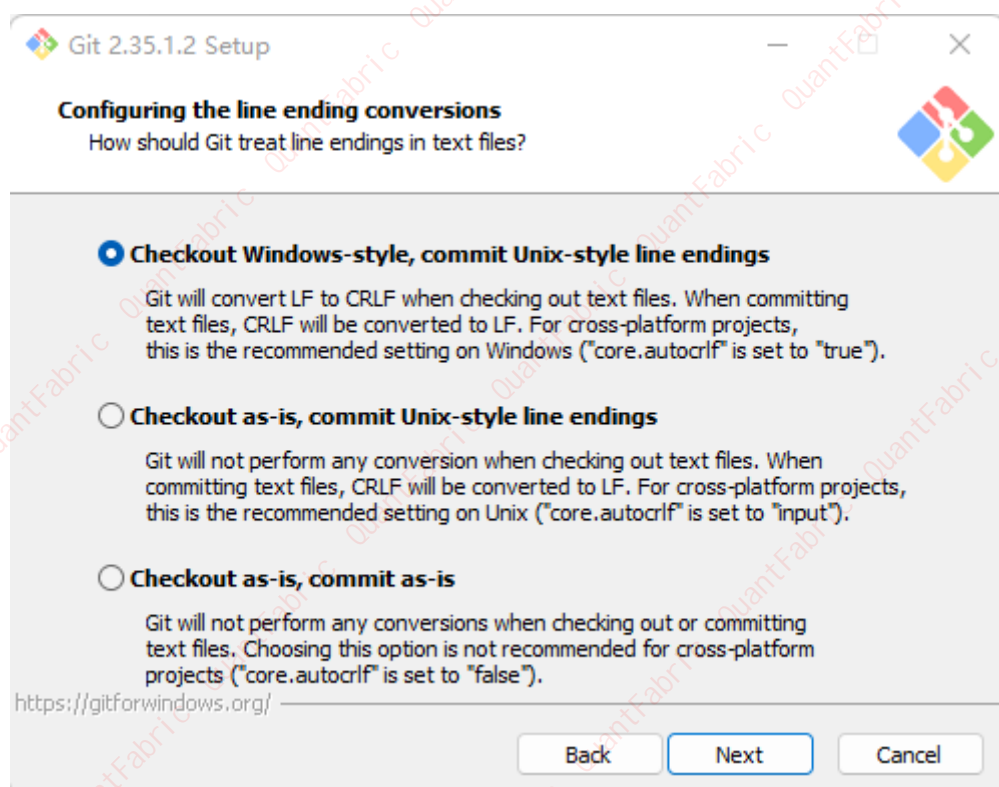
- 选择gitbash的ssh工具:



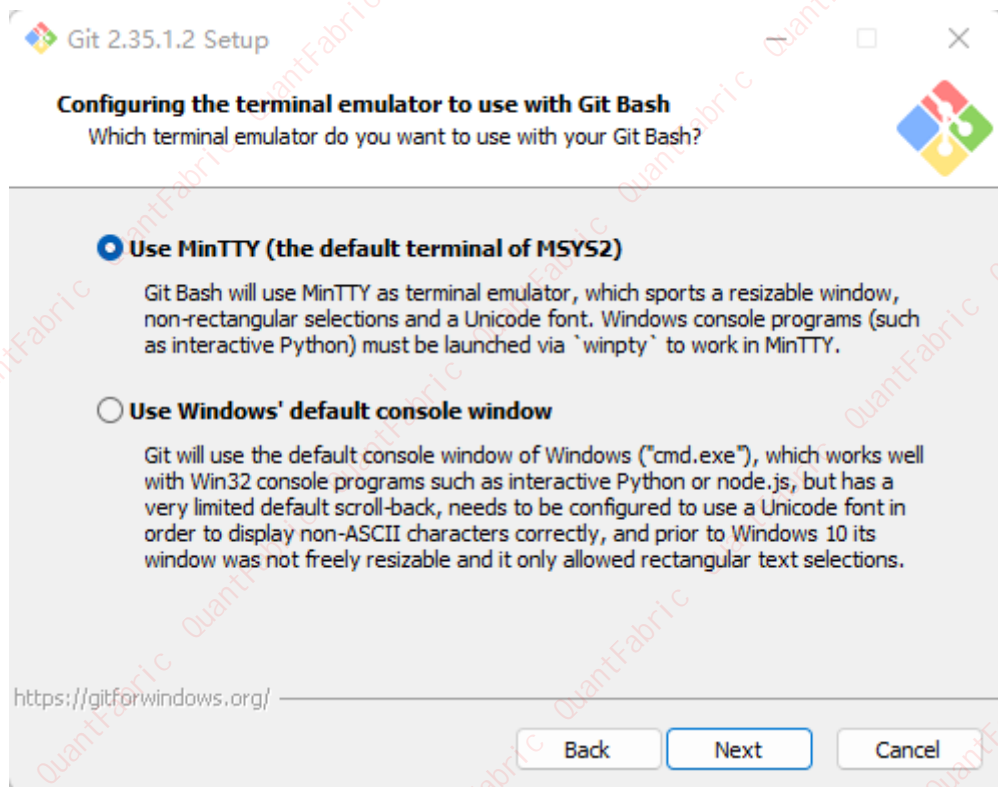
- 选择SSL库:



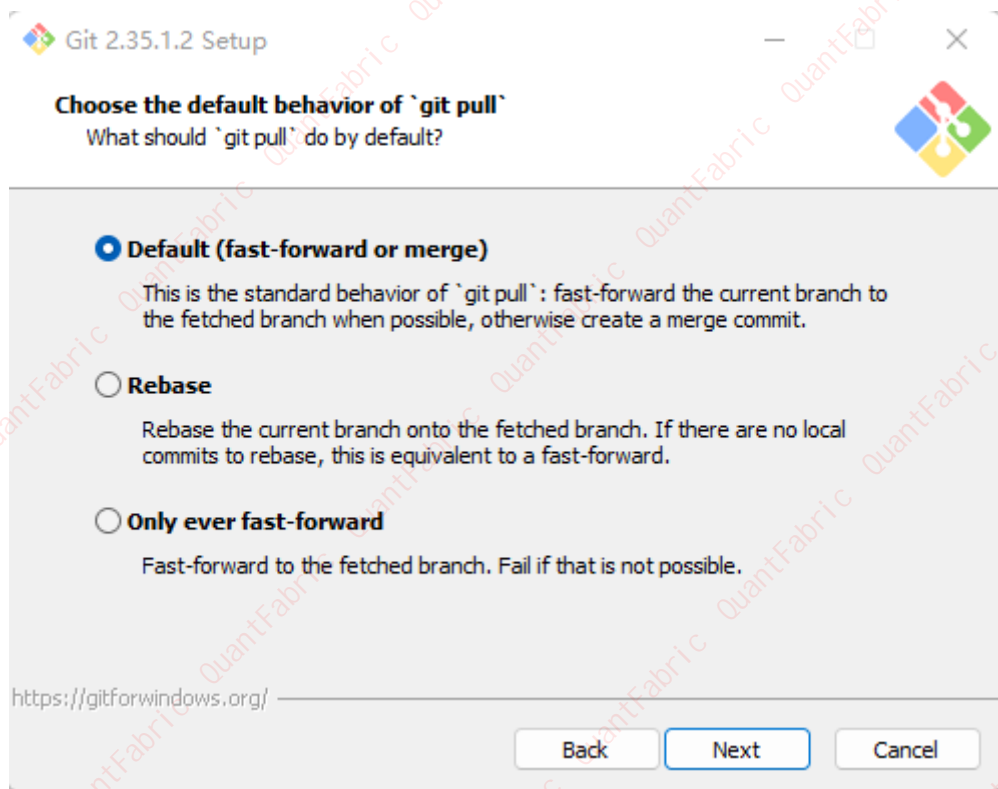
- 选择行结束符:



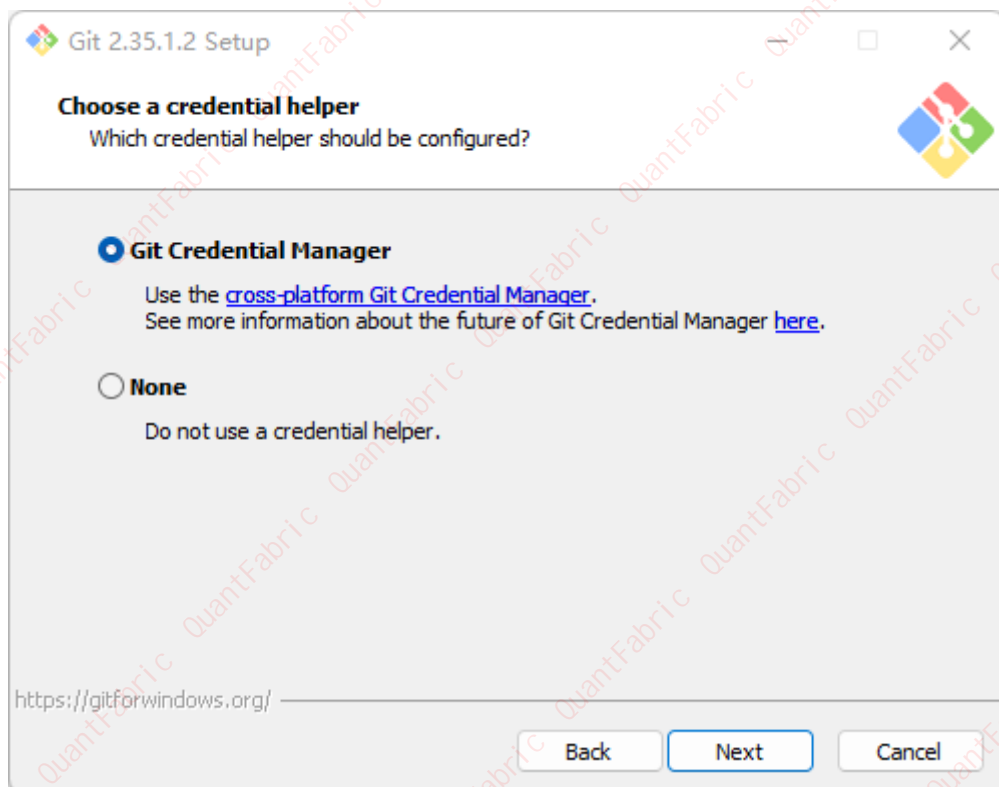
- 配置GitBash终端:



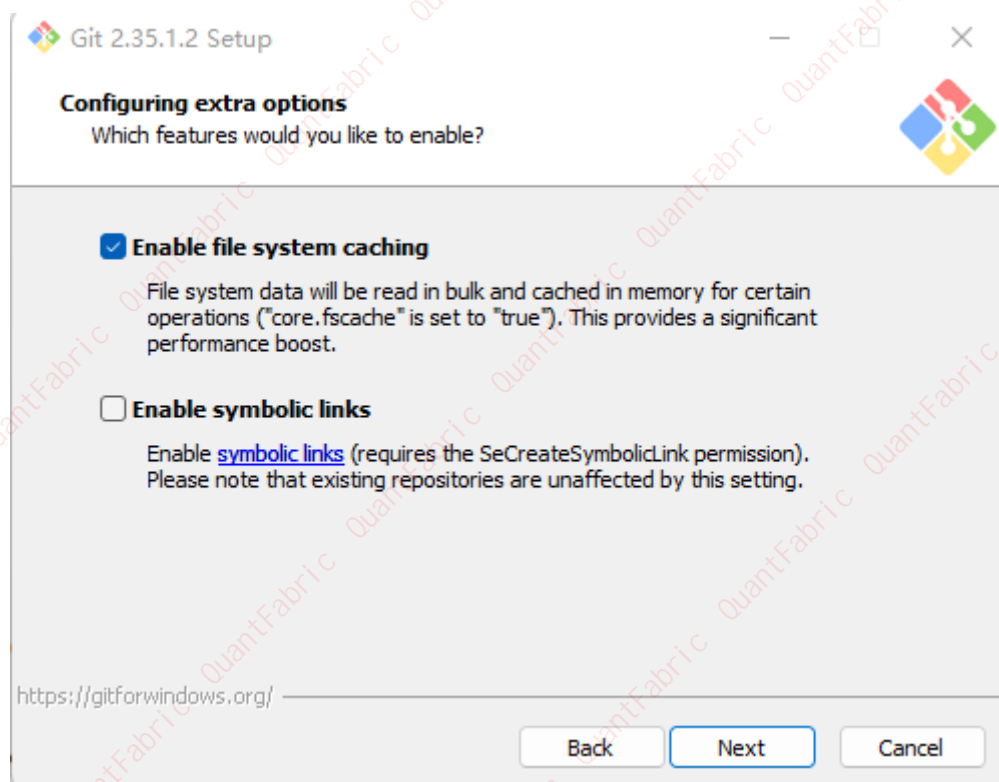
- 设置git pull默认分支合并策略:



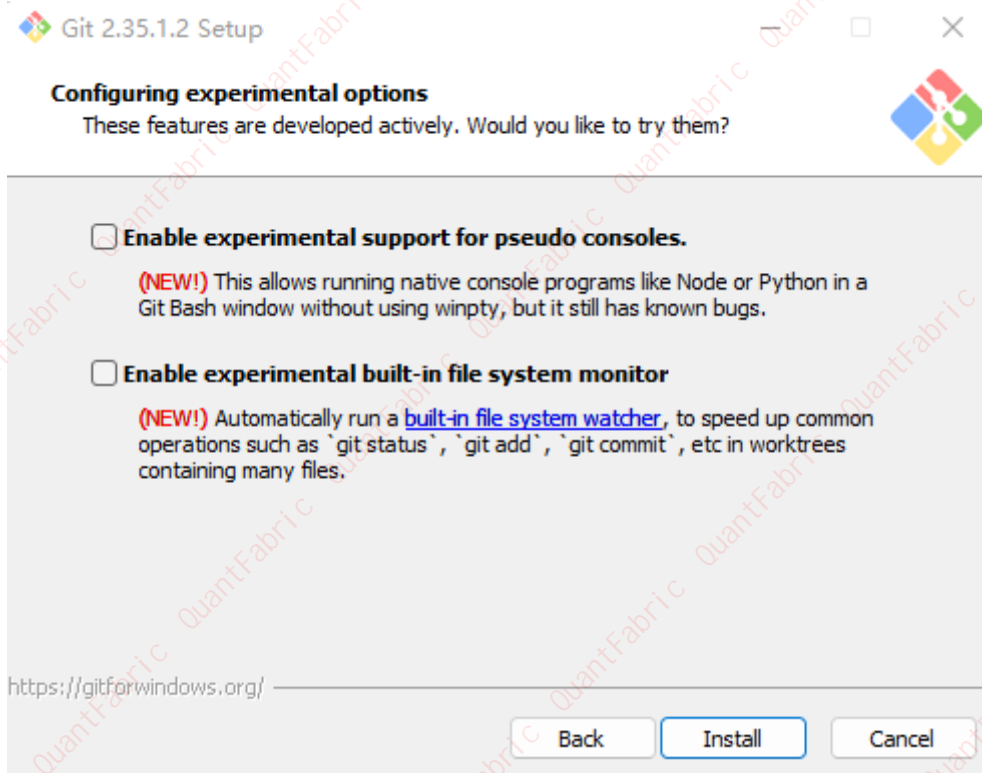
- 配置Git身份认证管理:



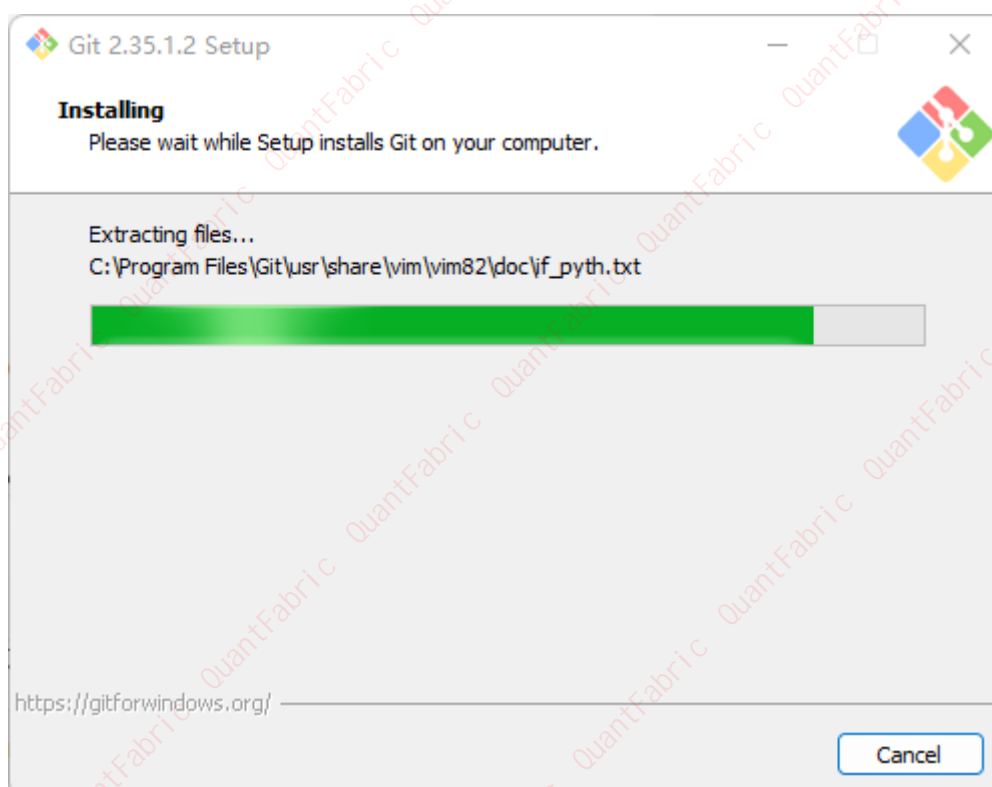
- 配置其它选项:



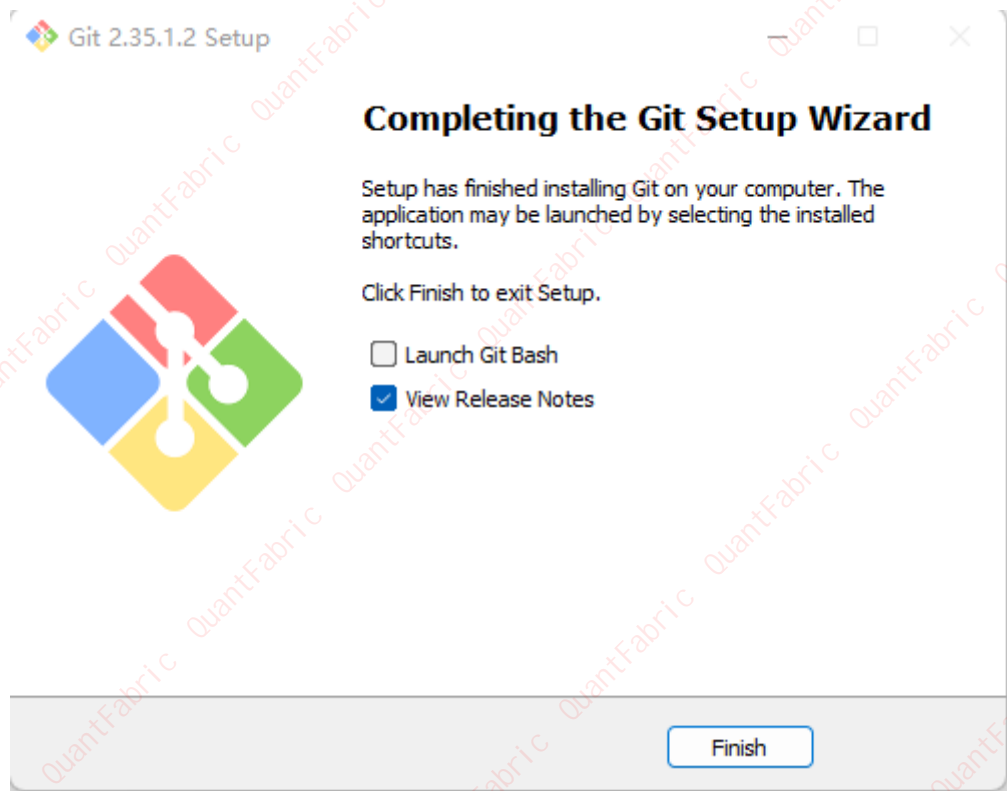
- 配置体验新功能:



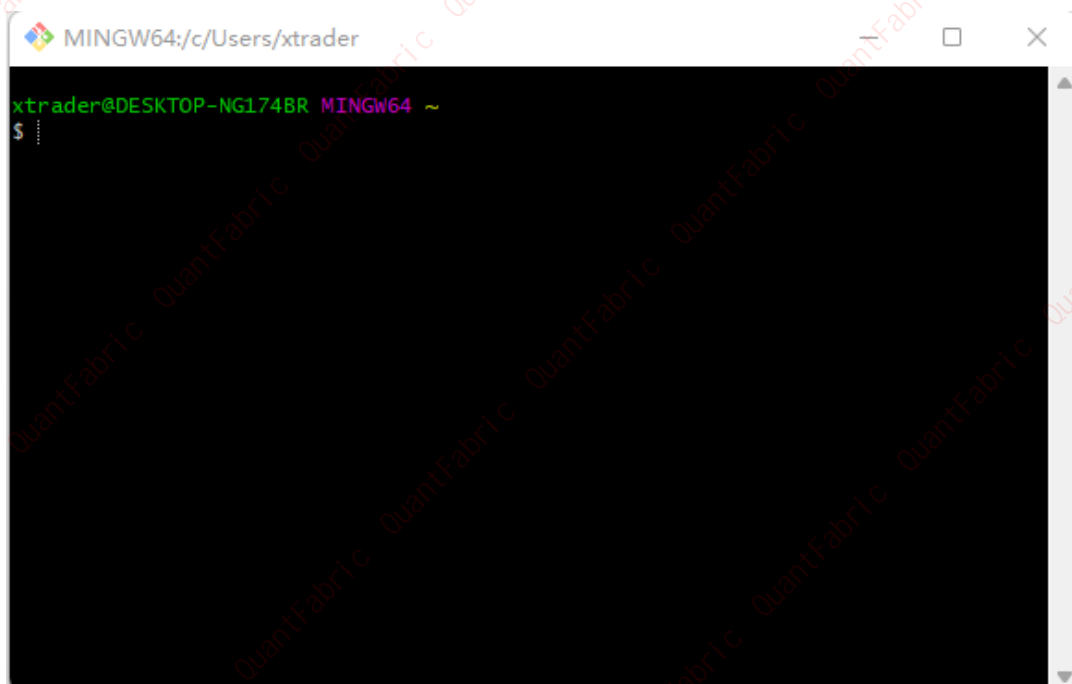
- 执行安装:



- 安装完成:



- 鼠标右键或点击桌面GitBash开启GitBash使用:



- Git学习文档: ProGit

二、SSH工具

1、SSH工具

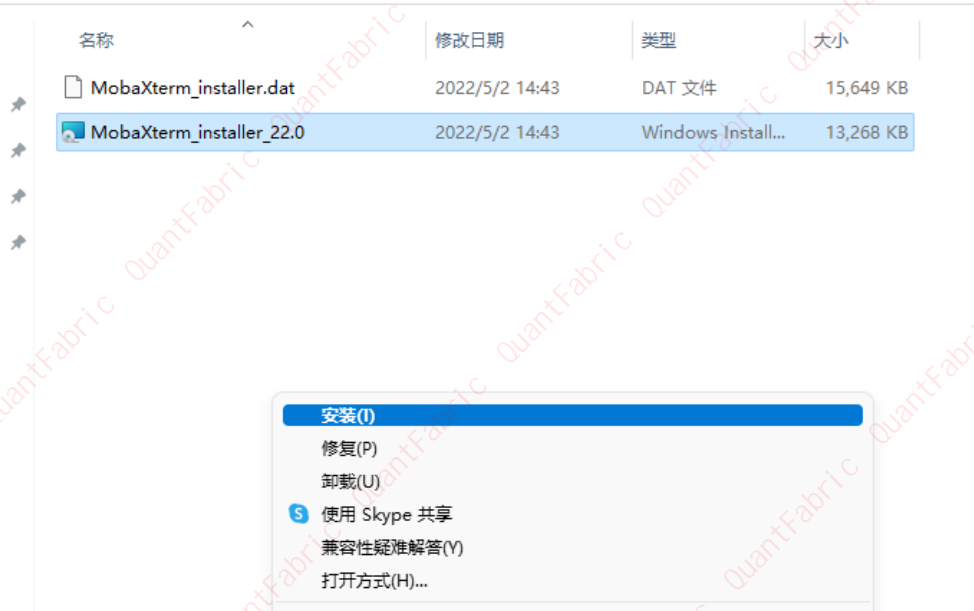
- FinalShell : 国产SSH工具, 支持MAC、Windows、Linux 平台, 支持批量服务器管理、实时硬盘监控、进程监控, 支持SSH和Windows远程桌面。
- MobaXterm : MobaXterm提供了所有重要的远程网络工具 (如SSH、X11、RDP、VNC、FTP、MOSH等) 以及Windows 桌面上的Unix命令 (bash、ls、cat、sed、grep、awk、rsync等), 登录后默认开启sftp模式, 仅支持Windows。
- Putty: Putty是最简单轻量级的SSH工具, 无需安装, 支持多系统版本。
- SecureCRT: SecureCRT是一款功能强大的付费SSH工具, 支持Windows、Mac、Linux、IOS等平台。SecureCRT除了包括一般工具都有的特点外, 还包括自动注册、对不同主机保持不同的特性、

打印功能、颜色设置、可变屏幕尺寸、用户定义的键位图等功能。

2、MobaXterm

- MobaXterm下载: [MobaXterm](#)
- MobaXterm安装流程:
 - 解压MobaXterm安装包将进行安装:

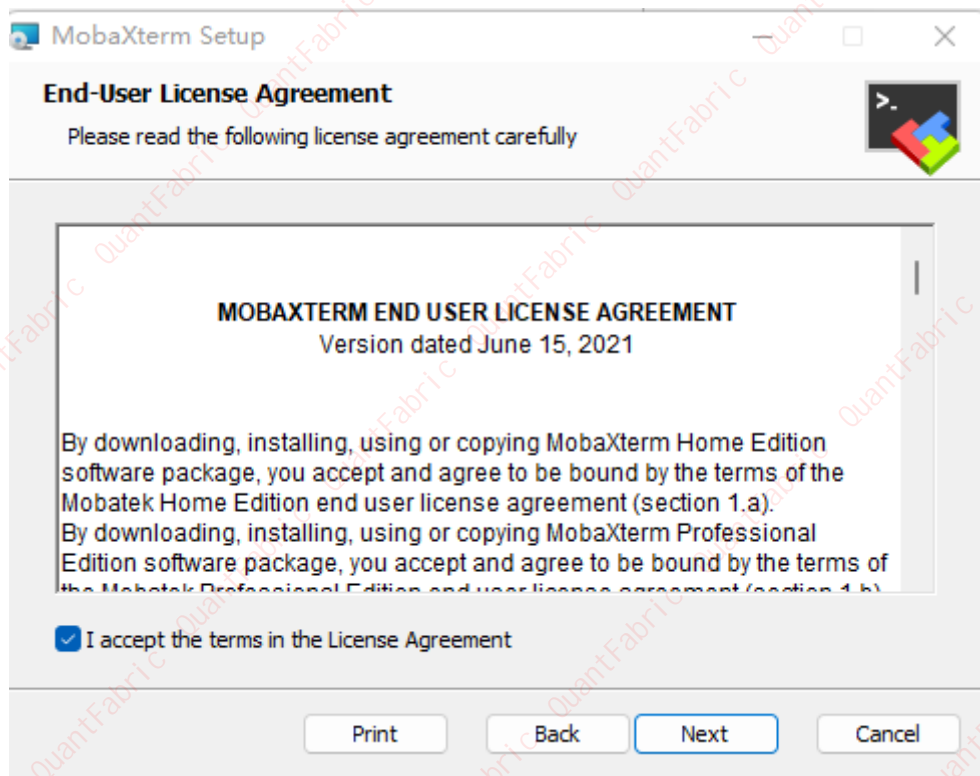
• MobaXterm_Installer_v22.0



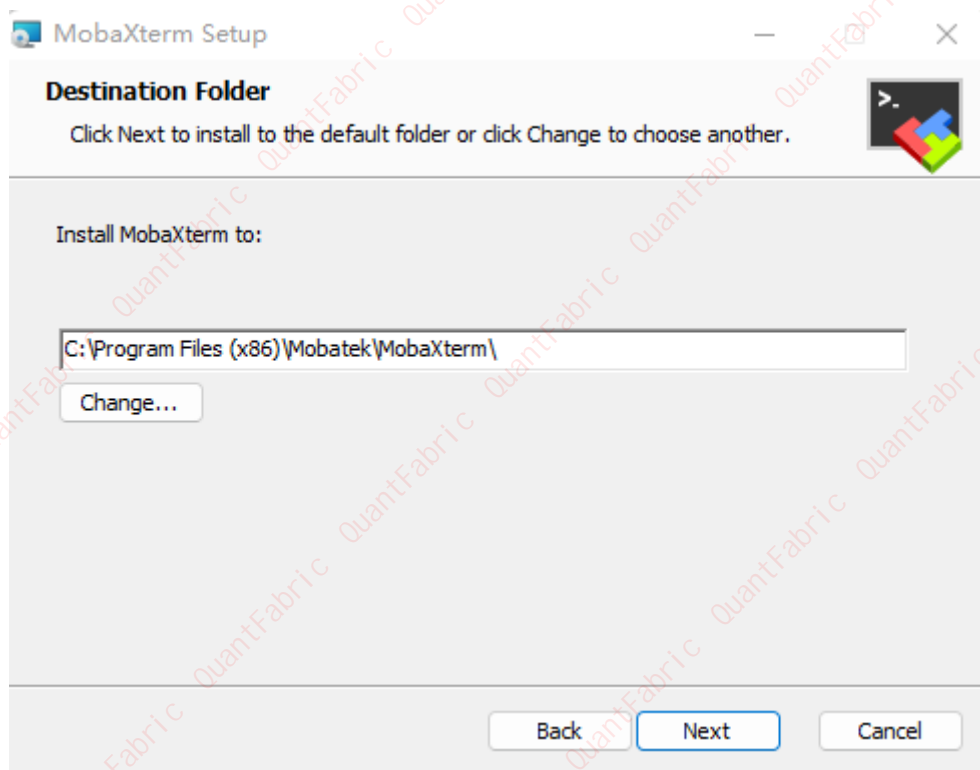
- 安装设置:



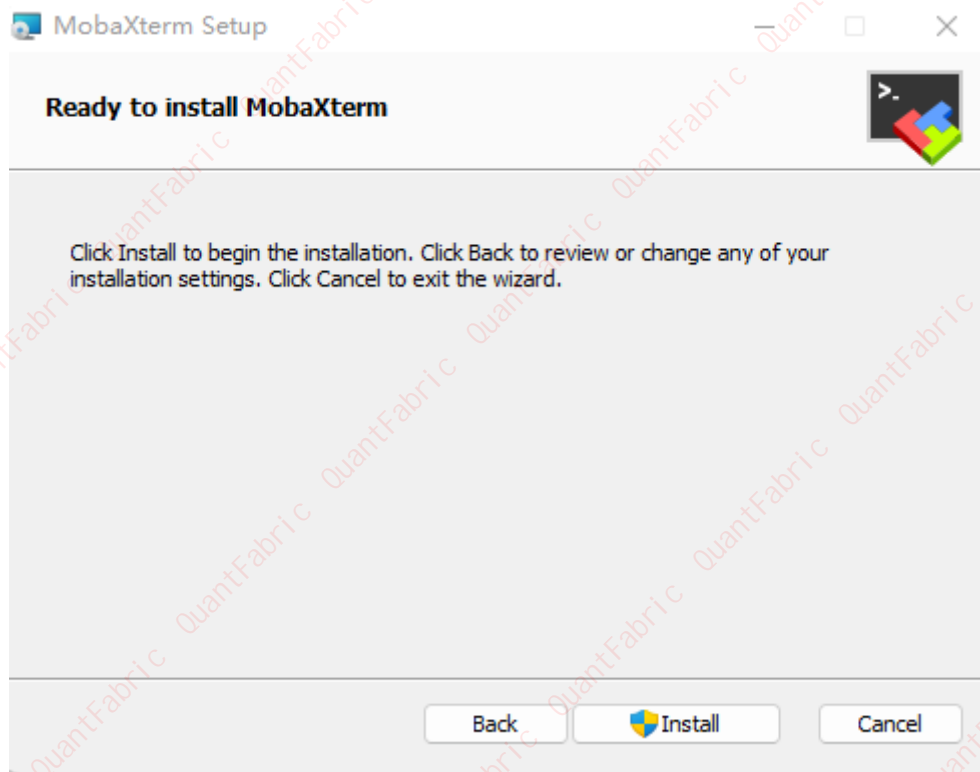
- 确认许可:



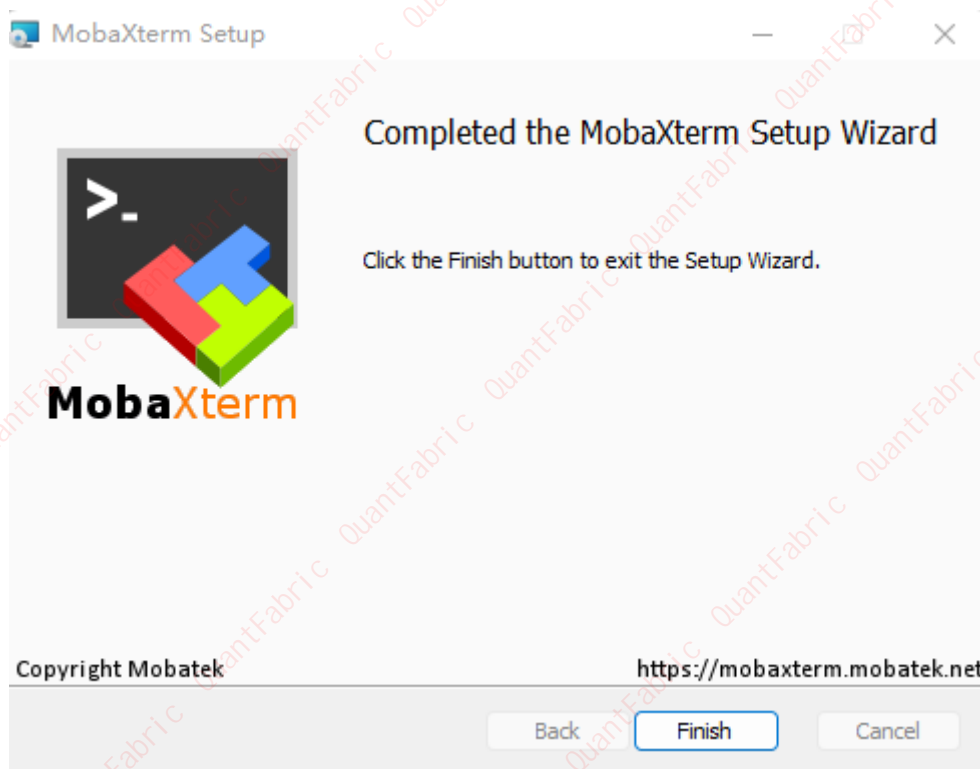
- 安装路径选择:



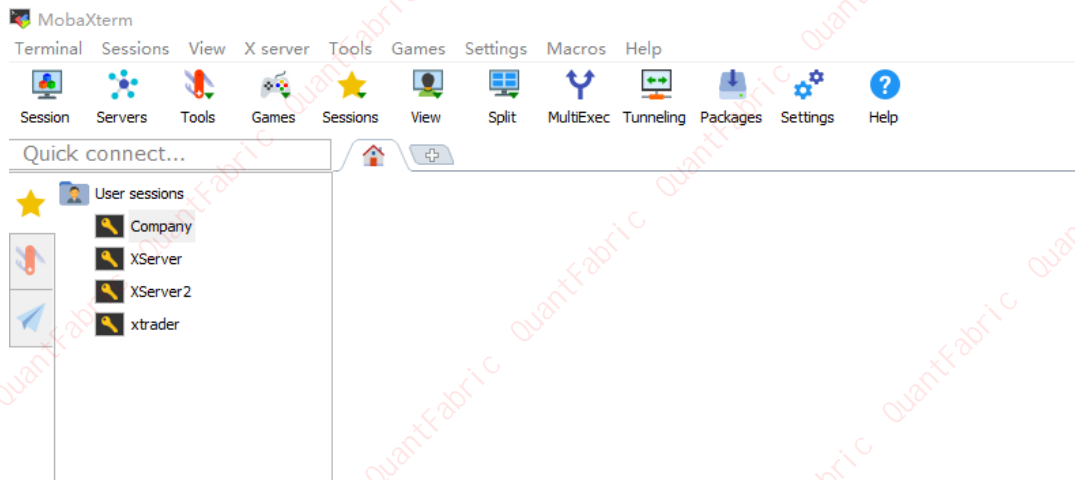
- 执行安装:



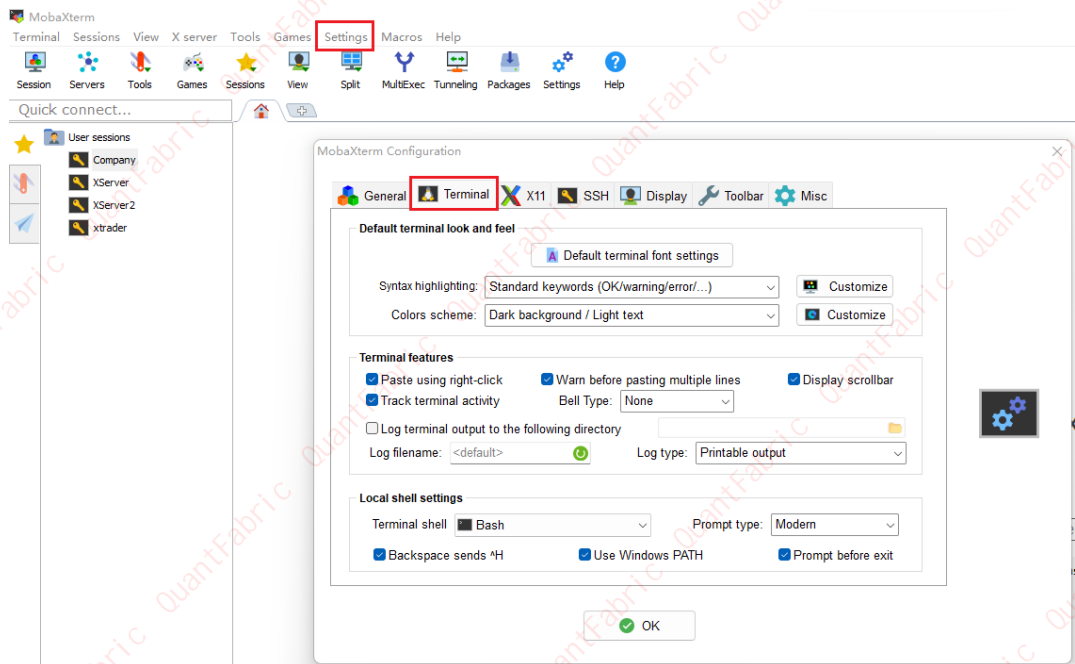
- 安装完成:



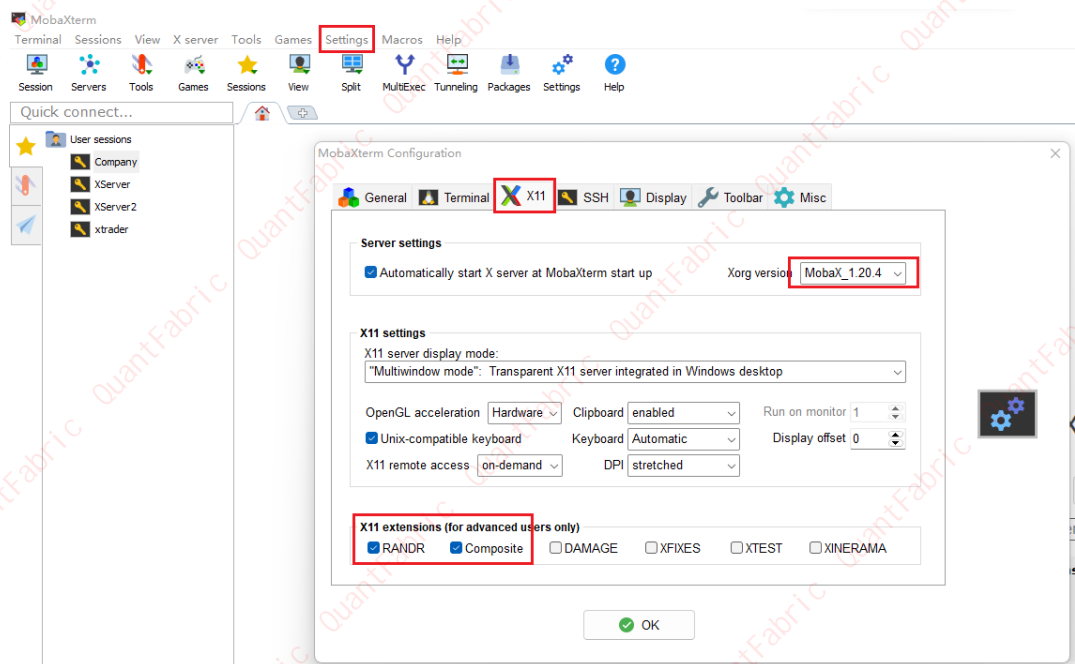
- 鼠标双击打开桌面MobaXterm程序:



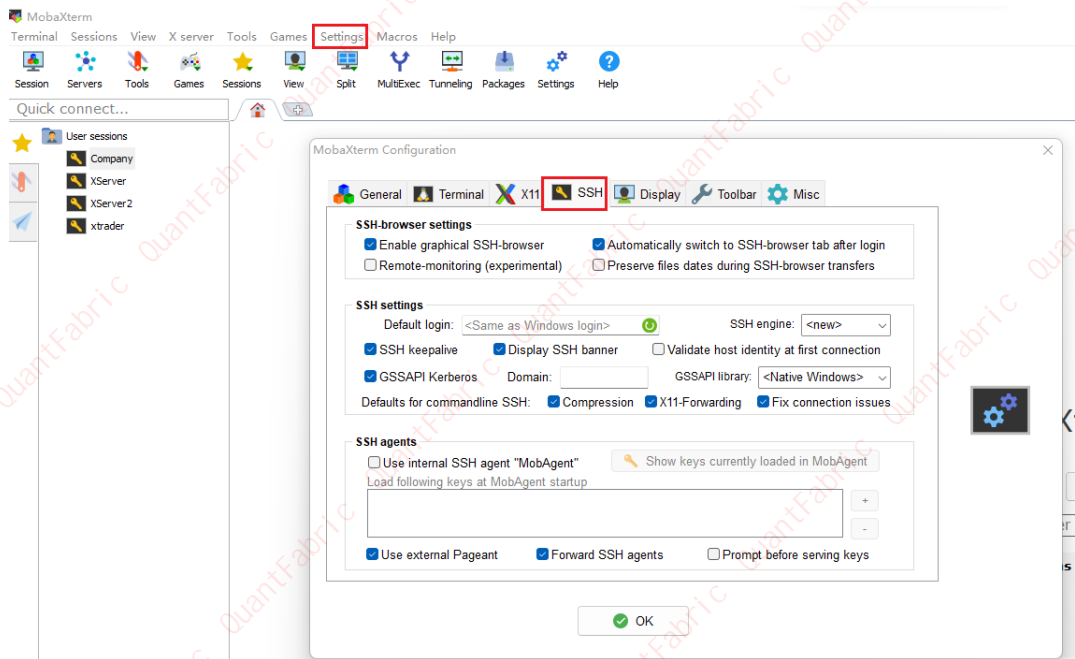
○ MobaXterm Terminal设置:



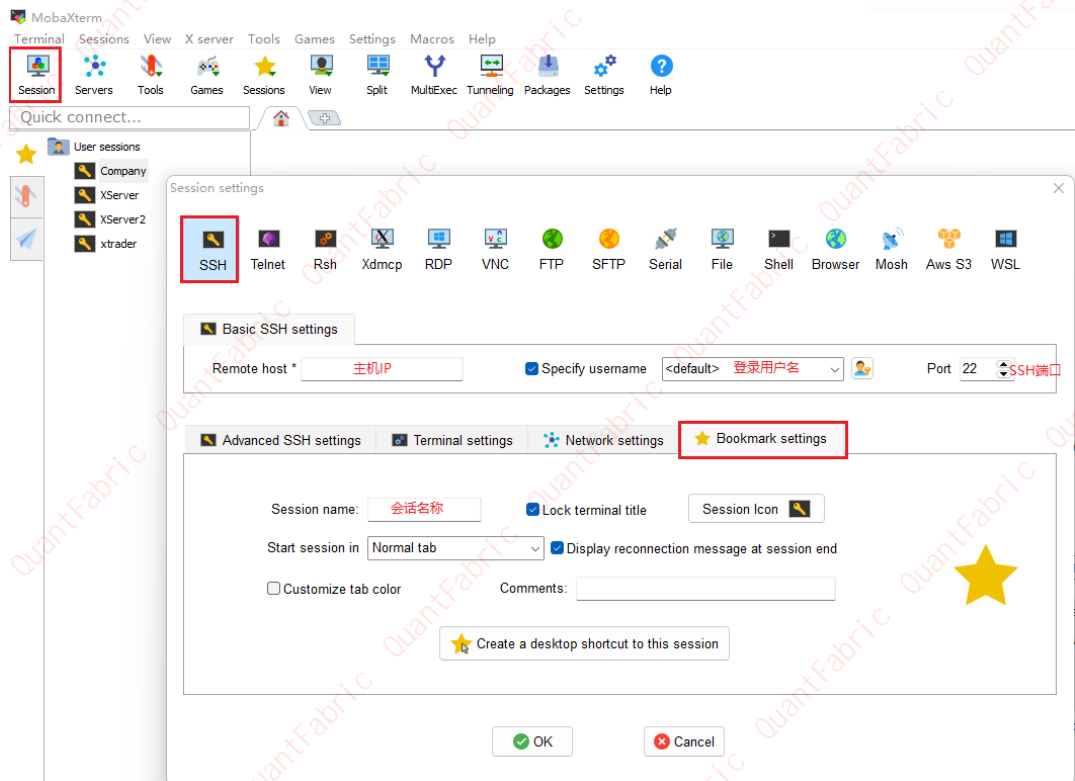
○ MobaXterm X11设置:



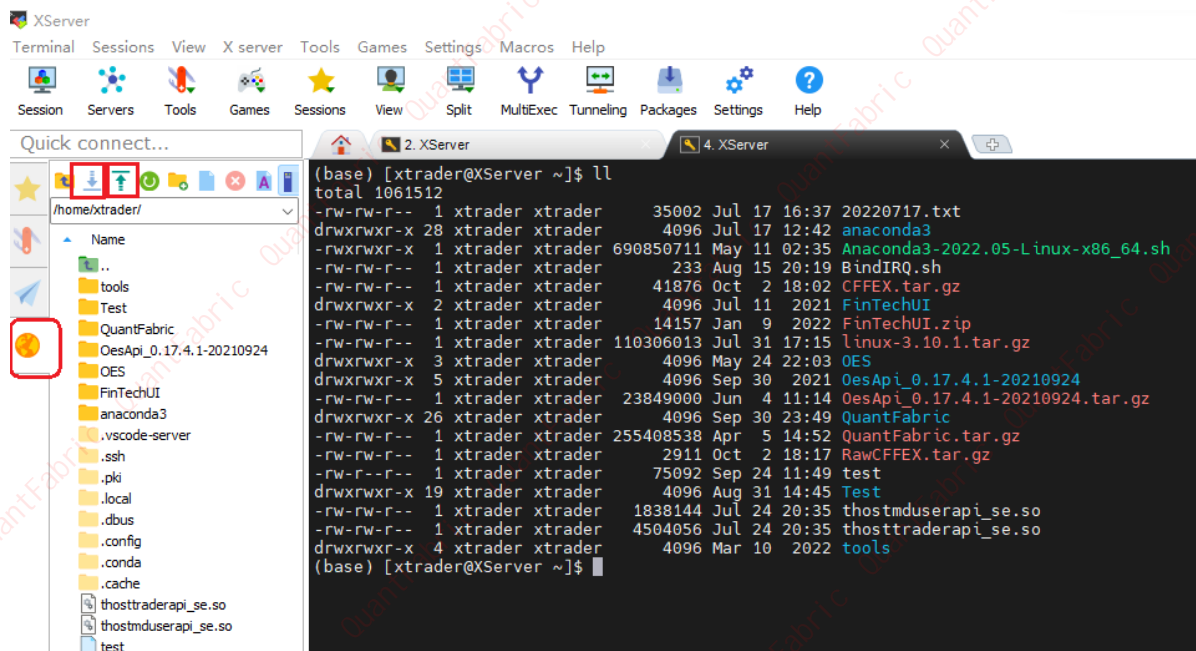
○ MobaXterm SSH设置:



会话连接建立:



• MobaXterm SFTP功能:



3、sshd服务端

- 远程主机sshd配置文件/etc/ssh/sshd_config:

```
Port 51622 # 修改ssh端口
PasswordAuthentication yes # 默认支持口令登录
```

- 重启sshd使改动生效:

```
/usr/sbin/sshd reload
```

4、创建用户

- 使用root用户登录Linux, 创建xtrader用户:

```
useradd xtrader
```

- 设置xtrader用户密码:

```
passwd xtrader # 输入两次密码
```

- 设置xtrader用户sudo权限, 打开 /etc/sudoers。

```
xtrader ALL=(ALL) NOPASSWD:ALL
```

- 普通用户切换到root用户:

```
su - root
```

5、ssh免密登录

- ssh密钥生成：

```
ssh-keygen -t rsa -C "your_email@example.com"
```

- ssh-keygen用于生成秘钥：
 - t：指定密钥类型，默认是rsa，可选dsa、ecdsa、ed25519、rsa。
 - C：指定注释，比如邮箱。
 - f：指定密钥文件名。
- ssh-keygen会在 ~/.ssh下生成 id_rsa私钥文件和id_rsa.pub公钥文件，私钥由客户端本地留存，公钥需保存到远程主机 ~/.ssh/authorized_keys文件内。
- 基于秘钥实现免密登录，通常需要先客户端PC生成公钥，然后将公钥拷贝到远程主机，拷贝过程既可以手动（在远程主机用户目录下创建.ssh目录，然后将公钥存入.ssh/authorized_keys文件中即可），也可以使用ssh-copy-id命令操作。

```
ssh-copy-id [-i [identity_file]] [-p port] [user@]hostname
ssh-copy-id -i .ssh/id_rsa.pub xtrader@Server
ssh-copy-id -p 30022 xtrader@Server
```

- 远程主机需要保证 .ssh和 authorized_keys都只有用户自己有写权限，否则验证无效。

```
chmod -R 700 ~/.ssh/
chmod 600 ~/.ssh/authorized_keys
```

- Windows客户端可以使用Git Bash的ssh-keygen和ssh-copy-id。
- SSH免密登录示例如下：

```
xtrader@DESKTOP-NG174BR MINGW64 ~
$ ssh-copy-id -p 51622 xtrader@47.108.252.223
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/c/Users/xtrader/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
xtrader@47.108.252.223's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh -p '51622'
'xtrader@47.108.252.223'"
and check to make sure that only the key(s) you wanted were added.

xtrader@DESKTOP-NG174BR MINGW64 ~
$ ssh -p '51622' 'xtrader@47.108.252.223'
Last login: Mon Oct  3 11:20:38 2022 from 171.221.139.134

welcome to Alibaba Cloud Elastic Compute Service !

(base) [xtrader@XServer ~]$
```

三、防火墙

1、Linux网络防火墙

```
systemctl start firewalld          # 开启防火墙
systemctl stop firewalld           # 关闭防火墙
systemctl status firewalld         # 查看状态
systemctl disable firewalld        # 禁止开机启动
firewall-cmd --zone=public --list-ports # 查看所有打开的端口
firewall-cmd --zone=public --add-port=80/tcp --permanent # 开放TCP 80端口
firewall-cmd --zone=public --remove-port=80/tcp --permanent # 删除防火墙80端口配置
firewall-cmd --reload              # 更新防火墙规则
firewall-cmd --zone=public --query-port=80/tcp # 查询TCP 80端口
```

2、云服务器

- 针对云服务器安装部署Linux开发环境，Linux服务器自身有Linux网络防火墙，但云服务厂商本身仍然有一层网络防火墙，可以登录相应云服务(阿里云、天翼云等)查看、设置相应服务端口的开放情况。
- 如果不熟悉Linux系统环境，建议关闭Linux网络防火墙，保留云厂商防火墙。

四、GCC安装

- 安装EPEL

```
yum -y install epel-release
```

- 安装SCL源

```
yum install centos-release-scl scl-utils-build
```

- 安装devtoolset-9、rh-python36

```
yum install devtoolset-9
yum install rh-python36
```

- 开启SCL软件集，修改用户目录下`.bashrc`，追加。

```
source /opt/rh/devtoolset-9/enable
```

- 执行source，立即生效

```
source ~/.bashrc
```

- GCC版本查看：

```
[xtrader@XServer ~]$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/opt/rh/devtoolset-9/root/usr/libexec/gcc/x86_64-redhat-
linux/9/lto-wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --enable-bootstrap --enable-
languages=c,c++,fortran,lto --prefix=/opt/rh/devtoolset-9/root/usr --
mandir=/opt/rh/devtoolset-9/root/usr/share/man --infodir=/opt/rh/devtoolset-
9/root/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --
enable-shared --enable-threads=posix --enable-checking=release --enable-multilib
--with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --
enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only
--with-linker-hash-style=gnu --with-default-libstdcxx-abi=gcc4-compatible --
enable-plugin --enable-initfini-array --with-isl=/builddir/build/BUILD/gcc-
9.3.1-20200408/obj-x86_64-redhat-linux/isl-install --disable-libmpx --enable-
gnu-indirect-function --with-tune=generic --with-arch_32=x86-64 --build=x86_64-
redhat-linux
Thread model: posix
gcc version 9.3.1 20200408 (Red Hat 9.3.1-2) (GCC)
```

- SQLite开发库安装:

```
sudo yum install sqlite-devel
```

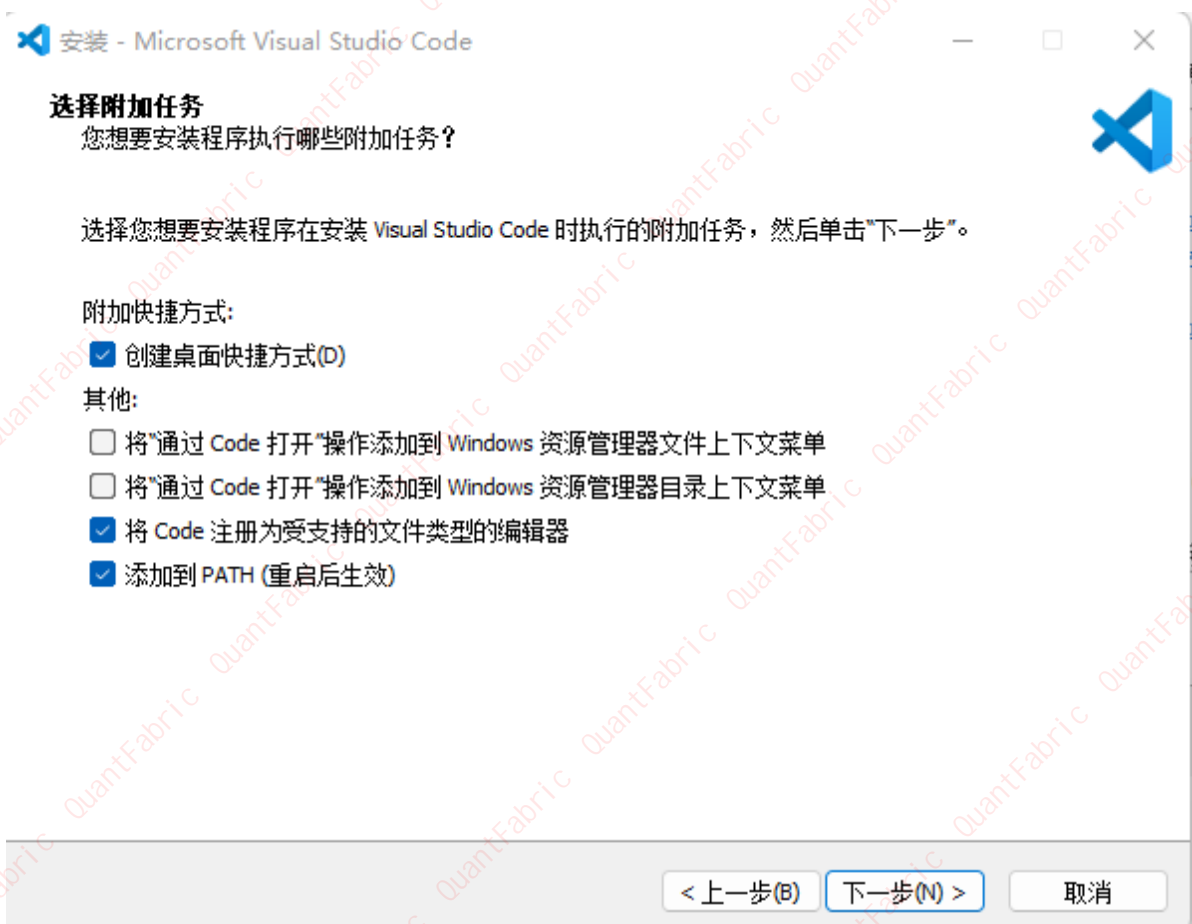
五、VSCode

1、VSCode安装

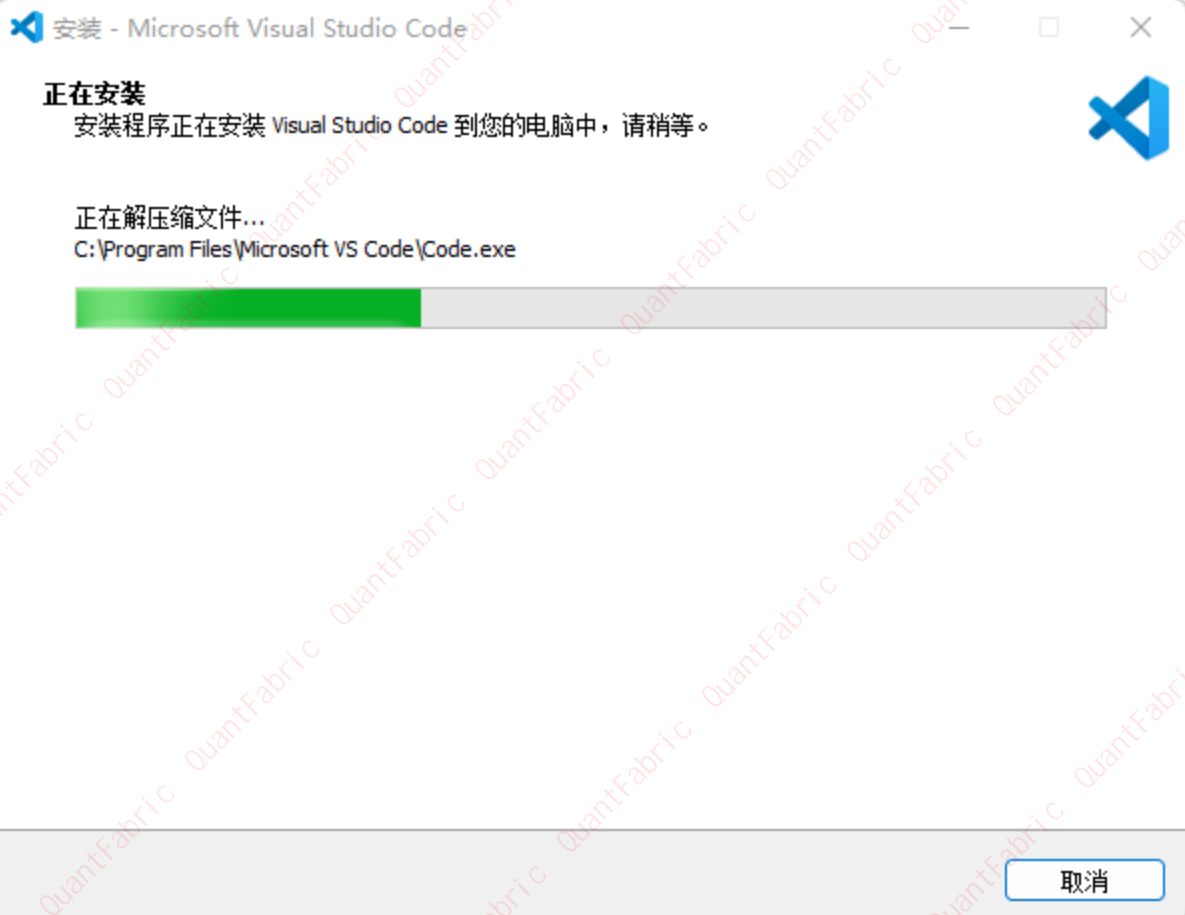
- 下载: [VSCode](#)
- 点击鼠标右键选择管理员身份安装VSCodeSetup:



- 选择配置:



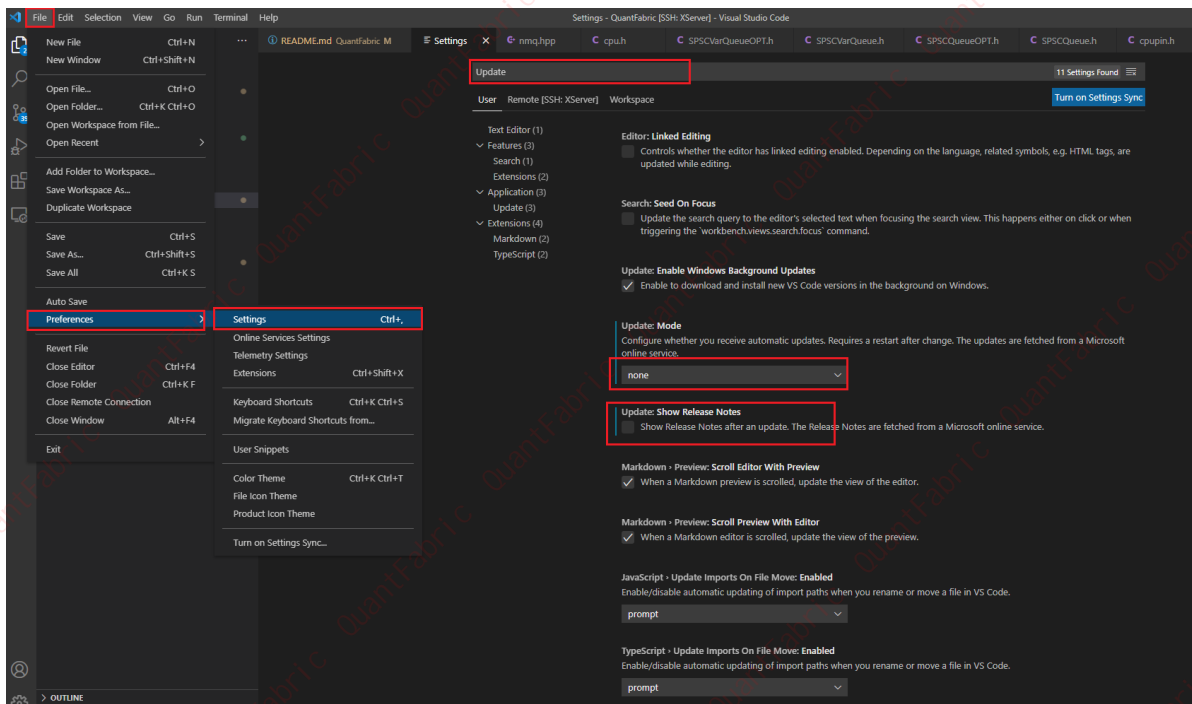
- 执行安装:



- 安装完成:



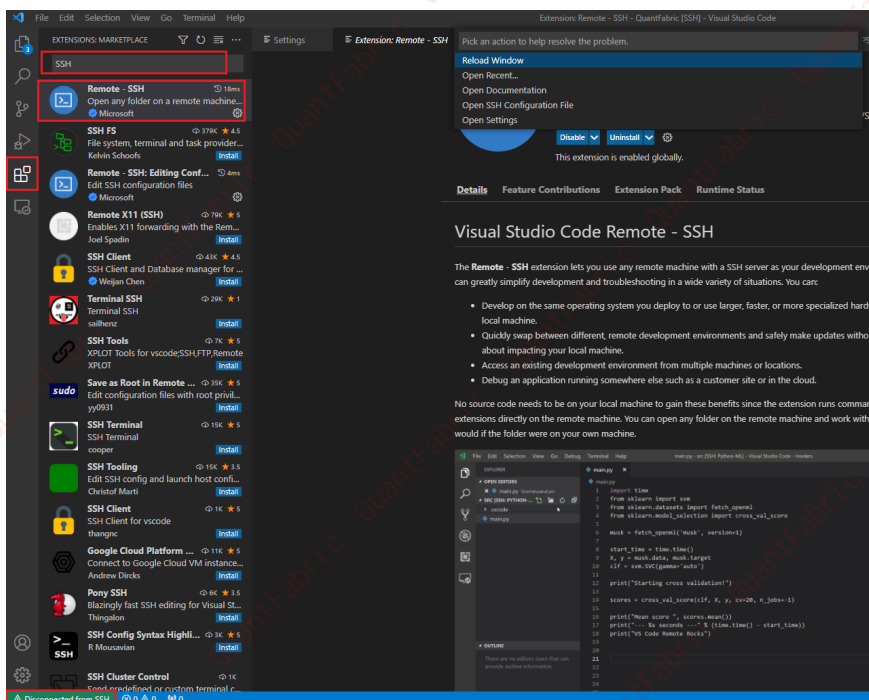
- 关闭自动更新服务:



- 注意：不要升级VSCode客户端，由于升级VSCode版本可能会导致VSCode客户端与运行在Linux服务器的vscode-server服务端不兼容，并且由于下载升级vscode-server需要通过国际网络，因此可能很容易导致VSCode客户端升级后连接vscode-server服务端一直处于失败状态。

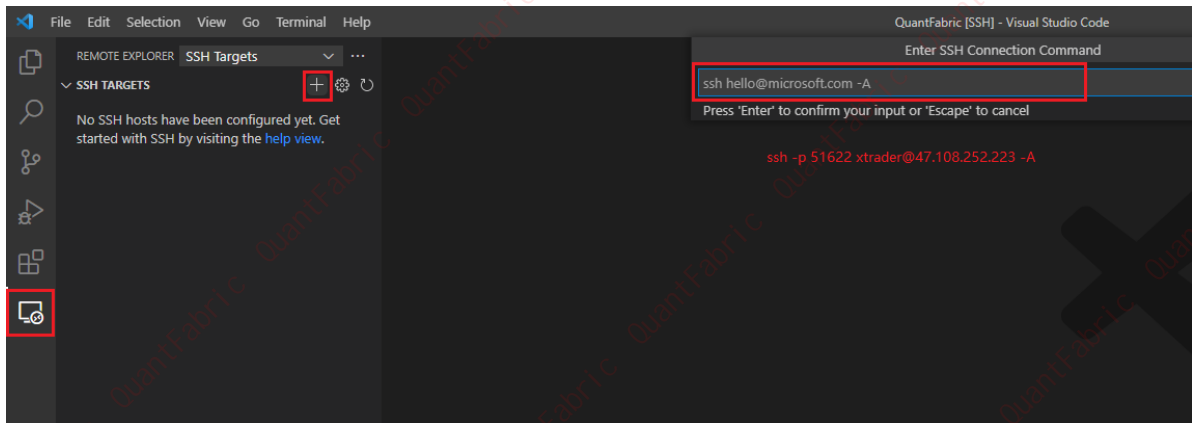
2、Remote-SSH插件安装

- VSCode应用商店搜索SSH，选中Remote-SSH插件安装。



3、远程开发

- 连接远程Linux服务器：



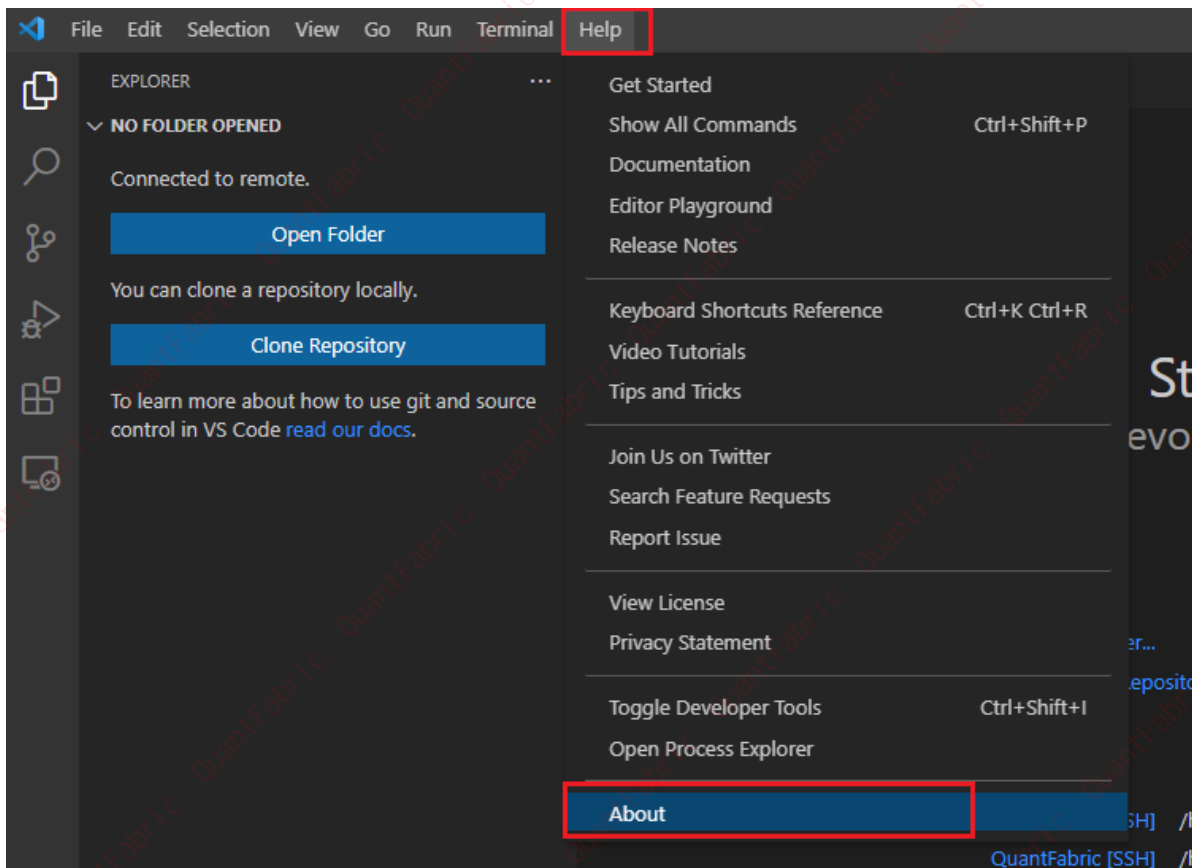
- VSCode远程连接完成后，安装在用户目录下.vscode-server目录。

```
[xtrader@XServer .vscode-server]$ tree -L 3
.
├── bin
│   ├── b3318bc0524af3d74034b8bb8a64df0ccf35549a
│   │   ├── bin
│   │   ├── extensions
│   │   ├── LICENSE
│   │   ├── node
│   │   ├── node_modules
│   │   ├── out
│   │   ├── package.json
│   │   ├── product.json
│   │   ├── server.sh
│   │   └── vscode-remote-lock.xtrader.b3318bc0524af3d74034b8bb8a64df0ccf35549a
├── data
│   ├── logs
│   │   └── 20221003T122057
│   ├── Machine
│   ├── machineid
│   ├── User
│   │   └── globalstorage
└── extensions
```

- bin目录下面存放VS Code Server程序，extensions目录存放VS Code Server端安装的插件，data目录下是用户数据。

4、vscode-server离线安装

- VSCode版本确认：



Visual Studio Code



Visual Studio Code

Version: 1.62.0 (system setup)

Commit: b3318bc0524af3d74034b8bb8a64df0ccf35549a

Date: 2021-11-03T15:23:01.379Z (11 mos ago)

Electron: 13.5.1

Chrome: 91.0.4472.164

Node.js: 14.16.0

V8: 9.1.269.39-electron.0

OS: Windows_NT x64 10.0.22000

OK

Copy

- 下载相应版本的vscode-server，可以在Windows端下载后上传也可以直接在Linux服务器下载

```
[xtrader@XServer ~]commit_id=c47d83b293181d9be64f27ff093689e8e7aed054
[xtrader@XServer ~]curl -SSL
"https://update.code.visualstudio.com/commit:${commit_id}/server-linux-x64/stable" -o vscode-server-linux-x64.tar.gz
```

- 安装vscode-server:

```
[xtrader@XServer ~]mkdir -p ~/.vscode-server/bin/${commit_id}
[xtrader@XServer ~]tar zxvf vscode-server-linux-x64.tar.gz -C ~/.vscode-server/bin/${commit_id} --strip 1
[xtrader@XServer ~]touch ~/.vscode-server/bin/${commit_id}/0
```

六、Qt

1、桌面环境安装

```
yum -y groupinstall "GNOME Desktop"
yum -y groupinstall "Development Tools"
yum install -y libGL libGL-devel
yum install -y mesa-libGL-devel mesa-libGLU-devel freeglut-devel
yum install -y libxcb libxcb-devel libxrender libxrender-devel xcb-util-wm xcb-
util-wm-devel xcb-util xcb-util-devel xcb-util-image xcb-util-image-devel xcb-
util-keysyms xcb-util-keysyms-devel
```

2、Qt安装

- Qt版本: [Qt 5.12.12](#)
- 下载Qt 5.12.12:

Name	Last modified	Size	Metadata
↑ Parent Directory		-	
submodules/	25-Nov-2021 08:09	-	
single/	25-Nov-2021 08:08	-	
qt-opensource-windows-x86-5.12.12.exe	25-Nov-2021 08:12	3.7G	Details
qt-opensource-mac-x64-5.12.12.dmg	25-Nov-2021 08:11	2.7G	Details
qt-opensource-linux-x64-5.12.12.run	25-Nov-2021 08:10	1.3G	Details
md5sums.txt	25-Nov-2021 08:18	210	Details

For Qt Downloads, please visit qt.io/download

Qt® and the Qt logo is a registered trade mark of The Qt Company Ltd and is used pursuant to a license from The Qt Company Ltd.
All other trademarks are property of their respective owners.

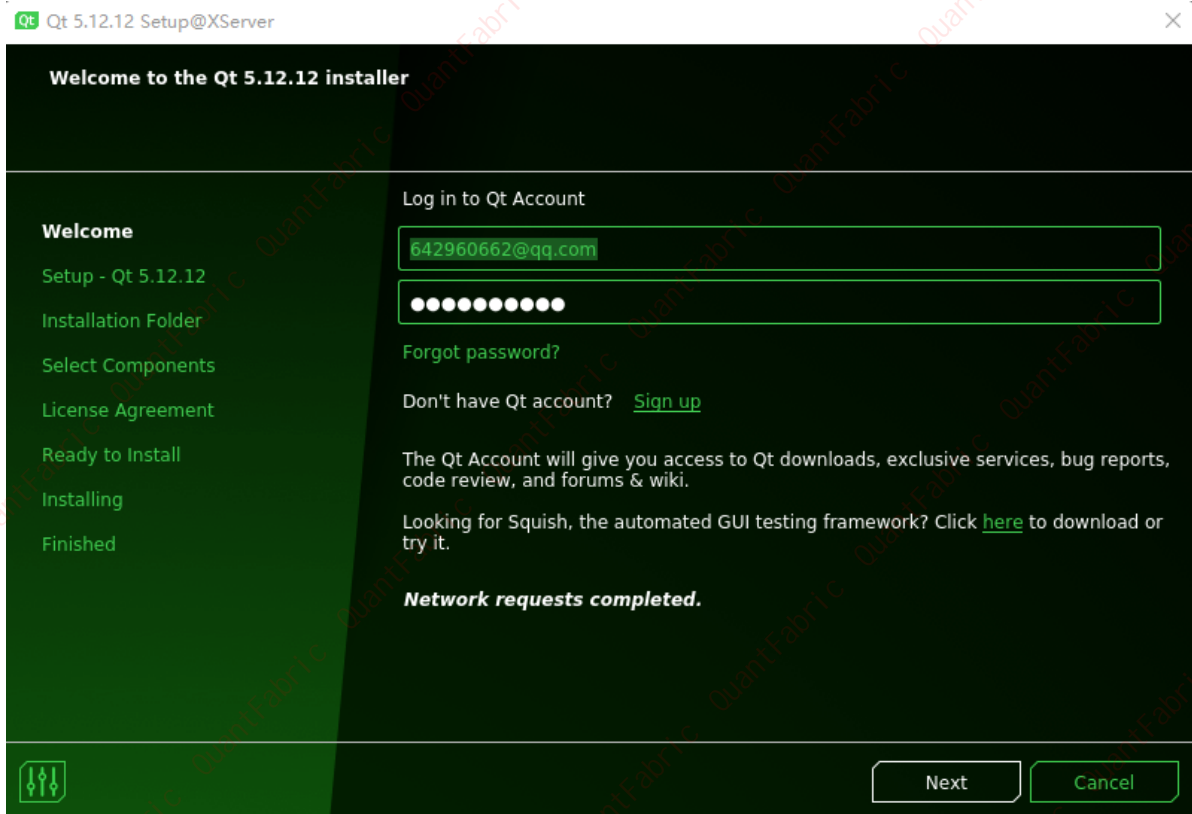
The Qt Company Ltd, Bertel Jungin aukio D3A, 02600 Espoo, Finland. Org. Nr. 2637805-2

[List of official Qt-project mirrors](#)

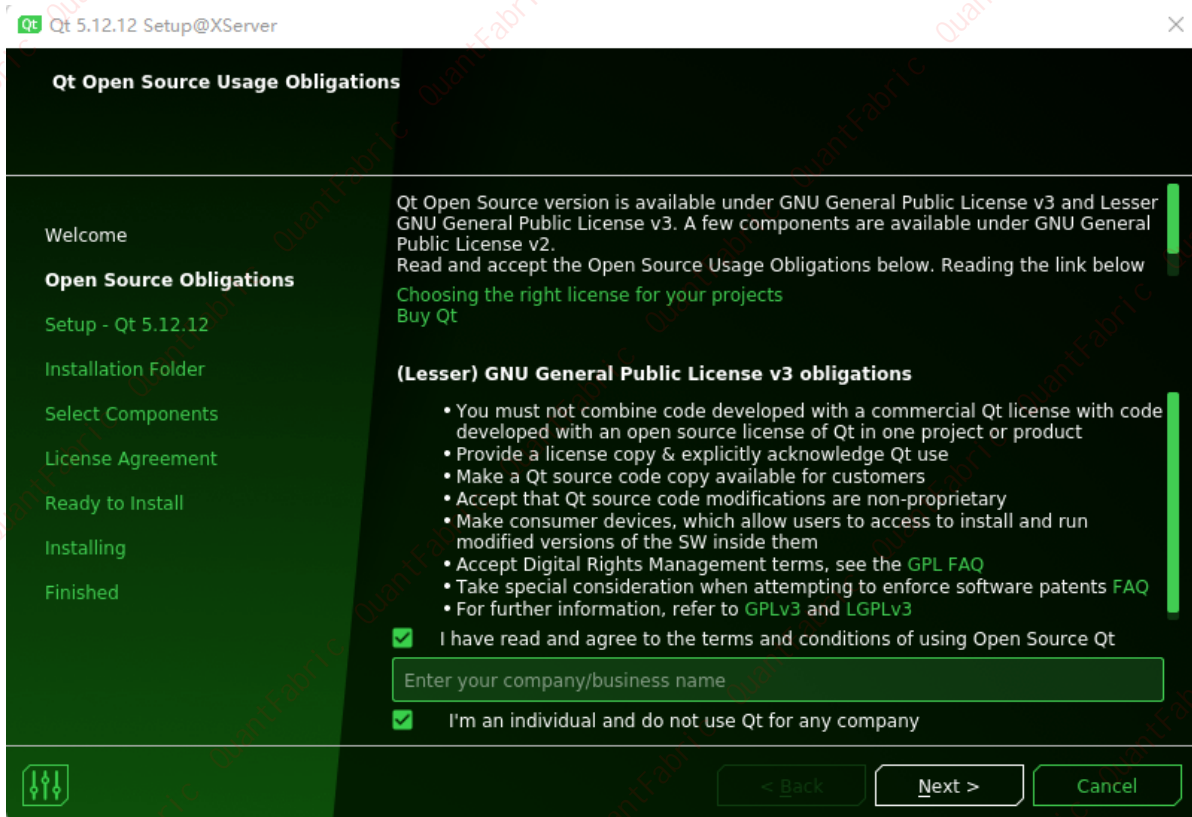
- 通过MobaXterm SFTP工具将安装包上传至Linux服务器，执行安装操作:

```
sudo ./qt-opensource-linux-x64-5.12.12.run
```

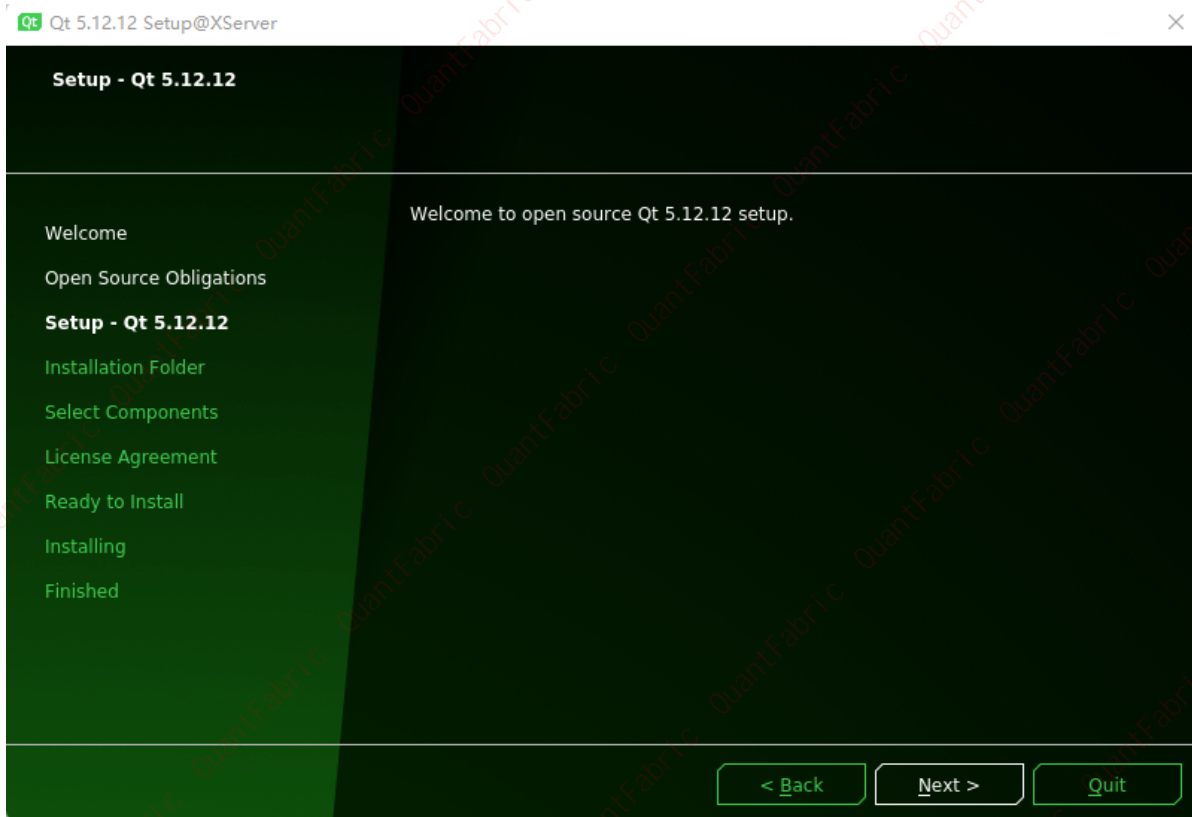
- 登录Qt账户认证，需要到Qt官网注册账户: [Qt Account](#)



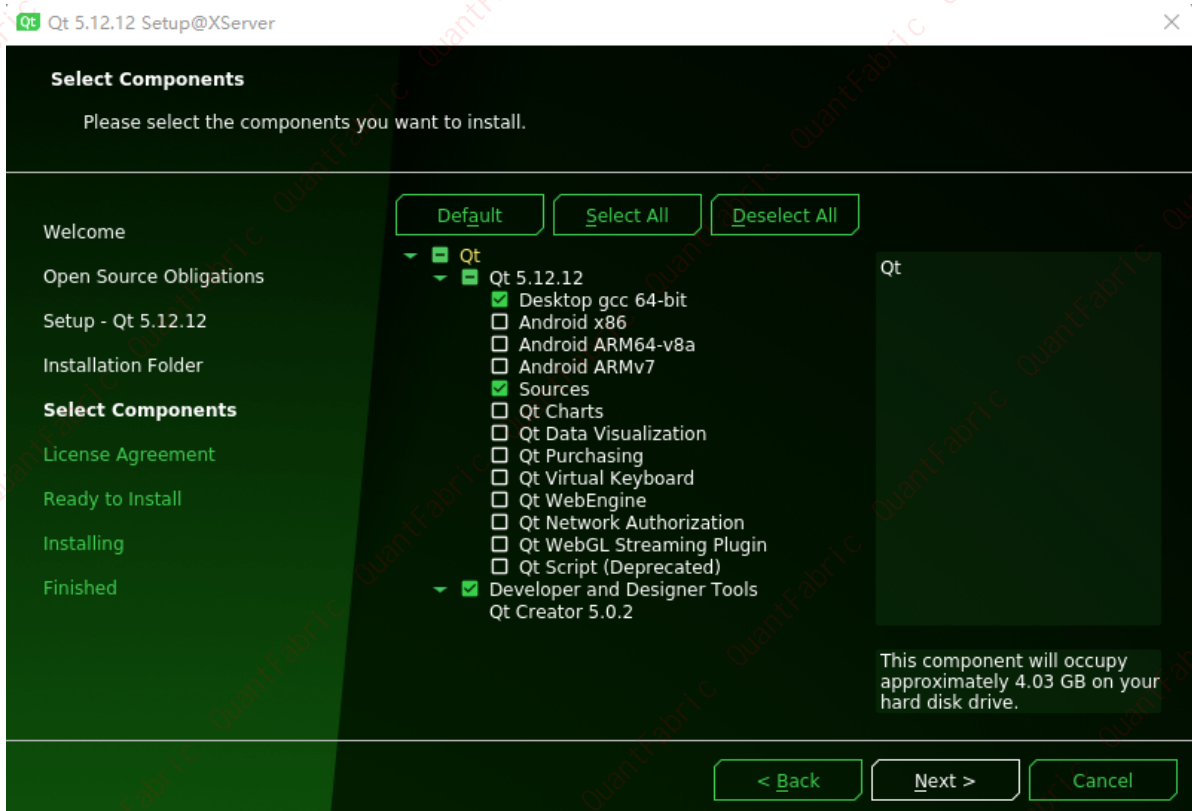
- 许可协议确认:



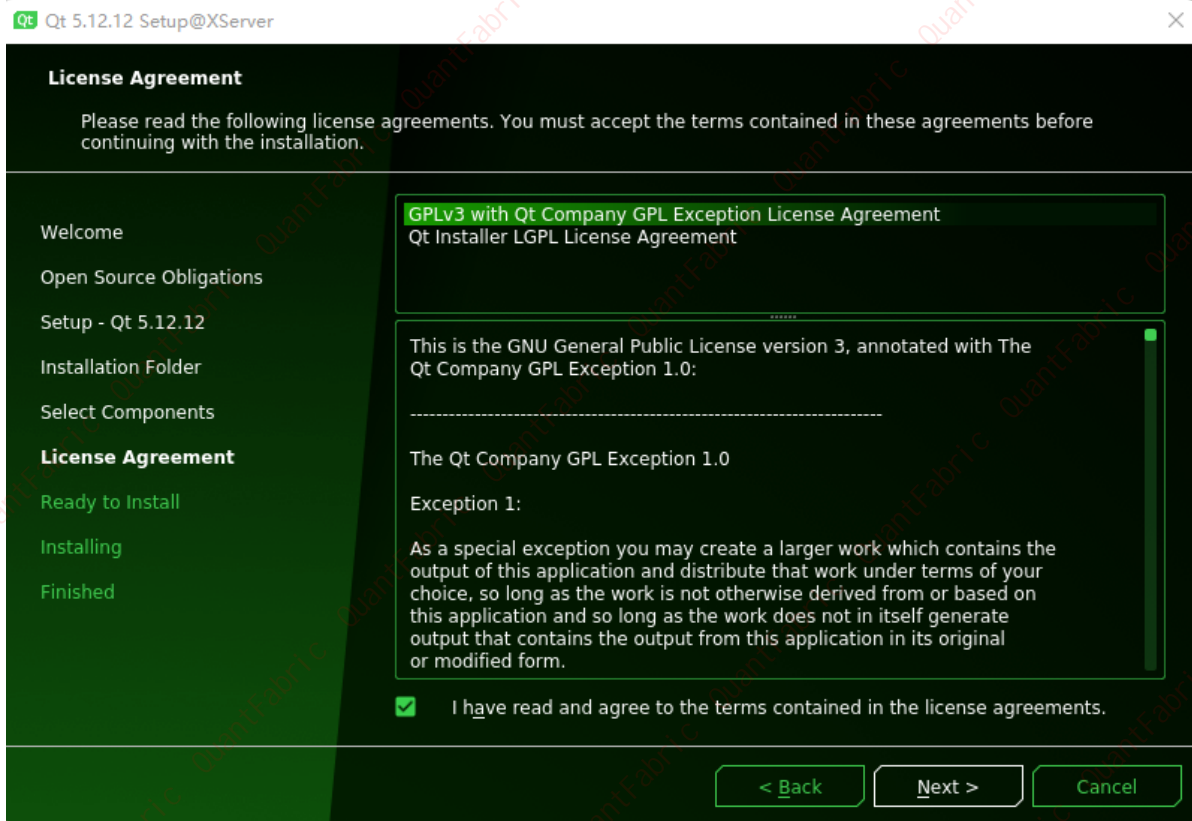
- 准备安装:



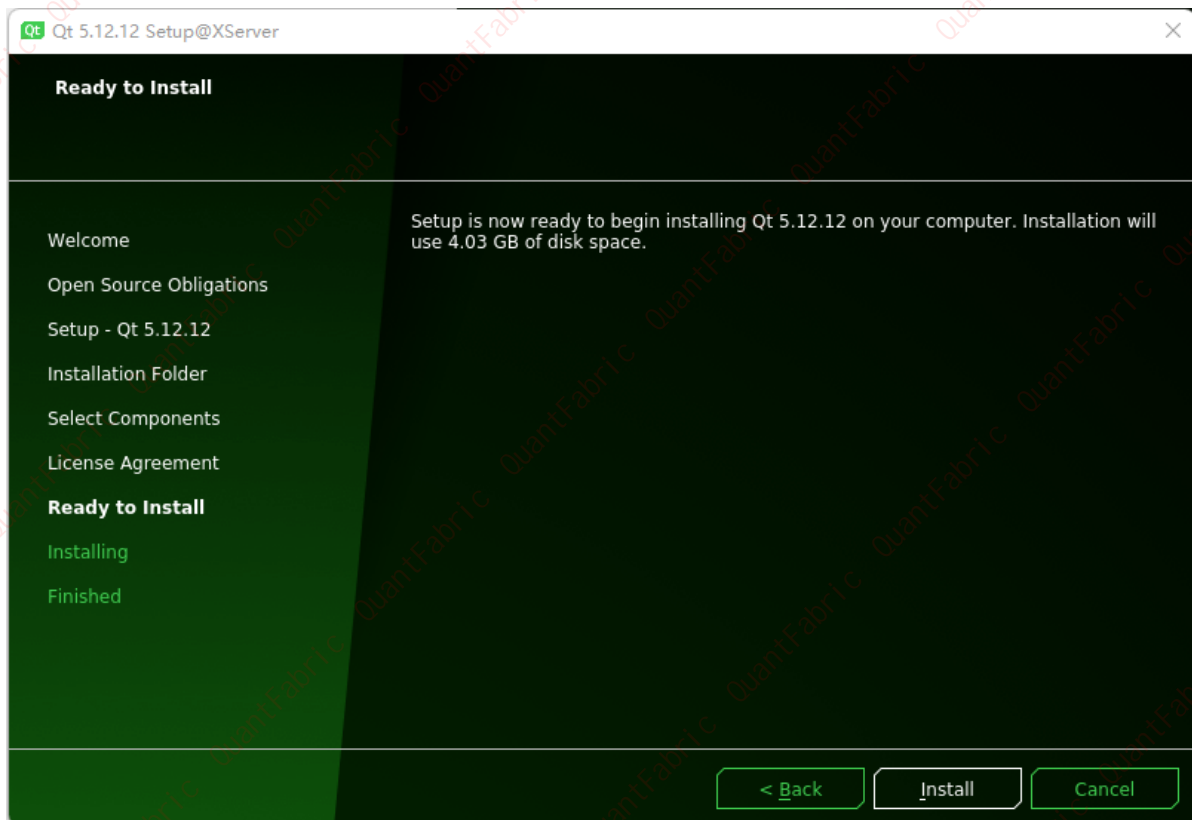
- 选择安装路径
- 安装组件选择:



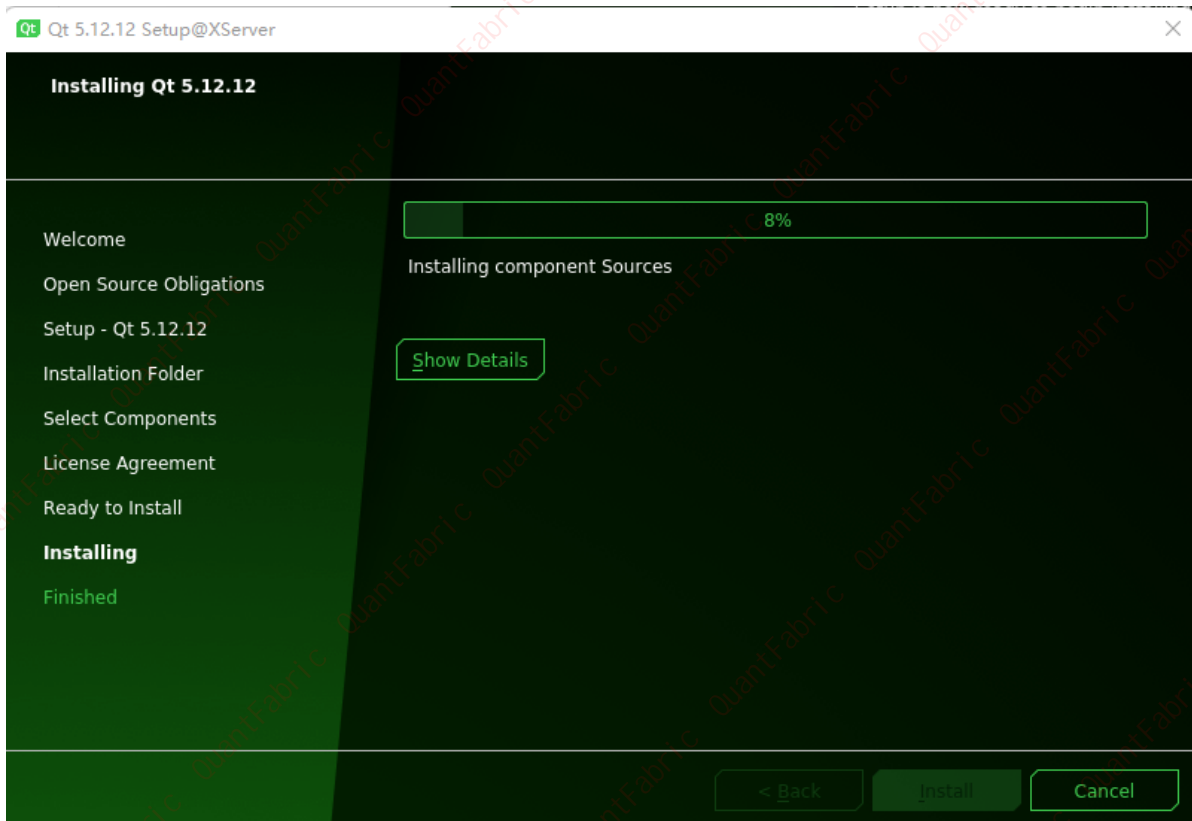
- 许可协议确认:



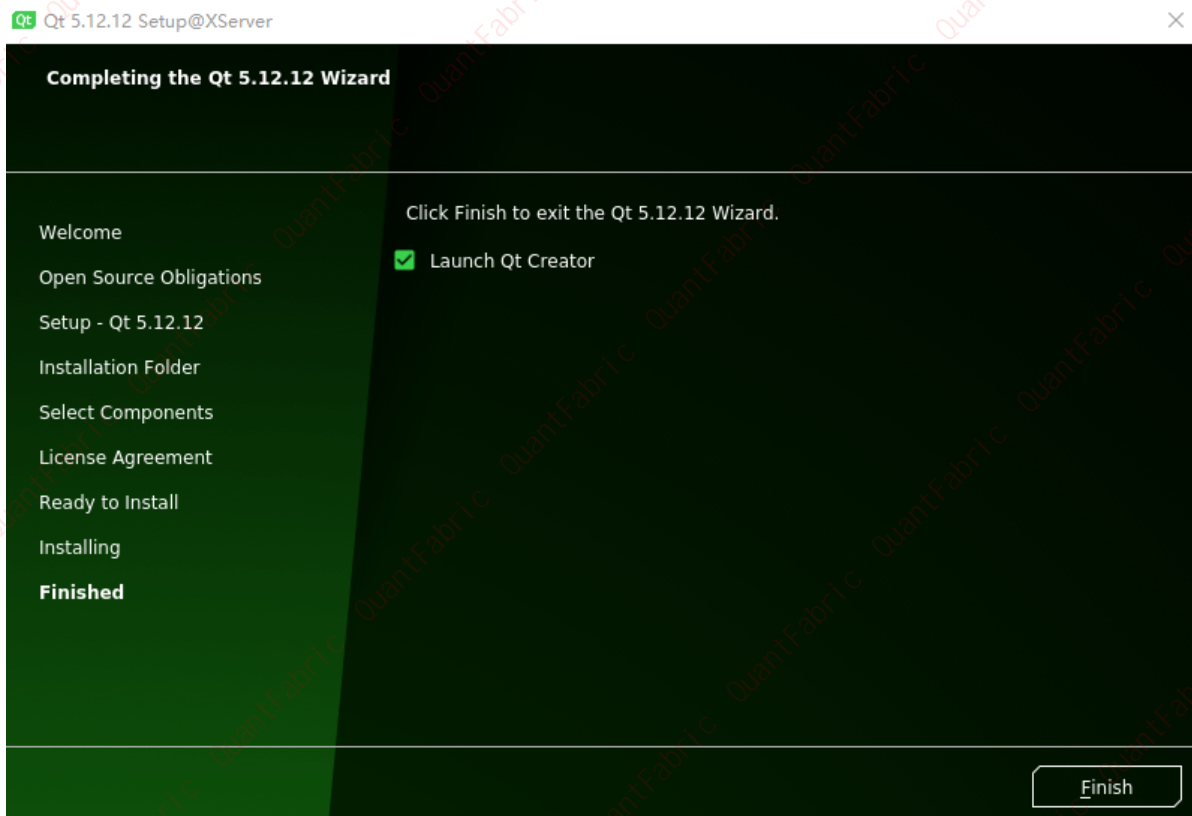
- 执行安装:



- 安装过程:



- 安装完成:



- 修改用户的 `.bash_profile` 增加QT环境变量设置:

```
export QTDIR=/home/xtrader/Qt5.12.12/  
export PATH=$QTDIR/5.12.12/gcc_64/bin:$QTDIR/Tools/QtCreator/bin/:$PATH
```

- 执行生效:

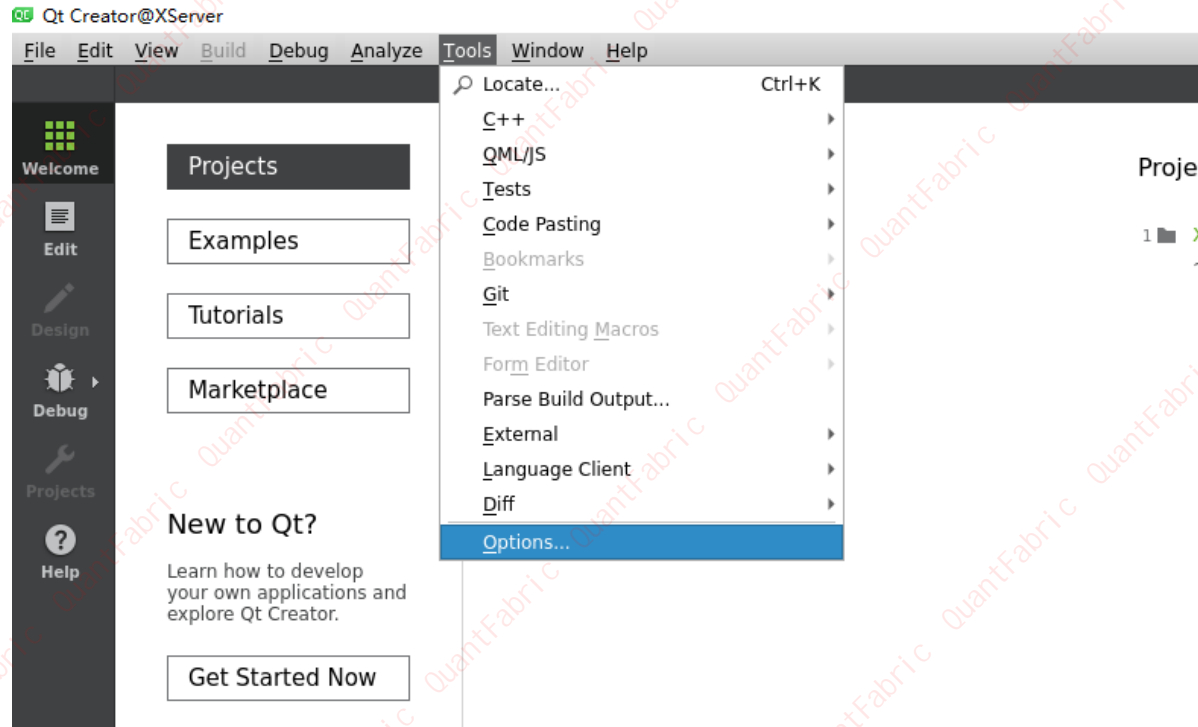
```
source .bash_profile
```


- 启动qtcreator

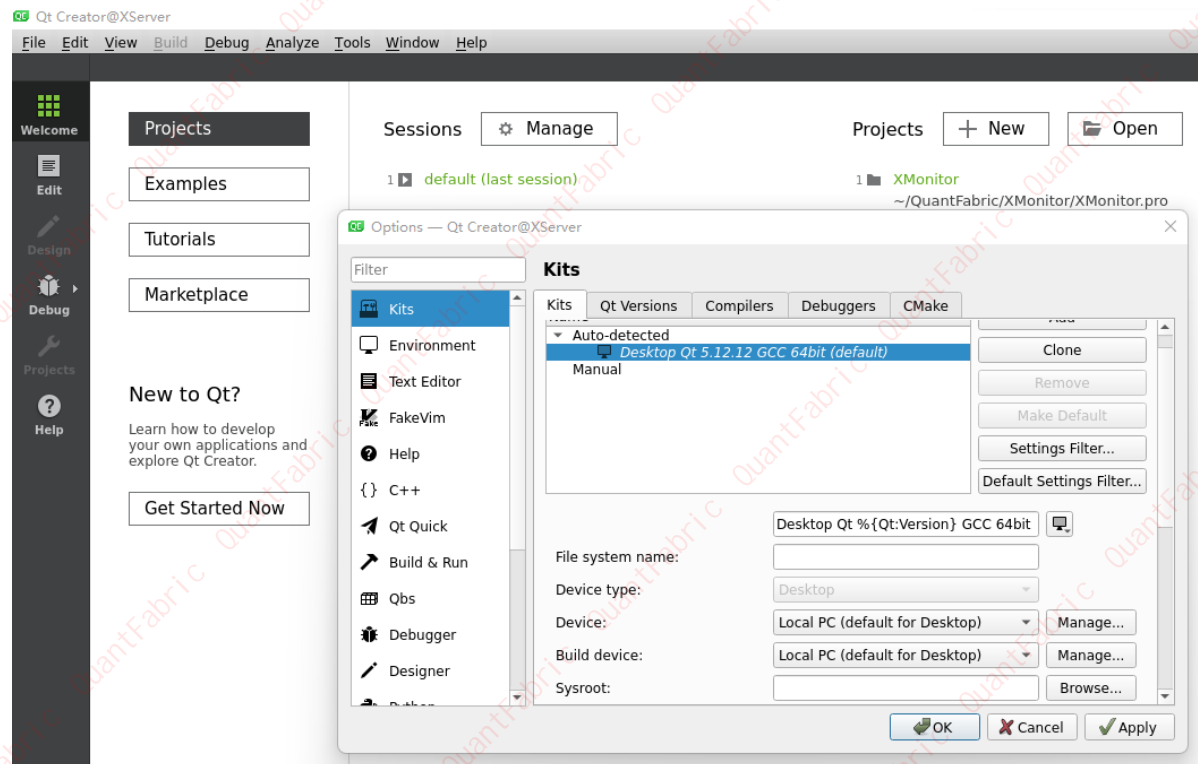
```
[xtrader@XServer ~]$ qtc
```

3、Qt开发套件配置

- 启动qtcreator



- Qt Kit配置:



- 如果Linux开发服务器在开发者内网，可以使用qtcreator进行开发调试；如果Linux开发服务器不在开发者内网，建议使用VSCode进行编码开发。
- XMonitor编译构建参看后续内容。

4、错误处理

- 执行qt-opensource-linux-x64-5.12.12.run安装程序时报错:

```
qt.qpa.xcb: could not connect to display
qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "" even though it
was found.
```

This application failed to **start** because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.

Available platform plugins are: eglfs, linuxfb, minimal, minimalegl, offscreen, vnc, xcb.

Aborted (core dumped)

- 原因: 通过MobaXterm使用root账户远程登录Linux服务器后, 使用su切换到xtrader用户, 此时xtrader账户环境缺少DISPLAY环境变量, 需要export原来root账户的DISPLAY信息。

```
export DISPLAY=localhost:10.0
```

- 通过SSH工具远程登录Linux服务器后, 如果需要执行GUI程序(使用X11转发机制), 则不能切换用户, 即只能使用SSH登录账户执行GUI程序。
- 如果遇到类似问题, 可以在MobaXterm命令行窗口执行 `ssh -vvv root@host` 查看详细debug信息。

七、CMake

- 下载CMake: [CMake 3.24.2](#)
- 下载cmake-3.24.2-linux-x86_64.sh并上传到Linux服务器:
- 将cmake-3.24.2-linux-x86_64.sh拷贝到 /usr/local/ 目录下。
- 执行安装:

```
sudo sh cmake-3.24.2-linux-x86_64.sh
```

- 更新覆盖cmake:

```
sudo rm /usr/bin/cmake
sudo ln -s /usr/local/cmake-3.24.2-linux-x86_64/bin/cmake /usr/bin/cmake
```

八、QuantFabric编译构建

1、Git子模块

- Git子模块用于解决不同开发团队之间团队协作的问题, 允许将一个Git仓库作为另一个Git仓库的子目录, 可以将另一个仓库克隆到自己项目中, 同时还保持提交的独立。

```

git submodule # 查看当前项目的子模块
git submodule init # 初始化子模块
git submodule update # 更新项目内子模块到最新版本
git submodule update --remote # 更新子模块为远程项目的最新版本
git submodule foreach 'git pull origin master' # 多个子模块遍历更新代码
git submodule add <url> <path> # 增加子模块
git submodule add -b <branch> <url> <path> # 增加子模块并切换到branch分支

```

- 子模块信息保存在主项目的 .gitmodules 文件。
- 子模块删除：
 - 删除子模块目录

```

git rm --cached XXXXX
rm -rf XXXXX

```

- 删除 .gitmodules 文件中子模块信息：

```

[submodule "XXXXX"]
    path = XXXXX
    url = https://github.com/xxxx/XXXXX.git

```

- 删除 .git/config 中子模块信息：

```

[submodule "XXXXX"]
    url = https://github.com/xxxx/XXXXX.git
    active = true

```

- 删除 .git 目录的子模块：

```

rm -rf .git/modules/XXXXX

```

2、QuantFabric编译构建

- 项目地址：[QuantFabric](https://github.com/QuantFabric/QuantFabric)
- QuantFabric量化交易系统下载：

```

git clone --recurse-submodules git@github.com:QuantFabric/QuantFabric.git

```

- 安装SQLite3开发工具包：

```

yum install sqlite-devel

```

- QuantFabric编译构建：

```

cd QuantFabric # 进入QuantFabric目录
git submodule init # 初始化子模块
git submodule update --remote # 更新子模块
sh build_release.sh # 编译构建

```

- 编译构建完成时，可执行文件和so文件位于build目录下。

3、XMonitor编译构建

- XMonitor编译构建流程：

```
cd XMonitor          # 进入XMonitor目录
git pull
git submodule init    # 初始化子模块
git submodule update --remote # 更新子模块
mkdir build
cd build
qmake ..
make
```

- 编译构建完成时，可执行文件位于build目录下。
- 由于CMake对于Qt工程构建不完美，本人仍然使用qmake对XMonitor进行单独编译构建。如果需要使用CMake构建XMonitor，请参看[CMake构建Qt工程实践](#)。