代码托管

用户指南

文档版本 01

发布日期 2024-07-25

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

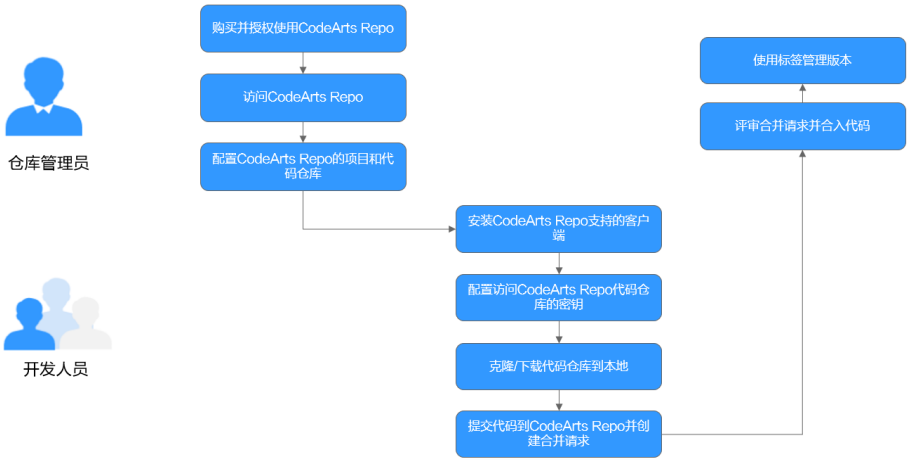
地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 代码托管(CodeArts Repo)使用流程

代码托管(CodeArts Repo)的使用流程如下图所示。



2 购买并授权使用 CodeArts Repo

前提条件

在购买CodeArts Repo前，您需要已拥有租户账号或者Tenant Administrator权限的IAM用户账号，配置权限的策略请参考**创建用户组并授权**。

购买 CodeArts Repo 套餐

步骤1 使用IAM账号**登录CodeArts Repo购买页面**。

步骤2 在购买CodeArts Repo套餐页面，参考表填写购买参数。

表 2-1 购买 CodeArts Repo 套餐参数表格

参数	说明
计费模式	该参数不可修改，默认为包年/包月。CodeArts Repo套餐的计费模式为包月或者包年。
区域	该参数必填。当前CodeArts Repo中国站支持如下局点：华北-北京一、华北-北京四、华东-上海一、华东-上海二、华南-广州、西南-贵阳一。 **说明** • 不同区域购买的资源不能跨区域使用，请谨慎选择。 • 中国站支持购买国际站区域。
产品	该参数不可修改，默认为CodeArts Repo套餐。

参数	说明
规格	该参数必填，根据您的需要，选择基础版或者专业版套餐。 • 基础版，该套餐提供以下功能：分支权限管理、代码评审、仓库配置和工作项关联。您可以使用总容量不超过50GB的代码仓库，每个仓库的容量最大为10GB，每次推送的文件大小不超过200MB，并且您可以创建任意数量的代码仓库。 • 专业版，该套餐提供以下功能：包含基础版所有功能，并且提供合并请求模板、检视意见分类及模板和星级评价。您可以使用总容量不超过500GB的代码仓库，每个仓库的容量最大为20GB，每次推送的文件大小不超过300MB，并且您可以创建任意数量的代码仓库。 **说明** 基础版更适用于个人开发者和小微型企业，专业版适用于中大型企业。
购买人数	该参数必填，根据您的需要，选择购买人数，至少1人，最多9999人。
购买时长	该参数必填，根据您的需要，选择购买时长，支持购买1~9个月、1~3年。您还可以根据需要，选择是否勾选自动续费，请参考**自动续费**规则，关于续费时长，如果您是按月购买，每次续费1个月，次数不限；如果您是按年购买：每次续费1年，次数不限。
协议	该参数必填。

步骤3 填完购买参数后，确认订单内容无误后，单击“去支付”，付款后，进入**Repo控制台页**，左上角切换到您购买的区域，可查看到购买的套餐信息。

----结束

说明

- 如果您当前正处于套餐期，无法进行购买操作。
- 如果您想要体验一站式体验一站式、全流程、安全可信的软件开发生产线(CodeArts)，您可以使用IAM账号登录**开通/购买软件开发生产线服务组合套餐**，购买CodeArts套餐，CodeArts套餐包含的Repo套餐。
- 如果您购买了10

购买资源扩展

代码托管服务支持对存储容量扩展，详情介绍请参见资源扩展。

步骤1 进入购买资源扩展页面。

步骤2 根据需要选择区域、产品、购买时长、是否自动续费相关配置项，勾选同意声明后单击“下一步：确认订单”。

步骤3 确认订单内容：如果需要修改，单击“上一步”；如果确认无误，单击“下一步”。

步骤4 根据页面提示完成支付。

----结束

3 访问 CodeArts Repo

从帮助中心首页进入 Repo 首页

步骤1 登录华为云控制台页面，选择。

步骤2 单击页面左上角 ☰ ，在服务列表中选择“开发与运维 > 代码托管CodeArts Repo”。

步骤3 CodeArts Repo页面有两种访问方式：首页入口和项目入口。

- **首页入口**

  单击“立即使用”，进入CodeArts Repo首页。该页面展示的是与当前用户相关的构建任务列表。

- **项目入口**

  a. 单击“立即使用”，进入CodeArts Repo首页。

  b. 单击导航栏“首页”。

  c. 单击需要查看的项目名称。

  d. 选择“持续交付 > 代码托管”，进入指定项目下代码仓库列表页。

----结束

4 安装 CodeArts Repo 支持的客户端

Repo当前支持的客户端及安装指导链接请参见表4-1。

表 4-1 Repo 支持的 Git 客户端

客户端名称	操作系统	官方的安装指导链接
Git客户端	Windows系统	Windows Git客户端安装指导
	Linux系统	Linux Git客户端安装指导
	Mac系统	Mac Git客户端安装指导
TortoiseGit客户端	Windows系统	Windows TortoiseGit客户端安装指导

安装好Windows Git客户端后，需要配置用户名和邮箱，在Git Bash中输入以下命令行：

```
git config --global user.name 您的用户名
git config --global user.email 您的邮箱
```

说明

- CodeArts Repo暂不支持使用github desktop进行管理。
- 用户名可以由字母、数字、常用符号组成，但是不支持以ASCII码表中小于等于32的字符以及‘.’、‘,’、‘:’、‘;’、‘<’、‘>’、‘"’、‘\\’、‘\’字符作为开头和结尾，如果首尾出现以上字符，则会直接被忽略。如为方便管理，可以考虑配置成与代码托管服务相同的用户名。
- CodeArts Repo当前支持TLS1.2和TLS1.3版本协议，在Git为最新版本的前提下，您可以执行如下命令指定TLS协议版本。其中，test.com为在CodeArts Repo下Git上传/下载的域名，tls1_2表示指定TLS协议版本为TLS1.2。不同Git客户端的解决方案您可以参考适配CodeArts Repo支持的TLS协议版本。

  ```
  openssl s_client -connect test.com:443 -tls1_2
  ```

5 配置访问 CodeArts Repo 代码仓库的密钥

5.1 密钥概述

Repo的代码仓库支持SSH和HTTPS两种访问协议，您可以选择以下两种方式之一进行配置。

- SSH密钥是一种安全的连接方式，用于在本地计算机与您账号下的Repo之间建立安全连接。不同的用户通常使用不同的计算机，因此在使用SSH方式连接Repo代码仓库前，需要在自己的电脑上生成自己的SSH密钥，并将公钥添加到Repo中。一旦在本地计算机上配置了SSH密钥，并添加公钥到Repo中，此账号下的所有代码仓库与这台计算机之间都可以使用该密钥进行连接。
- HTTPS密码是一种用于HTTPS协议方式下载、上传时使用的用户凭证。

须知

- 在Repo中，HTTPS协议所支持的单文件推送大小不超过200M。如果需要传输大于200M的文件，请使用SSH方式。
- 可以绑定邮箱的账号才能使用HTTPS协议。

- GPG(GNU Privacy Guard)是一种用于数字签名和认证的手段。当您需要将本地代码推送到代码托管仓库时，GPG公钥在Git中用于对代码的提交和Tag进行签名和验证，以确保提交的来源可信以及代码的完整性。

5.2 配置 SSH 密钥

步骤1 运行Git Bash，先检查本地是否已生成过SSH密钥。请在Git Bash中执行如下命令：

```
cat ~/.ssh/id_rsa.pub
```

- 如果提示“No such file or directory”，说明您这台计算机没生成过SSH密钥，请继续执行**步骤2**。
- 如果返回以ssh-rsa开头的字符串，说明您这台计算机已经生成过SSH密钥，如果想使用已经生成的密钥请直接跳到**步骤3**，如果想重新生成密钥，请从**步骤2**向下执行。

步骤2 生成SSH密钥。在Git Bash中生成密钥的命令如下：

```
ssh-keygen -t rsa -b 4096 -C your_email@example.com
```

其中，-t rsa表示生成的是RSA类型密钥，-b 4096是密钥长度（该长度的RSA密钥更具安全性），-C your_email@example.com表示在生成的公钥文件中添加注释，方便识别这个密钥对的用途。

如果选择ED25519算法，在Git Bash中生成密钥的命令如下：

```
ssh-keygen -t ed25519 -b 521 -C your_email@example.com
```

其中，-t ed25519表示生成的是ED25519类型密钥，-b 521是密钥长度（该长度的ED25519密钥更具安全性），-C your_email@example.com表示在生成的公钥文件中添加注释，方便识别这个密钥对的用途。

输入生成密钥的命令后，直接回车，密钥会默认存储到~/.ssh/id_rsa路径下，对应的公钥文件为~/.ssh/id_rsa.pub。

步骤3 复制SSH公钥到剪切板。请根据您的操作系统，选择相应的执行命令，将SSH公钥复制到您的剪切板。

- Windows：
  ```
  clip < ~/.ssh/id_rsa.pub
  ```
- Mac：
  ```
  pbcopy < ~/.ssh/id_rsa.pub
  ```
- Linux (xclip required)：
  ```
  xclip -sel clip < ~/.ssh/id_rsa.pub
  ```

步骤4 登录并进入Repo的代码仓库列表页，单击右上角昵称，选择“个人设置” > “代码托管” > “SSH密钥”，进入配置SSH密钥页面。

也可以在Repo的代码仓库列表页，单击右上角“设置我的SSH密钥”，进入配置SSH密钥页面。

步骤5 在**“标题”**中为您的新密钥起一个名称，将您在**步骤3**中复制的SSH公钥粘贴进**“密钥”**中，单击确定后，弹出页面“密钥已设置成功，单击 立即返回，无操作3S后自动跳转”，表示密钥设置成功。

----结束

说明

- 在一台电脑上配置了SSH密钥并添加公钥到CodeArts Repo中后，所有该账号下的代码仓库与这台电脑之间都可以使用该SSH密钥进行连接；而不同的用户通常使用不同的电脑。因此，在使用SSH方式连接CodeArts Repo之前，每个用户都需要在自己的电脑上配置各自的SSH密钥。
- 在配置SSH密钥时，提示：“此密钥已存在，请重新生成密钥”，表示该密钥在该账号或者其它账户下被添加过。解决办法：可参考如上操作步骤，在本地重新生成一次SSH密钥，再把生成的密钥配置到CodeArts Repo。

5.3 配置访问令牌

登录您的代码托管服务仓库列表页，单击右上角昵称，选择“个人设置 > 代码托管 > 访问令牌”，单击“新建Token”，参考下列表格填写参数。

表 5-1 参数说明

参数	说明
Token名称	必填参数。自定义名称，字符上限为200。
描述	非必填参数。此处描述为空，列表会显示“--”。字符上限为200。
权限	默认勾选且不可修改。读/写仓库：授予使用Https方式，读写访问仓库权限。
失效时间	必填参数。设置Token失效的时间。 **说明** 默认为当前日期的30天之后，包含当天共30天。例如7月3号新建的Token，失效日期默认为8月2号23点59分59秒。其中，失效日期最长可设置为1年，且不可为空。

填写完上述参数后，Token成功生成，请复制此Token，并在应用或脚本中使用。

须知

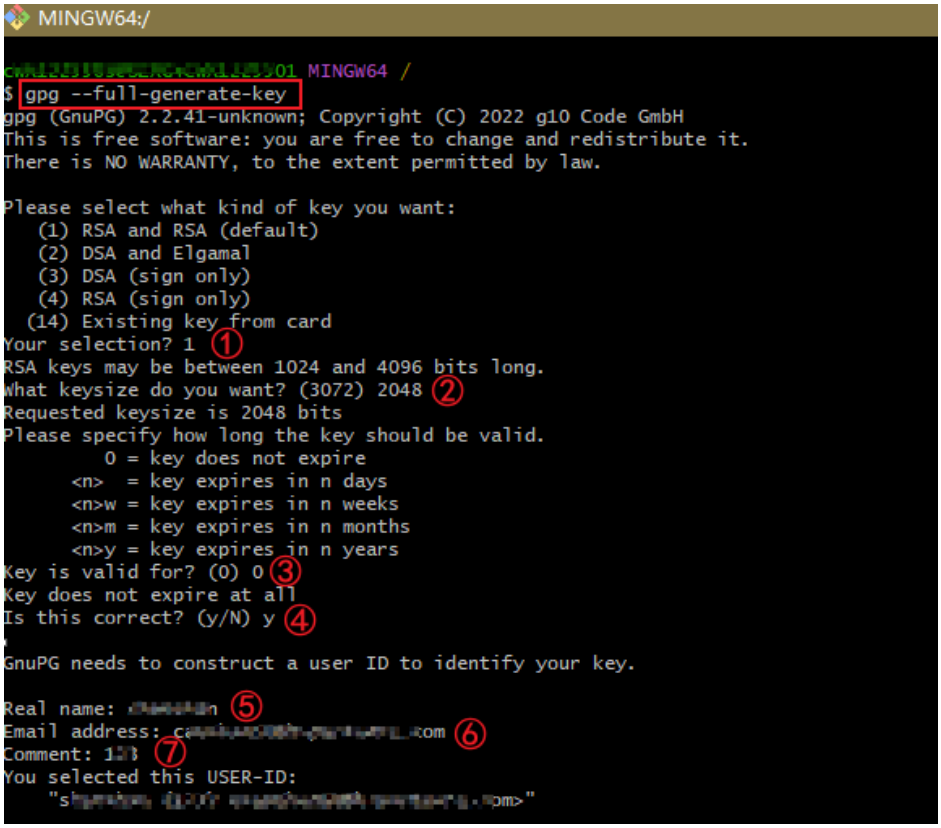
- 为保证仓库权限，关闭此弹窗后Token将不再展示，请妥善保管，如遗失或忘记可重新生成。
- CodeArts Repo生成Token数量上限为20个。

5.4 配置 GPG 公钥

步骤1 在**gpg4win官网**下载GPG密钥生成工具。

步骤2 在本地Git客户端执行**gpg --full-generate-key**命令，按照提示，依次选择加密算法、密钥长度、过期时间、正确性后，输入用户名、邮箱、注释，如**图5-1**所示。

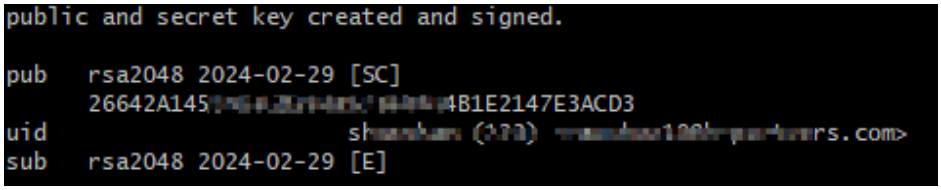
图 5-1 生成 GPG 密钥对



步骤3 确认信息无误后，按照提示，输入**O**并按回车键，弹出输入密码窗口和确认密码窗口，正确输入密码后生成密钥。

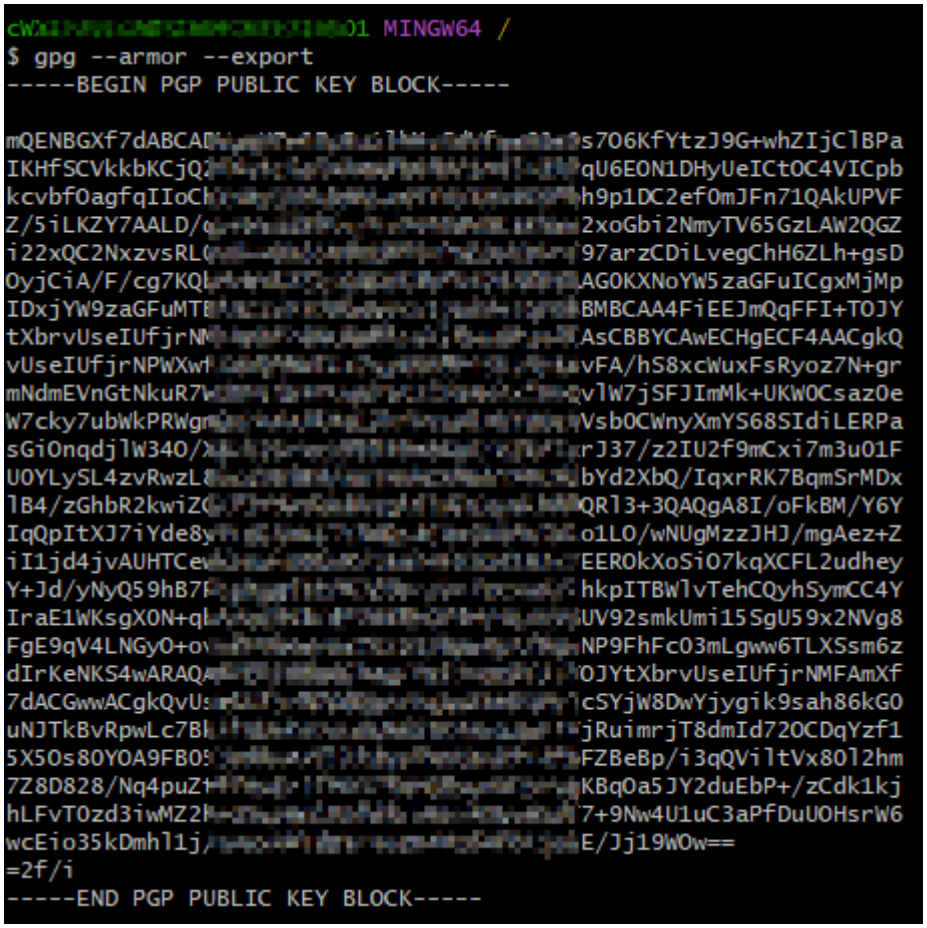
出现图5-2所示的信息，说明GPG密钥生成成功。

图 5-2 GPG 密钥生成成功



步骤4 执行**gpg --armor --export**命令导出公钥，如图4 导出GPG公钥所示，并复制公钥到剪切板，包括“-----BEGIN PGP PUBLIC KEY BLOCK-----”和“-----END PGP PUBLIC KEY BLOCK-----”。

图 5-3 导出 GPG 公钥



步骤5 登录您的代码托管服务仓库列表页，单击右上角昵称，选择“个人设置 > 代码托管 > GPG公钥”，进入GPG设置页面。

步骤6 单击“新建GPG公钥”按钮，进入新建GPG公钥页面，填写如下参数：

表 5-2 新建 GPG 公钥参数列表

参数	说明
GPG标题	必填参数。自定义GPG公钥名称，限制在200字符以内。
GPG公钥	必填参数。将**步骤4**中复制的GPG公钥粘贴到此输入框。
描述	非必填参数。当描述为空，列表显示--，字符上限为200。

步骤7 单击“确定”，成功新建GPG公钥，返回至GPG公钥列表页面。

说明

- 同一个GPG公钥不能重复使用，如果添加失败，请检查您是否已经添加过此公钥、粘贴时前后是否有多余的空格。
- 添加成功后，可以在“GPG公钥”列表页面查看到您添加的公钥，当您确认不再使用时，可以将其删除。

----结束

6 配置 CodeArts Repo 的项目和代码仓库

在CodeArts Repo进行在CodeArts首页新建CodeArts项目，操作步骤可参见需求管理。由于Repo的权限可继承CodeArts项目级权限配置。

说明

当用户被管理员从项目中移除后，该用户自动从项目下的代码仓库中移除，但SSH密钥不会被删除。此时用户将没有访问该项目和代码的权限，不能再访问项目和代码仓库。

6.1 新建并配置 CodeArts Repo 项目设置

6.1.1 配置项目级的代码仓库设置

在代码托管首页，进入项目首页，选择“设置” > “策略设置” > “仓库设置”。参数填写请参见表格表6-1。

表 6-1 项目级仓库设置参数填写表格

参数	说明
开启强制继承	非必填参数。如果勾选此参数，本项目下的所有代码组和代码仓库均使用以下参数的设置，且代码组和仓库下设置不可更改，请谨慎选择。
禁止Fork仓	非必填参数。勾选此选项，表示任何人不可以Fork该项目下的代码仓库。
MR预合并	非必填参数。勾选此选项，表示启用MR预合，服务端会自动生成MR预合并的代码，相比客户端使用命令做预合并操作更高效简洁、构建结果更准确，适用于对构建实时性要求严格的场景。

参数	说明
分支名规则	非必填参数。所有分支名都必须匹配正则表达式，分支名规则不能超过500个字符。如果此字段不填写，则允许任何分支名。规则需要满足基本的Tag命名规则： ● 不能超过500个字符。 ● 不支持以 - . refs/heads/ refs/ remotes/ 开头，不支持空格 [\ < ~ ^ : ? * ! ( ) ' " | 等特殊字符，不支持以. / .lock结尾。
Tag名规则	非必填参数。所有Tag名都必须匹配正则表达式。如果此字段不填写，则允许任何Tag名。 需满足基本的Tag命名规则： ● 不能超过500个字符。 ● 不支持以 - . refs/heads/ refs/ remotes/ 开头，不支持空格 [\ < ~ ^ : ? * ! ( ) ' " | 等特殊字符，不支持以. / .lock结尾。

配置项目级的保护分支规则

CodeArts Repo可以保护代码分支的安全性，阻止管理者外的人推送代码、阻止任何人强行推送代码或者阻止任何人删除这个分支，您可以将这个分支设置保护分支。具体具体操作过程如下：在代码托管首页，进入项目首页，选择“设置” > “策略设置” > “保护分支”，单击“新建保护分支”，请根据如下步骤填写参数：

步骤1 填写分支名称。该参数必填，请您根据自己的需要输入完整的分支名或者带通配符的分支名。如果分支中包含单斜杠（/），由于fnmatch语法规则，该分支无法用通配符“*”匹配。

步骤2 可以为管理员/项目经理、committer和开发人员设置推送或者合并的权限，两种权限不能同时拥有，原因是保护分支不能强行被推送代码或者合入代码，支持批量新建、编辑、删除保护分支。

----结束

如果您想让本项目下所有代码组和仓库均使用以上设置，勾选“开启强制继承”即可。

配置项目级的提交规则

CodeArts Repo支持为代码的提交建立校验、限制规则，以确保代码质量。在代码托管首页，进入项目首页，选择“设置” > “策略设置” > “提交规则”，单击“新建提交规则”，参数填写请参见表格**表6-2**。

表 6-2 项目级提交规则的参数填写

参数名	参数说明
规则名称	必填参数，自定义规则名称。
分支规则	必填参数，需要输入完整规则名或创建一个正则表达式。需要对输入进行校验，包括分支名的校验和正则表达式校验。
提交规则	非必填。 ● **提交信息匹配规则**：提交信息默认为空，不会对提交信息校验，任何提交信息都可以提交。若符合正则匹配，则允许提交。您也可以设置在提交信息中必须包含工作项单号，实现代码的E2E追溯，限制500个字符。 ● **提交信息负面匹配规则：**提交信息负面匹配规则默认为空，不会对提交信息校验，任何提交信息都可以提交。若符合正则匹配，则不允许提交。容限制500个字符。 ● **提交人**：提交人默认为空，不会对提交人校验，任何人都可以提交，限制200个字符。 提交人可通过“git config -l”查看user.name的值，并通过“git config --global user.name”设置user.name的值。 例如： 设置提交人规则：([a-z][A-Z]{3})([0-9]{1,9}) ● **提交人邮箱地址**：提交人邮箱地址默认为空，不会对提交人邮箱地址校验，任何邮箱地址都可以提交，限制200个字符。 提交人可通过“git config -l”查看user.email的值，并通过“git config --global user.email”设置邮箱。 例如： 设置提交人邮箱规则：@my-company.com$

参数名	参数说明
文件基本属性规则	非必填。 • **禁止提交的文件名称**：禁止提交的文件名称规则默认为空，不会对文件名校验，任何文件都可以提交，建议正则编写时使用规范的正则语句进行匹配，文件名禁用规则处默认会根据规则校验文件所属路径，限制2000个字符。 例如： 设置禁止提交的文件名称规则：(\.jar|\.exe)$ • **单文件大小限制(MB)**：文件大小上限默认显示为50，表示添加或更新文件大小超过50MB，推送将被拒绝。 **说明** 创建仓库时默认的提交规则（default）中的单文件大小限制为50MB，新建提交规则时单文件大小限制默认推荐50MB，最大不可以超过200MB。
二进制规则	非必填。 二进制规则默认不勾选，默认勾选“禁止禁止新增二进制文件（对特权用户无效）”。**“允许修改二进制文件”**勾选后，提交文件为modify状态的二进制文件不会拦截，可直接上传。二进制文件可以直接删除，不会进行二进制检查。 • 禁止新增二进制文件（对特权用户无效）。 • 允许修改二进制文件（对特权用户无效）。 • 二进制文件白名单（可直接入库的文件，限制2000个字符）。 • 特权用户（特权用户上限为50人）。 **说明** 如果特权用户已经不是仓库成员，单击保存会提示“特权用户校验失败”，需要将非仓库成员的特权用户移除才能保存成功。
规则生效时间	非必填。 在生效日期之后创建的所有提交都必须与hook设置相匹配才能被推送。如果此字段为空，则无论提交日期如何，都将检查所有提交。

配置项目级的合并请求规则

合并请求规则包含三个部分：合入机制、合入条件、MR设置和合并模式。

表 6-3 合入机制的参数说明

参数	说明
合入机制	必填参数。包含两个选项： ● **打分机制：**包含代码检视，以打分为基础，可设置最低合入分值，分值范围为0~5分。只有分数和必选评审达到门禁条件时，代码才可以合入，勾选打分机制时需设置最低分值。 ● **审核机制：**包含代码检视和合并审核两个步骤，以通过人数为基础，只有审核通过的人数达到门禁条件时，代码才可以合入。 **说明** ● 合并请求默认为“审核机制”，可手动切换为“打分机制”。 ● 修改合入机制后，会改变合并请求的工作流，但之前创建的合并请求仍保留之前的合入机制。

表 6-4 合入条件参数说明

参数	说明
合入条件	非必填参数。包括两个选项： ● 勾选“评审问题全部解决才能合入”，如果评审意见被勾选为“这是一个需要被解决的问题”，则合入条件会提示“存在未解决的评审意见”且“合入”按钮置灰；如果只是一个普通的评审意见，则不存在“已解决”开关，也不会被合入条件拦截。 ● 如果勾选“必须与CodeArts Req关联”，被关联的所有E2E单号校验必须通过；一个MR只能关联一个单号；可添加多个分支配置合并请求策略，支持手动输入通配符匹配，按回车确认，如：*-stable或production/*。

表 6-5 MR 设置参数说明

参数	说明
禁止合入自己创建的合并请求	勾选后，您在查看自己创建的MR时，“合入”按钮置灰，表示自己无法合入代码，需要找其他有合入权限的人合入。
禁止审核自己创建的合并请求	勾选后，您在查看自己创建的MR时，**“审核”**按钮置灰，自己无法审核，需要找其他有审核权限的人审核。
禁止检视自己创建的合并请求	勾选后，您在查看自己创建的MR时，**“检视”**按钮置灰，自己无法检视，需要找其他有检视权限的人检视。
允许仓库管理员强制合入	项目创建者和管理员有强制合入的权限，当合入条件不满足，也可通过**“强行合并”**按钮合入MR。
允许合并请求合并后继续做代码检视和评论	勾选后，已合入MR可继续做代码检视、评论。
是否将自动合并的MR状态标记为关闭状态（如果B MR中的所有commits都包含在A MR中，那么当A MR合并后，则B MR会自动合并。默认B MR会标记为merged状态，可以通过该选项控制将B MR标记为Closed状态）	● 未勾选时，自动合并的MR被标记为已合并。 ● 勾选后，自动合并的MR的状态将会标记为关闭状态。
不能重新打开一个已经关闭的合并请求	勾选后，当分支合并请求已经关闭后，不能将其重新置回“开启”状态，右上方的**“重开”**按钮将隐藏。 此设置一般用于流程管控，使历史评审不会被篡改。
合并请求合入后，默认删除源分支	合并成功后，源分支将被删除。 ● 已经设置成保护分支的源分支不会被删除。 ● 此设置对历史合入请求，不会生效，不必担心启用此设置会丢失分支。
禁止Squash合并	勾选后，**“Squash合并”**按钮被禁止，且合并请求中无该功能使用入口。

参数	说明
新建合并请求，默认开启Squash合并	Squash合并是指Git在做两个分支间的合并时，会把被合并分支上的所有变更“压缩（squash）”成一个提交，追加到当前分支的后面作为“合并提交”（merge commit），可以使分支变得简洁。Squash合并和普通Merge合并唯一的区别体现在提交历史上：对于普通Merge而言，在当前分支上的合并提交通常会有两个提交信息；而Squash Merge只有一个提交信息。

6.1.2 配置项目级的 CodeArts Repo 权限

步骤1 登录CodeArts Repo首页，并在左侧导航栏，选择“设置” > “通用设置” > “权限管理”，进入设置权限的页面。

步骤2 选择对应的“角色” > “代码托管”， 单击“编辑”，可设置角色的权限。

说明

1. 项目经理和其他具有管理权限的用户，可以在该页面修改不同角色在项目下的默认操作权限。
2. 可在“角色”列单击 + 创建角色，新增的角色名称不能与系统角色名称重复，但新增角色可复制已有角色的权限。新增角色如果没有复制已有角色的权限，没有任何权限，但是可根据需要添加自定义角色的权限，如表1所示。

----结束

表 6-6 设置项目级角色权限

角色/权限	操作权限	项目经理	产品经理	测试经理	运维经理	系统工程师	Committer	开发人员	测试人员	参与者	浏览者	自定义角色
分支	新建	B	C	C	C	B	B	B	C	C	D	C
	删除	B	C	C	C	B	B	B	C	C	D	C
代码	提交	B	C	C	C	A	A	A	C	C	D	C
	下载	B	C	C	C	A	A	A	C	C	D	C
代码组	新建	B	C	C	C	B	B	B	C	C	D	C

角色/权限	操作权限	项目经理	产品经理	测试经理	运维经理	系统工程师	Committer	开发人员	测试人员	参与者	浏览者	自定义角色
	删除	B	D	D	D	D	D	D	D	D	D	C
	设置	B	D	D	D	D	D	D	D	D	D	C
成员	添加	B	D	D	D	D	D	D	D	D	D	C
	修改	B	D	D	D	D	D	D	D	D	D	C
	删除	B	D	D	D	D	D	D	D	D	D	C
MR	新建	B	C	C	C	B	B	B	C	C	D	C
	编辑	B	D	D	D	C	B	C	D	D	D	C
	评论	B	C	C	C	B	B	B	C	C	C	C
	检视	B	D	D	D	B	B	B	D	D	C	C
	审核	B	D	D	D	C	B	C	D	D	D	C
	合并	B	D	D	D	C	B	C	D	D	D	C
	关闭	B	D	D	D	C	B	C	D	D	D	C
	重开	B	D	D	D	C	B	C	D	D	D	C
仓库	新建	B	C	B	C	B	B	B	C	C	D	C
	fork(MR)	B	C	B	C	B	B	B	C	C	D	C
	删除	B	D	D	D	D	D	D	D	D	D	C

角色/权限	操作权限	项目经理	产品经理	测试经理	运维经理	系统工程师	Committer	开发人员	测试人员	参与者	浏览者	自定义角色
	设置	B	D	D	D	D	D	D	D	D	D	C
Tag	新建	B	C	C	C	B	B	B	C	C	D	C
	删除	B	C	C	C	C	C	C	C	C	D	C

说明

- A：表示该角色默认拥有该权限且不可被移除。
- B：表示该角色默认拥有该权限且可被移除。
- C：表示该角色可分配到该权限。
- D：表示该角色不可分配到该权限。

6.2 新建 CoeArts Repo 的仓库

6.2.1 不同场景下新建代码仓库的区别

CodeArts Repo当前支持五种新建代码仓库的方式，您可以根据自己的使用习惯创建代码仓库。

6.2.2 新建自定义 CodeArts Repo 的代码仓库

步骤1 进入CodeArts Repo首页后，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。

步骤2 仓库类型选择“普通仓库”，根据表格填写参数。

表 6-7 新建仓库的参数说明

字段名称	说明
代码仓库名称	该参数为必填。填写该参数时，需要以大小写字母、数字、下划线开头，可包含大小写字母、数字、中划线、下划线、英文句点，但不能以.git、.atom或.结尾。
描述	该参数为非必填。该参数限制2000个字符。

字段名称	说明
选择gitignore	该参数为非必填。推荐您填写该参数，下拉框选择代码仓库要开发的编程语言，可以避免后续不必要的文件被跟踪，以此保持代码库的整洁和可维护性。
初始化设置	该参数为非必填。包括两个选项： • 允许生成README文件。推荐您勾选该选项，生成该文件后，您可以通过编辑README文件，记录项目的架构、编写目的等信息，帮助其他人更快了解该代码仓。 • 自动创建代码检查任务（免费）。推荐您勾选该选项，代码仓库创建完成后，在代码检查(CodeArts Check)任务列表中，可看到对应仓库的检查任务。
可见范围	该参数为非必填。您可根据自己的需求进行选择，包括两个选项： • 私有(仓库仅对仓库成员可见，仓库成员可访问仓库或者提交代码)。 • 公开只读(仓库对所有访客公开只读，但不出现在访客的仓库列表及搜索中)。 **说明** 代码仓库可以相互转换“私有”或者“公开”，进入要设置的代码仓库详情页面，选择“设置 > 基本设置 > 仓库信息”，修改代码仓库的可见范围。
添加开源许可证	该参数在“可见范围”选择为“公开只读”时必填。下拉框选择已有的许可证。

----结束

6.2.3 按模板新建 CodeArts Repo 的代码仓库

步骤1 进入CodeArts Repo首页后，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。

步骤2 仓库类型选择“模板仓库”，这里可选择“官方模板”或“个人模板”。其中，个人模板需要进入仓库设置，将官方模板设置为个人模板。选择模板后，根据**表格**填写参数。

----结束

6.2.4 导入 Git 平台的代码仓库到 CodeArts Repo

在线导入 Git 平台的代码仓库到 CodeArts Repo

步骤1 进入CodeArts Repo首页后，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。

步骤2 仓库类型选择“导入仓库”，导入方式选择“Git Url”，参数填写请参考**表6-8**。

表 6-8 导入 Git 平台代码仓库的参数表格

字段名称	说明
源仓库路径	该参数必填，该参数表示要导入的仓库路径。源仓库路径需要以（http://）或（https://）开头，以（.git）结尾。 **说明** • 如果仓库过大或者网络较差时，仓库导入时间可能会超过30min。如果出现导入超时，建议使用客户端clone/push来处理，具体可参考**导入外部仓库提示超时**。 • 该功能需要保证被导入的仓库域名和服务节点网络连通。
可见范围	该参数为非必填。该参数表示源仓库的可见范围，包括两个选项： • 公开只读(仓库对所有访客公开只读，但不出现在访客的仓库列表及搜索中)。当选择公开只读时，还需要填写“同步仓库设置”参数，具体可参考**表**。 • 私有(仓库仅对仓库成员可见，仓库成员可访问仓库或者提交代码)。当选择私有时，还需要填写： 1. 用户名和密码/AccessToken。 2. 填写“分支设置”参数，具体可参考**表**。
用户名	当代码仓库选择为私有时，该参数必填。该参数表示https克隆代码时的用户名，例如为GitHub的登录名称。
密码/Access Token	当代码仓库选择为私有时，该参数必填。该参数表示https克隆代码时的用户名，例如为GitHub的登录名称。该参数的获取方式请参考**获取Access Token**。

字段名称	说明
初始化设置	该参数为非必填。包括两个选项： • 允许生成README文件。推荐您勾选该选项，生成该文件后，您可以通过编辑README文件，记录项目的架构、编写目的等信息，帮助其他人更快了解该代码仓。 • 自动创建代码检查任务（免费）。推荐您勾选该选项，代码仓库创建完成后，在代码检查(CodeArts Check)任务列表中，可看到对应仓库的检查任务。

表 6-9 同步仓库设置的参数表格

字段名称	说明
分支设置	该参数必填。包括两个选项： • 默认分支。指新建代码仓库时自动创建的主分支，例如master分支。 • 全部分支。指代码仓库中的所有分支，包括默认分支及其他自定义分支。

----结束

说明

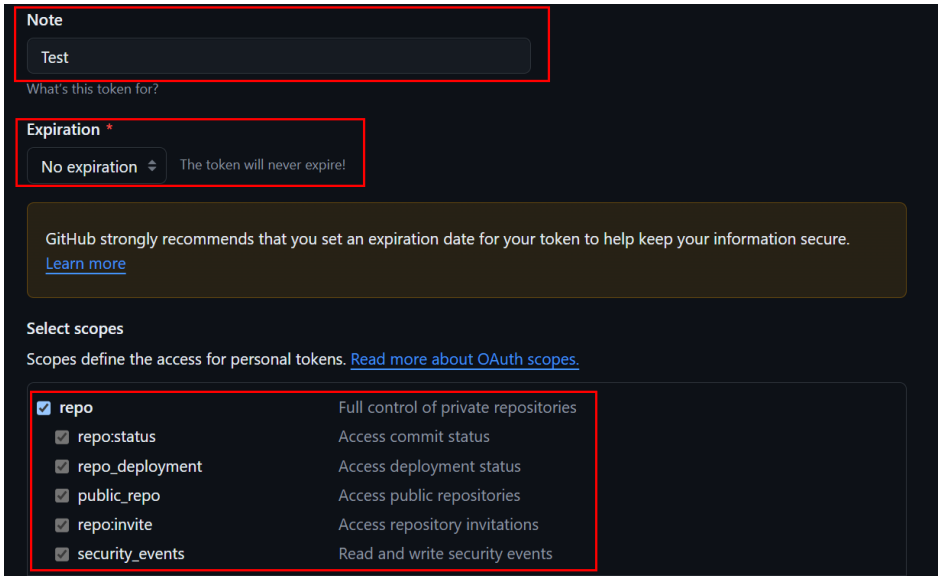
- 填写完参数后，会自动跳转到“我创建的”代码仓列表页面，如果新建代码仓库名称颜色为灰色，且仓库名称旁有红色感叹号，表示该仓库导入失败，可能原因：用户名或者密码/Access Token错误。可以将该代码仓删除，按照如上步骤操作，重新导入外部仓库。
- 当前Git支持的外部导入源包括：bitbucket.org、code.aliyun.com、coding.net、git.qcloud.com、gitee.com、github.com、gitlab.com、visualstudio.com、xiaolvyun.baidu.com。

获取 Access Token

步骤1 登录GitHub，单击右上角头像，选择“Settings” > “Developer settings”。

步骤2 选择“Personal access tokens ” > “Personal access tokens (classic)” > “Generate new token (classic)”，填写关键信息，如下图所示。

图 6-1 填写“新建 Token”的关键信息



步骤3 填写好关键信息，完成Token的新建，并自动跳转到新建的Token页面，由于该Token是临时生成的，请复制并保存该Token。

----结束

6.2.5 导入 SVN 平台的代码仓库到 CodeArts Repo

在线导入 SVN 平台的代码仓库到 CodeArts Repo

步骤1 进入CodeArts Repo首页后，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。

步骤2 仓库类型选择“导入仓库”，导入方式选择“SVN”。

表 6-10 导入 SVN 平台代码仓库的参数表格

字段名称	说明
源仓库路径	该参数必填，该参数表示要导入的仓库路径。源仓库路径需要以（ http:// ）开头。 **说明** • 如果仓库过大或者网络较差时，仓库导入时间可能会超过30min。如果出现导入超时，建议使用客户端clone/push来处理，具体可参考**通过Git Bash导入SVN平台的代码仓库到CodeArts Repo**。 • 在线导入的操作方式简单，且将SVN中的分支、Tags进行平移，如果后续想在此代码仓的基础上继续开发，请利用Git Bash客户端导入，具体可参考**通过Git Bash导入SVN平台的代码仓库到CodeArts Repo**。 • 该功能需要保证被导入的仓库域名和服务节点网络连通。
可见范围	该参数为非必填。该参数表示源仓库的可见范围，包括两个选项： • 公开只读(仓库对所有访客公开只读，但不出现在访客的仓库列表及搜索中)。当选择公开只读时，还需要填写“同步仓库设置”参数。 • 私有(仓库仅对仓库成员可见，仓库成员可访问仓库或者提交代码)。当选择私有时，还需要填写： 1. 用户名和密码/AccessToken。 2. 填写“分支设置”参数。
用户名	当代码仓库选择为私有时，该参数必填。该参数表示https克隆代码时的用户名，例如为GitHub的登录名称。
密码/Access Token	当代码仓库选择为私有时，该参数必填。该参数表示https克隆代码时的用户名，例如为GitHub的登录名称。该参数的获取方式请参考**获取Access Token**。

字段名称	说明
初始化设置	该参数为非必填。包括两个选项： • 允许生成README文件。推荐您勾选该选项，生成该文件后，您可以通过编辑README文件，记录项目的架构、编写目的等信息，帮助其他人更快了解该代码仓。 • 自动创建代码检查任务（免费）。推荐您勾选该选项，代码仓库创建完成后，在代码检查(CodeArts Check)任务列表中，可看到对应仓库的检查任务。

----结束

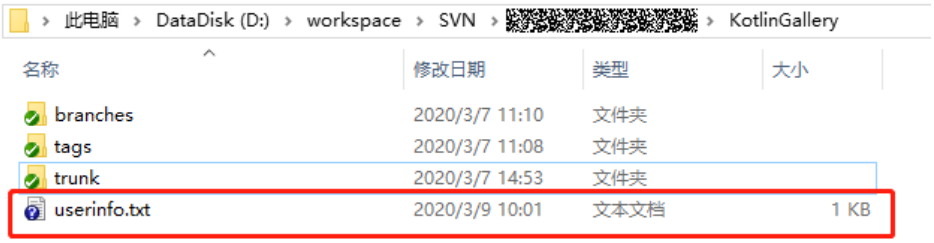
通过 Git Bash 导入 SVN 平台的代码仓库到 CodeArts Repo

步骤1 获取SVN代码库提交者信息。

1. 通过TortoiseSVN将待迁移的代码仓库下载到本地。
2. 进入本地SVN代码仓库（本文为KotlinGallery），在Git Bash客户端执行如下命令：

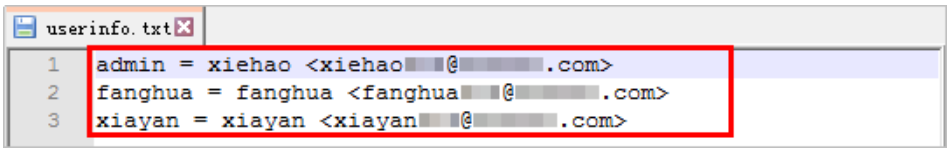
   ```
   svn log --xml | grep "^<author" | sort -u | \awk -F '<author>' '{print $2}' | awk -F '</author>' '{print $1}' > userinfo.txt
   ```

   执行完毕后，“KotlinGallery”目录下将生成文件“userinfo.txt”，如下图所示。

3. 打开文件“userinfo.txt”，可看到文件中显示所有对该仓库有提交操作的提交者信息。
4. 因为Git是用邮箱来标识一个提交者的，为了更好的将SVN已有的信息映射到Git仓库里，需要从SVN用户名到Git作一个映射关系。

   修改“userinfo.txt”，使每一行中，svn作者 = Git作者昵称 <邮箱地址>，映射关系的格式如下图所示。

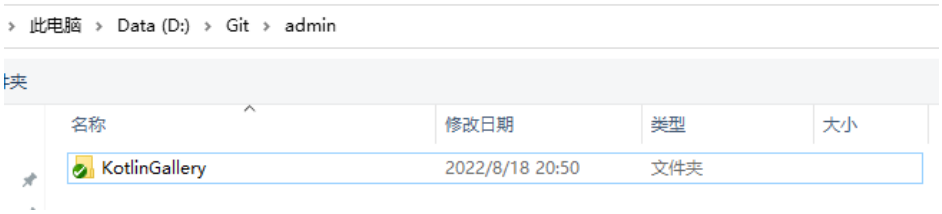
步骤2 建立本地Git仓库。

1. 执行命令**git init**，在本地新建一个空的Git代码仓库目录。

2. 将步骤1中的“userinfo.txt”文件拷贝到该目录下，并执行如下命令切换到该目录下。
```
cd 目标目录地址
```
3. 在该目录下启动Git Bash客户端，并执行如下命令克隆一个Git版本库。
```
git svn clone SVN仓库地址 --no-metadata --authors-file=userinfo.txt --trunk=trunk --tags=tags --branches=branches
```
命令行中的参数说明如下，请根据实际情况选择相应参数:

参数	说明
--no-metadata	表示不将 SVN 的元数据导入到Git仓库中，这样可以减小Git代码仓库的大小，但是可能会丢失一些SVN的历史信息。
--authors-file=userinfo.txt	表示使用指定的用户信息文件来进行作者信息的映射。
--trunk=trunk	表示将SVN仓库中的“trunk”分支作为Git代码仓库的主分支。
--tags=tags	表示将SVN代码仓库中的tags目录作为Git代码仓库的标签。
--branches=branches	表示将SVN代码仓库中的branches目录作为Git代码仓库的分支。

执行成功后，本地将生成一个名为KotlinGallery的Git代码仓库。



4. 执行以下命令，进入“KotlinGallery”文件夹，并验证当前Git仓库分支结构。
```
cd KotlinGallery
git branch -a
```


如上图所示，所有SVN中的目录结构均以Git分支的形式迁移成功。

步骤3 本地分支修正。

因此在上传到代码托管仓库前，需要先对本地分支进行调整，使之符合Git使用规范。

1. 进入本地Git代码仓库目录下，在Git Bash客户端执行如下命令，把Tags分支变成合适的Git标签。


```
cp -Rf .git/refs/remotes/origin/tags/* .git/refs/tags/
rm -Rf .git/refs/remotes/origin/tags
git branch -a
git tag
```

```
MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ cp -Rf .git/refs/remotes/origin/tags/* .git/refs/tags/

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ rm -Rf .git/refs/remotes/origin/tags

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git branch -a
* master
  remotes/origin/r1.1_hotfix
  remotes/origin/trunk

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git tag
r1.0
r1.1
```

2. 执行以下命令，把“refs/remotes”下面剩下的索引变成本地分支。

```
cp -Rf .git/refs/remotes/origin/* .git/refs/heads/
rm -Rf .git/refs/remotes/origin
git branch -a
git tag
```

```
MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ cp -Rf .git/refs/remotes/origin/* .git/refs/heads/

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ rm -Rf .git/refs/remotes/origin

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git branch -a
* master
  r1.1_hotfix
  trunk

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git tag
r1.0
r1.1
```

3. 执行以下命令，将trunk分支合入master分支，并删除trunk分支。

```
git merge trunk
git branch -d trunk
git branch -a
git tag
```

```
MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git merge trunk
Already up to date.

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git branch -d trunk
Deleted branch trunk (was bccf0d8).

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git branch -a
* master
  r1.1_hotfix

MINGW64 /d/workspace/Git/admin/KotlinGallery (master)
$ git tag
r1.0
r1.1
```

步骤4 上传本地代码仓到CocdeArts Repo。

1. 参考**配置SSH密钥**，设置代码仓库的SSH密钥。
2. 进入CodeArts Repo首页，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。
3. 仓库类型选择“普通仓库”，填写对应参数信息并去勾选“允许生成README文件”和“选择gitignore”，完成新的代码仓库创建，并自动跳转到该代码仓库首页。
4. 选择右上角的“克隆/下载” > “用HTTPS克隆”，复制HTTPS地址。
5. 执行以下命令，将本地代码仓库与CodeArts Repo进行关联，并推送master分支到CodeArts Repo的代码仓库。在执行命令时，需要您输入CoeArts Repo的HTTPS账号和密码。
   ```
   git remote add origin 新建的代码仓库的HTTPS地址
   git push --set-upstream origin master
   ```
   推送成功后，进入该代码仓库首页，选择“代码” > “分支”，查看到当前代码仓库下的master分支。
6. 继续执行以下命令，把本地其余分支推送到CodeArts Repo。
   ```
   git push origin --all
   ```
   推送成功后，进入该代码仓库首页，选择“代码” > “分支”，代码仓库下新增了r1.1_hotfix分支。
7. 执行以下命令，从本地推送tags到CodeArts Repo。
   ```
   git push origin --tags
   ```
   推送成功后，进入该代码仓库首页，选择“代码” > “Tags”，代码仓库下已有标签“r1.0”与“r1.1”。

----结束

6.2.6 把本地自建的代码仓库推送到 CodeArts Repo

如果您的代码仓还没有纳入过任何的版本系统，如Git或者SVN，在源代码的根目录，执行如下操作，把本地自建的代码仓导入到CodeArts Repo。

步骤1 进入CodeArts Repo首页，单击“新建仓库”，在“归属项目”下拉框中选择已有的项目或者“新建项目”。

步骤2 仓库类型选择“普通仓库”，填写对应参数信息并去勾选“允许生成README文件”和“选择gitignore”，完成新的代码仓库创建，并自动跳转到该代码仓库首页。

步骤3 执行命令**git init**，在本地新建一个空的Git代码仓库目录。

步骤4 执行命令**git add ***，将文件加入版本库。

步骤5 执行命令**git commit -m "init commit"**，创建初始提交。

----结束

说明

如果CodeArts Repo的仓库容量快满的时候，您可以进入代码仓库详情页，使用如下的方法清理代码仓库资源：

- 选择“代码 > 分支”，选择不需要的分支，单击 ，删除不需要的分支。
- 选择“代码 > 标签”，选择不需要的标签，单击 ，删除不需要的标签。
- 选择“设置 > 仓库管理 > 仓库加速”，清除缓存数据。
- 选择“设置 > 仓库管理 > 子模块设置”，删除不需要的子模块。

6.2.7 配置代码仓库级的权限

仓库权限矩阵仅支持管理员修改，项目管理员及各层父级代码组和仓库所有者可作为管理员。在确认您是管理员的前提下，进入代码托管首页，单击要设置的代码仓名称，进入代码仓的详情页，单击导航栏的“成员”，可为代码仓添加成员。完成代码仓的成员配置，单击导航栏的“设置”，进入仓库设置页面，选择“安全管理” > “权限管理”，若开启“使用项目级权限配置”，当前角色列表成员的权限将与项目权限保持一致，且会覆盖当前的权限配置。

单击右侧的  ，可同步项目自定义角色，自定义角色默认没有仓库的操作的权限，同步后，可根据需要添加表6-11所示的权限。

表 6-11 配置代码仓角色权限

角色/权限	操作权限	项目经理	产品经理	测试经理	运维经理	系统工程师	Committer	开发人员	测试人员	参与者	浏览者	自定义角色
仓库	fork	B	C	B	C	B	B	B	C	C	D	C
	删除	B	D	D	D	D	D	D	D	D	D	C
	设置	B	D	D	D	D	D	D	D	D	D	C
代码	提交	B	C	C	C	A	A	A	C	C	D	C
	下载	B	C	C	C	A	A	A	C	C	D	C
成员	添加	B	D	D	D	D	D	D	D	D	D	C
	修改	B	D	D	D	D	D	D	D	D	D	C
	删除	B	D	D	D	D	D	D	D	D	D	C
分支	新建	B	C	C	C	B	B	B	C	C	D	C
	删除	B	C	C	C	B	B	B	C	C	D	C
Tag	新建	B	C	C	C	B	B	B	C	C	D	C
	删除	B	C	C	C	C	C	C	C	C	D	C

角色/权限	操作权限	项目经理	产品经理	测试经理	运维经理	系统工程师	Committer	开发人员	测试人员	参与者	浏览者	自定义角色
MR	新建	B	C	C	C	B	B	B	C	C	D	C
	编辑	B	D	D	D	C	B	C	D	D	D	C
	评论	B	C	C	C	B	B	B	C	C	C	C
	检视	B	D	D	D	B	B	B	D	D	C	C
	审核	B	D	D	D	C	B	C	D	D	D	C
	合并	B	D	D	D	C	B	C	D	D	D	C
	关闭	B	D	D	D	C	B	C	D	D	D	C
	重开	B	D	D	D	C	B	C	D	D	D	C

说明

- A：表示该角色默认拥有该权限且不可被移除。B：表示该角色默认拥有该权限且可被移除。C：表示该角色可分配到该权限。D：表示该角色不可分配到该权限。
- 关于公开仓库的权限矩阵，默认添加下载权限与评论权限，且不可编辑，其他权限与私仓默认权限一致。

6.2.8 配置 CodeArts Repo 代码仓库设置

配置代码仓库级的仓库设置

如果在项目级“仓库设置”勾选了“开启强制继承”，代码仓库下不支持“仓库设置”。

如果不继承项目级配置，可参考下表设置参数。

表 6-12 代码仓库级的仓库设置参数填写表格

参数	说明
默认分支管理	此参数非必填。默认将“master”分支设置为默认分支，即创建代码仓库时的主分支。

参数	说明
开启开发人员创建分支权限白名单	此参数非必填。默认不勾选，勾选后，开启开发人员创建分支权限白名单，只有开发人员角色的仓库成员才能进入此白名单。非开发人员将不会被显示，并且即使配置后也不会生效。
禁止Fork仓	非必填参数。勾选此选项，表示任何人不可以Fork该项目下的代码仓库。
MR预合并	非必填参数。勾选此选项，表示启用MR预合，服务端会自动生成MR预合并的代码，相比客户端使用命令做预合并操作更高效简洁、构建结果更准确，适用于对构建实时性要求严格的场景。
分支名规则	非必填参数。所有分支名都必须匹配正则表达式，分支名规则不能超过500个字符。如果此字段不填写，则允许任何分支名。规则需要满足基本的Tag命名规则： ● 不能超过500个字符。 ● 不支持以 - . refs/heads/ refs/ remotes/ 开头，不支持空格 [\ < ~ ^ : ? * ! ( ) ' " | 等特殊字符，不支持以. / .lock结尾。
Tag名规则	非必填参数。所有Tag名都必须匹配正则表达式。如果此字段不填写，则允许任何Tag名。 需满足基本的Tag命名规则： ● 不能超过500个字符。 ● 不支持以 - . refs/heads/ refs/ remotes/ 开头，不支持空格 [\ < ~ ^ : ? * ! ( ) ' " | 等特殊字符，不支持以. / .lock结尾。

配置代码仓库级的保护分支规则

如果勾选“继承项目设置”，代码仓库下不支持再次“新建保护分支”，代码仓库下的成员均可执行该操作。

如果不继承项目级配置，可参考下表设置参数。

表 6-13 新建保护分支的参数表格

参数名称	参数解释
选择需要添加的保护分支	根据自己的需要输入完整的分支名或者带通配符的分支名。 仅支持单个添加，不支持批量添加。要求以“refs/heads/”开头，结尾可以有“*”，其它位置不可以出现特殊字符。
添加权限	该参数非必填。支持对管理员/项目经理、Committer和开发人员添加如下权限： ● 推送。拥有该权限，可推送Commit到保护分支。 ● 合并。拥有该权限，可以对该保护分支合入合并请求。 **说明** 如果打开推送权限，默认同步打开合并权限，且不可单独关闭合并权限。

配置代码仓库级的保护 Tag 规则

进入要设置设置的代码仓库首页，，选择“设置 > 策略设置 > 保护Tags”，单击“新建保护Tag”，参考下表填写配置参数。

表 6-14 新建保护 Tag 参数说明

参数	说明
选择需要保护的Tag	该参数必填。根据自己的需要输入完整的Tag或者带通配符的Tag。 仅支持单个添加，不支持批量添加。要求以“refs/heads/”开头，结尾可以有“*”，其它位置不可以出现特殊字符。
允许创建	该参数必填。表示添加“允许创建保护Tag”的角色。您可以在下拉框选择允许创建的角色。

配置代码仓库级的提交规则

CodeArts Repo支持为代码的提交建立校验、限制规则，以确保代码质量，您可以勾选“继承项目设置”，自动继承并使用项目下设置且不支持更改。您也可以进入要配置的代码仓库首页，选择“设置” > “策略设置” > “提交规则”，单击“新建提交规则”，参数填写请参见表格表6-2。

配置代码仓库的子模块设置

子模块（submodule）是Git为管理仓库共用而衍生出的一个工具，通过子模块您可以将公共仓库作为子目录包含到您的仓库中，并能够双向同步该公共仓库的代码，借助子模块您能将公共仓库隔离、复用，能随时拉取最新代码以及对它提交修复，能大大提高您的团队效率。

有种情况经常会遇到：某个工作中的项目A需要包含并使用项目B（第三方库，或者你独立开发的，用于多个父项目的库），如果想要把它们当做两个独立的项目，同时又想在项目A中使用项目B，可以使用Git的子模块功能。 子模块允许您将一个Git仓库作为另一个Git仓库的子目录。 它能让你将另一个仓库克隆到自己的项目中，同时还保持提交的独立。

子模块将被记录在一个名叫“.gitmodules”的文件中，其中会记录子模块的信息：

```
[submodule "module_name"]      #子模块名称
path = file_path               #子模块在本仓库（父仓）中文件的存储路径。
url = repo_url                 #子模块（子仓库）的远程仓地址
```

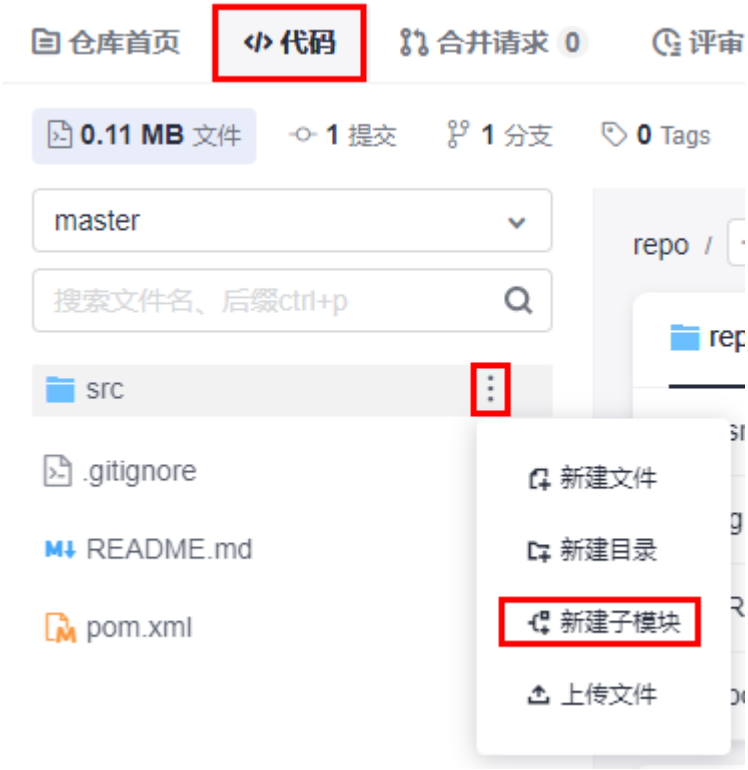
这时，位于“file_path”目录下的源代码，将会来自“repo_url”。

控制台操作

- **控制台添加子模块**
  - **入口一：**

    可以在仓库文件列表中的某个文件夹下添加子模块。

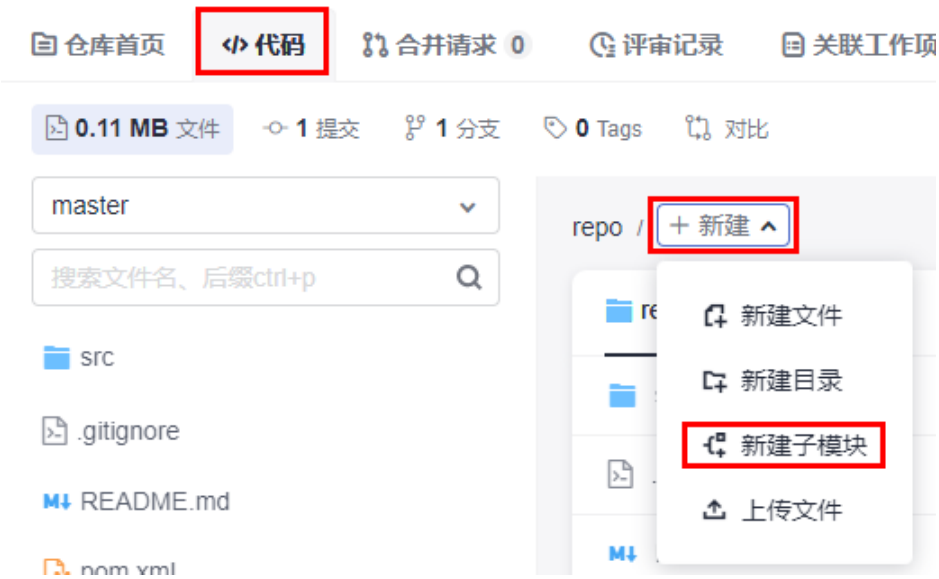
    单击扩展按钮 ⋮，选择“新建子模块”，如下图所示。

  - **入口二：**

    可以在仓库的**“代码”**页签中，添加子模块。

    单击扩展按钮 + 新建 ，选择“新建子模块”，如下图所示。



- **入口三：**
  可以在仓库设置中，为仓库创建子模块。
  其操作路径为“设置 > 仓库管理 > 子模块设置 > 新建子模块”。
- **填写说明：**
  使用以上三种方法均可进入“新建子模块”页面。
  请参考下表填写，完成后单击“确定”按钮，即可完成新建子仓库操作。

表 6-15 新建子模块—字段说明

字段	填写说明
子模块仓库路径	选择一个仓库作为子仓库。
子模块仓库分支	选择同步子仓库的目标分支到父仓库。
子模块文件路径	配置子模块文件在本仓库下的路径，注意用“/”分割层级。
提交信息	作为您新建子仓库的备注信息，可以在文件历史中查找到本次操作，限制2000个字符。

说明

子模块新建完成后，可以在仓库文件列表的对应目录内找到子模块（子仓库）内容，其对应的文件左侧图标为 。

- **控制台查询子模块状态、同步、删除子模块**
  管理员可以通过查看“设置”页面下的“子模块设置”页面，查看子模块状态，同步子模块，删除子模块。

- **控制台同步部署密钥**

  对于客户端提交的子模块，需要仓库管理员在“设置”页面下的“子模块设置”页面，将父仓库的部署密钥同步到子仓库中，从而保证在构建父仓库时，可以将对应提交的子仓库一同拉取下来。

Git客户端操作

步骤1 添加Submodule。

```
git submodule add <repo> [<dir>] [-b <branch>] [<path>]
```

示例：

```
git submodule add git@***.***.com:****/WEB-INF.git
```

步骤2 拉取包含submodule的仓库。

```
git clone <repo> [<dir>] --recursive
```

示例：

```
git clone git@***.***.com:****/WEB-INF.git --recursive
```

步骤3 获取远端Submodule更新。

```
git submodule update --remote
```

步骤4 推送更新到子库。

```
git push --recurse-submodules=check
```

步骤5 删除Submodule。

1. 删除“.gitsubmodule”中对应submodule的条目。
2. 删除“.git/config”中对应submodule的条目。
3. 执行命令，删除子模块对应的文件夹。

   ```
   git rm --cached {submodule_path}    #注意更换为您的子模块路径
   ```

   说明

   注意：路径不要加后面的“/”。

   示例：你的submodule保存在“src/main/webapp/WEB-INF/”目录，则执行命令为：

   ```
   git rm --cached src/main/webapp/WEB-INF
   ```

----结束

更多详情请参见官方文档**Git 工具 - 子模块**。

7 管理 CodeArts Repo 的代码仓库

配置代码仓库间的同步设置

CodeArts Repo支持将当前仓库设置自定义同步至其他仓库，当前功能仅支持跨项目同步，暂时不支持跨区域同步。

一般推荐用于基于该仓库Fork出的仓库，因为Fork仓库时虽然会复制其所有分支和文件内容，但并不会自动复制仓库设置。

如果您已开启继承设置后，无法使用同步设置。

仅有仓库的“设置”权限成员可以执行此操作，仓库内的仓库成员可以查看该页面。

进入要设置的代码仓库首页，选择 “设置 > 仓库管理 > 同步设置”。单击“添加仓库”，在弹框中选择目标仓库。

说明

- 同步仓库需保证网络连通。
  - 对于公开平台，CodeArts Repo支持访问代码仓库。
  - 对于连接内网私有仓库平台，用户需自行保证CodeArts Repo到用户仓库的网络畅通。
- 常见的同步失败原因：
  1. **“提交规则”**同步失败：一般是因为源仓库没有设置提交规则。
  2. **“保护分支”**同步失败：一般是因为源仓库与目标仓库的分支命名不一样。

配置 Webhook 设置

开发人员可在Webhook界面配置第三方系统的URL，并根据项目需求订阅代码托管仓库的分支推送(push)、标签推送(tag push)等事件。当订阅事件发生时，可通过Webhook向第三方系统的URL发送 POST请求，用以触发自己系统（第三方系统）的相关操作，例如：触发自己系统（第三方系统）界面的通知弹窗；或触发自己系统（第三方系统）的构建、更新镜像、部署等操作。

Webhook设置位于仓库详情中的 “设置 > 服务集成 > Webhook设置”。

此设置只针对被设置的仓库生效。仓库内的仓库成员可以查看该页面。

表 7-1 新建 Webhook 字段说明

字段	说明
名称	可自定义名称。
描述	用于描述该WebHook。
URL	必填项。WebHook URL需第三方CI/CD系统提供。
Token类型	用于第三方服务WebHook接口鉴权，分为以下三项： ● X-Repo-Token ● X-Gitlab-Token ● X-Auth-Token
Token	用于第三方CI/CD系统鉴权, 鉴权信息放在http请求header。
事件类型	系统可订阅以下事件： ● 推送事件 – 如果勾选推送事件，则出现**分支过滤正则规则**。 **说明** **分支过滤正则规则**，默认为.*，代表全部分支，长度上限不超过500字符。 **分支过滤正则规则**需符合正则表达式。 – 在代码托管仓库进行代码更新，如LFS文件代码更新、子模块中代码更新、在线或本地Git客户端中推送代码更新均会触发该事件。 ● Tag推送事件 在代码托管仓库新建或删除Tag会触发该事件。 ● 合并请求事件 – 在代码托管仓库新建合并请求会触发该事件。 – 在代码托管仓库更新合并请求会触发该事件。如更新代码内容/更新合并请求状态（关闭、重开）/更新合并请求标题或描述/更新合并人/更新工作项/删除源分支/更新Squash合并。 – 在代码托管仓库合入合并请求会触发该事件。 ● 评论事件 – 在代码托管仓库添加检视意见会触发该事件。如在代码文件中添加检视意见、在提交详情文件变更下添加检视意见、在合并请求文件变更中添加检视意见。 – 在代码托管仓库提交详情和在合并请求详情中添加评论会触发事件。

说明

- 每个仓库最多只能设置20个Webhook。
- 您在配置Webhook的时候，还可以选择设置您的Token，该Token会与您的Webhook URL关联，系统会将该Token放在请求头的X-Repo-Token字段发送给您。

8 克隆/下载代码仓库到本地

8.1 使用不同方式克隆/下载代码仓库的区别

克隆代码仓和下载代码仓都是获取代码仓库的方式，但是它们的具体操作和效果有所不同。

1. 克隆代码仓库到本地

   使用SSH密钥或者HTTPS协议克隆代码仓，是将整个代码仓库的内容复制到本地计算机上，并创建一个本地仓库，这个本地仓库包含了完整的代码提交历史记录、分支(Branches)、标签(Tags)，可以进行版本控制和修改。当前Repo支持使用Git Bash 和TortoiseGit客户端克隆代码仓，在通过SSH密钥克隆Repo的代码仓库之前，需要**配置访问Repo的SSH密钥**。

2. 下载代码仓库

   下载代码仓则是将代码仓库中的某个或某些文件或文件夹下载到本地计算机上，并不包含完整的代码提交历史记录、分支(Branches)、标签(Tags)，不能进行版本控制和修改。当前Repo支持通过浏览器下载代码。

因此，如果需要对代码仓库进行版本控制和修改，您需要选择使用SSH密钥或者HTTPS协议克隆代码仓库；如果您只需要获取代码仓库的某个或者某些文件，可以选择使用浏览器下载代码仓库。

须知

- 如果要克隆代码仓库并在本地进行代码开发，请先在CodeArts Repo进入要克隆的代码仓库主页，选择“分支” > “新建分支”，基于主分支创建一条属于您的开发分支。
- CodeArts Repo当前仅支持一次克隆一个代码仓库，如果想要一次克隆多个代码仓库到本地，您可以通过Shell或者批处理命令实现多个仓库下载。

8.2 使用 SSH 密钥克隆代码仓库到本地

使用 SSH 协议在 Git Bash 克隆代码仓库到本地

步骤1 登录**CodeArts Repo首页**。

步骤2 进入要克隆的代码仓库主页，单击“克隆/下载”按钮，并复制SSH地址。

步骤3 在本地Git Bash客户端，执行命令**cd D:/Repo**，进入您要克隆代码仓的地址。如下命令表示克隆的代码仓将克隆到D盘的Repo文件夹下。

步骤4 执行如下命令，克隆代码仓到该目录下。

```
git clone 代码仓库的SSH地址
```

如果您是第一次克隆仓库，会询问您是否信任远程仓库，输入“yes”即可。

如果出现**下图**，说明克隆仓库成功。如果克隆代码仓库失败，请根据下面的“说明”排查并解决问题。

图 8-1 使用 SSH 密钥克隆代码仓成功示意图

----结束

说明

如果在执行步骤3时，Git Bash报错“git@test.com: Permission denied.fatal: Could not read from remote repository.Please make sure you have the correct access rights and the repository exists.”，表示您还未配置访问Repo的SSH密钥，请先配置SSH密钥，具体请参考**配置SSH密钥**。

使用 SSH 密钥在 TortoiseGit 克隆代码仓库到本地

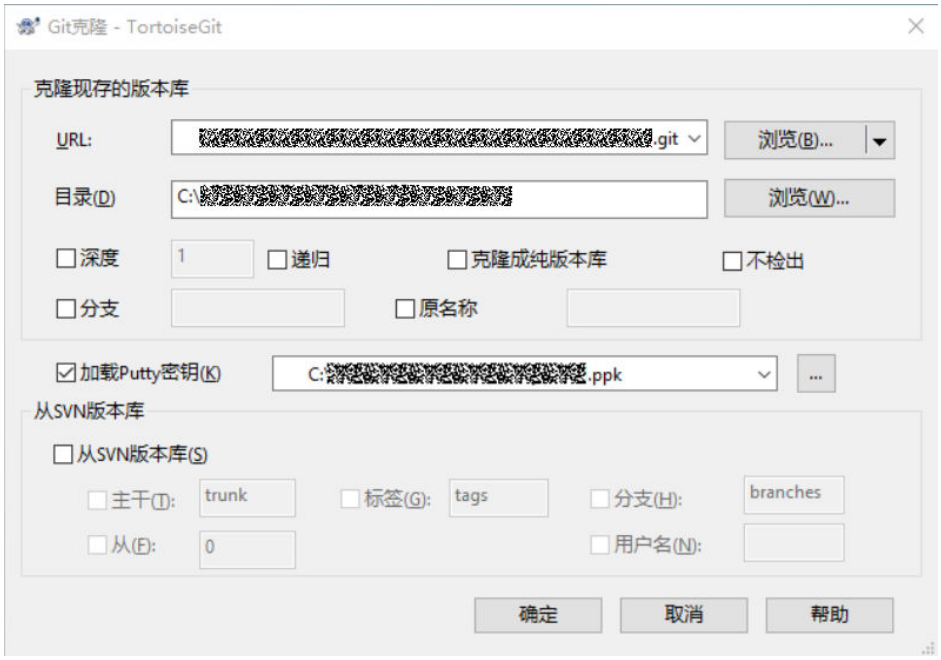
步骤1 登录**CodeArts Repo首页**。

步骤2 进入要克隆的代码仓库主页，单击“克隆/下载”按钮，并复制SSH地址。

步骤3 进入您的本地仓库目录下，右键选择“Git克隆”菜单选项。

步骤4 在弹出的窗口中将步骤1复制的SSH地址粘贴到URL输入框中，勾选“加载Putty密钥”并选择安装TortoiseGit客户端时生成的私钥文件，如**下图**所示。

图 8-2 弹出框界面



步骤5 单击“确定”，如果您是第一次在TortoiseGit客户端克隆代码仓，系统会询问您是否信任远程仓库，单击“是”即可。

----结束

8.3 使用 HTTPS 协议克隆代码仓库到本地

使用 HTTPS 协议在 Git Bash 克隆代码仓库到本地

步骤1 登录**CodeArts Repo首页**。

步骤2 进入要克隆的代码仓库主页，单击“克隆/下载”按钮，并复制HTTPS链接。

步骤3 在本地Git Bash客户端，执行命令**cd D:/Repo**，进入您要克隆代码仓的地址。如下命令表示克隆的代码仓将克隆到D盘的Repo文件夹下。

步骤4 执行如下命令，克隆代码仓到该目录下。

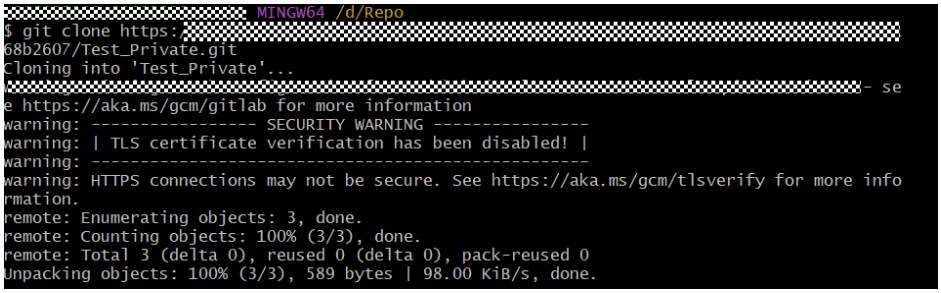
```
git clone 代码仓库的HTTPS链接
```

如果您是第一次克隆代码仓库，您需要填写用户名和密码，有两种类型的用户名和密码，根据您的配置情况，选择如下的一种方式即可：

- 如果需要查看用户名和密码，请登录并进入Repo的代码仓库列表页，单击右上角昵称，选择“个人设置” > “代码托管” > “HTTPS密码”，获取您的用户名和密码，如果忘记密码，可以重新设置HTTPS密码。
- Token用户名和密码。其中，Token的用户名为“private-token”，Token密码为您配置的Token，如果遗失或忘记，可参考**配置访问令牌**重新生成Token。

如果出现**下图**，说明克隆仓库成功。如果克隆代码仓库失败，请根据**说明**去排查解决问题。

图 8-3 使用 HTTPS 协议克隆代码仓成功示意图



----结束

说明

- 在执行**步骤3**时，Git Bash报错“fatal: unable to access 'https:test.com/Test_Private.git/': SSL certificate problem: unable to get local issuer certificate”，请在执行git clone 命令前，先执行如下命令，使Git在使用HTTPS协议克隆代码仓时不进行SSL证书验证：
  ```
  git config --global http.sslVerify false
  ```
- 在执行**步骤3**时，Git Bash报错“fatal: unable to access 'https://test.com/Remote_Test.git/': Failed to connect to test.com port 443 after 21161 ms: Couldn't connect to server”，表示网络不通，请联系您本地所属网络管理员。
- 在执行**步骤3**时，Git Bash报错“fatal: unable to access 'https://xxx.git/'': Recy failure: Connection was reset”，表示域名解析错误，解决办法请参考常见问题。
- 在执行**步骤3**时，Git Bash报错“fatal: destination path 'Test_Private' already exists and is not an empty directory.”，表示Test_Private代码仓已克隆到该路径下且代码仓不为空，解决办法：切换一个新的空目录，重新执行**步骤3**。
- 在执行**步骤3**时，Git Bash报错“fetal: Authentication failed for 'https:/xxx.git/'”，表示您的密码有误，可以登录并进入Repo的代码仓库列表页，单击右上角昵称，选择“个人设置”>“代码托管”>“HTTPS密码”，获取您的用户名和密码，如果忘记密码，可以重新设置HTTPS密码。
- 在CentOS系统下使用HTTPS协议克隆代码时，报错“The requested URL returned error: 401”。这是由于Git版本不匹配。
- 如果您想要通过将访问令牌嵌入HTTPS下载链接，您可以在**步骤3**执行如下命令。其中，password为通过您配置的Token，如果遗失或忘记，可参考**配置访问令牌**重新生成，{project_name}为项目名称，{repository_name}为要克隆的代码仓库名称。
  ```
  git clone https://private-token:password@codehub.test.com/{project_name}/{repository_name}.git
  ```

8.4 使用浏览器下载代码包到本地

步骤1 登录**CodeArts Repo首页**。

步骤2 进入要克隆的代码仓库主页，单击“克隆/下载”按钮。

步骤3 在弹出的窗口中单击需要的代码包类型即可直接下载。

----结束

9 提交代码到 CodeArts Repo 并创建合并请求

9.1 设置代码仓库级的合并请求规则

合并请求配置是指代码合入条件、合入模式的配置，且项目级的合入请求规则可继承到代码仓库、代码组。

您可以勾选“继承项目设置”，自动继承并使用项目下设置且不支持更改。您也可以进入要配置的代码仓库首页，选择“设置”>“策略设置”>“合并请求”。合并请求有两种机制，打分机制和审核机制，两种模式区别如下：

- 打分机制，该模式仅包含代码检视，以打分为基础，只有分数达到门禁条件时，代码才可以合入。
- 审核机制，该模式包含代码检视和合并审核两个步骤，以通过人数为基础，只有检视和审核通过的人数达到门禁条件时，代码才可以合入。

  当您选择合入机制后，参考**下表**填写其余参数，且该配置对整个代码仓库生效。

表 9-1 设置合入条件、合入模式的参数表

参数名称	参数解释
合入条件	此参数非必填，共2个选项： ● 评审问题全部解决才能合入。勾选后，如果评审意见被勾选为“这是一个需要被解决的问题”，则合入条件会提示“存在未解决的评审意见”且“合入”按钮置灰；如果只是一个普通的评审意见，则不存在“已解决”开关，也不会被合入条件拦截。 ● 必须与CodeArts Req关联。包含如下3个子选项： 1. 只能关联一个单号。勾选后，一个MR只能关联一个单号。 2. 所有E2E单号校验必须通过。勾选后，被关联的所有E2E单号校验必须通过。 3. 选择分支配置合并请求策略。可添加多个分支配置合并请求策略，支持手动输入通配符匹配，按回车确认，如：*-stable或production/*。

参数名称	参数解释
MR设置	该参数非必填。包含如下选项： ● 禁止合入自己创建的合并请求。勾选后，您在查看自己创建的MR时，“合入”按钮置灰，自己无法合入，需要找其他有合入权限的人合入。 ● 禁止审核自己创建的合并请求。 ● 禁止检视自己创建的合并请求。 ● 允许仓库管理员及项目经理强制合入。 ● 允许合并请求合并或关闭后继续做代码检视和评论。 ● 是否将自动合并的MR状态标记为关闭状态。如果A MR中的包含在A MR中所有Commits，那么当A合并后，则B MR会自动合并。默认B MR会标记为合并状态，可以通过该选项控制将B MR标记为关闭状态。 ● 不能重新打开一个已经关闭的合并请求。默认打开，您可以根据自己的需要打开或关闭。 ● 新建合并请求，默认开启合并后删除源分支。 ● 禁止Squash合并（合入MR时禁止Squash合并）。 ● 新建合并请求，默认开启Squash合并。

参数名称	参数解释
合并模式	此参数必填，共3个选项： ● 通过Merge Commit合并。勾选后，每次合并操作都会产生一个merge commit点，只要没有检测到冲突就能够执行合并操作。即不管基线点是不是最新的点，无冲突就可以合并。 – Squash合并不产生Merge节点：勾选后，squash合并不会产生merge节点。 – 使用MR合入者生成Merge Commit ： 勾选后，可用于记录Commit信息。 使用MR创建者生成Merge Commit： 勾选后，可用于记录Commit信息。 ● 通过Merge commit 合并(记录半线性历史)。勾选后，每次合并操作会记录一个merge commit提交，但是与“通过Merge commit合并”不同，必须基于目标分支最新的commit提交点进行提交，否则会提示开发者进行rebase操作。这种合并模式下可以非常确定一点，如果merge request能够正确构建，合并完成后目标分支也能够正确构建。 ● Fast-forward 合并。勾选后，每次合并操作不会记录一个merge commit提交，且必须基于目标分支最新的commit提交点进行提交，否则会提示开发者进行rebase操作。

设置分支策略

如果您在上述选择“合入机制”为“审核机制”，并且想为某个分支配置合并策略，您可以进入要配置的代码仓库首页，选择“设置”>“策略设置”>“合并请求”，单击“新建分支策略”，参考下表填写参数。

单击“新建分支策略”按钮，可以为指定分支或该仓库下的全部分支设置合入策略。

表 9-2 新建分支策略参数说明

参数	说明
分支	该参数必填。下拉框选择您想要设置的分支，支持选择全部分支。

参数	说明
最小检视人数	该参数必填。默认为0，表示无需检视人检视通过，也可通过检视门禁。
最小审核人数	该参数必填。默认为0，表示无需审核人审核通过，也可通过审核门禁。
重置审核门禁	该参数非必填。默认勾选，表示当重新推送代码到MR的源分支时，将MR审核门禁重置。
重置检视门禁	该参数非必填。默认勾选，表示当重新推送代码到MR的源分支时，将MR检视门禁重置。
仅能从以下审核人/检视人中追加审核人/检视人	该参数非必填。勾选后，可配置**“追加审核人”**名单与**“追加检视人”**名单，当您想在**“审核人”**与**“检视人”**的必选名单外追加成员时，只允许从**“追加审核人”**名单与**“追加检视人”**名单中追加成员。
开启流水线门禁	该参数非必填。勾选后，合并前需要满足流水线门禁都通过的条件，将CI融入代码开发流程。
合并人	该参数非必填。可配置**必选合并人名单**，在新建合并请求时，该名单将自动同步至合并请求中。
审核人	该参数非必填。可配置**必选审核人名单**，在新建合并请求时，该名单将自动同步至合并请求中。
检视人	该参数非必填。可配置**必选检视人名单**，在新建合并请求时，该名单将自动同步至合并请求中。

说明

分支策略优先级示例如下：

- 假设在仓库下有A策略与B策略，它们配置的分支相同，则系统默认使用最新创建的分支策略。
- 假设在仓库下有A策略与B策略，A策略配置的分支为a分支与b分支，B策略配置的分支为a分支，在发起目标分支为a分支的合并请求时，系统默认使用B策略。

在审核机制下未设置分支策略，则在发起合并请求时使用默认分支策略，该分支策略支持编辑、查看但不可删除，策略配置如下：

- **分支**：*，默认全部分支且不可修改。
- **最小检视人数**：默认为 0。
- **最小审核人数**：默认为 0。
- **重置审核门禁**：默认勾选。
- **重置检视门禁**：默认勾选。
- **仅能从以下审核/检视人中追加审核人/检视人**：默认不勾选。
- **开启流水线门禁**：默认不勾选。
- **合并人**：默认为空。
- **审核人**：默认为空。
- **检视人**：默认为空。

  必选名单举例：

  - **最小审核人数**为2人，如果**必选审核人名单**为空，**追加审核人名单**2人均审核通过，审核门禁通过。
  - **最小审核人数**为2人，如果**必选审核人名单**非空，则该名单内至少一人审核通过，审核门禁才可通过。

9.2 在 CodeArts Repo 编辑代码并提交合并请求

进入要编辑的代码仓库首页，单击“代码”进入代码首页，基于要合入的代码分支新建一个分支。选择要基于修改的分支，根据您的选择进行编辑代码和新建合并请求：

- 如果要新增某个代码文件，单击“新建”，可以新建代码文件，也可以从本地上传单个代码文件，基于某个分支修改后，在“代码”页面右侧，单击“新建合并请求”。
- 如果要在线修改某个代码文件，在“代码”页面，单击要修改的文件名，进入要修改的文件页面，单击  进入文件的编辑模式，编辑并保存后，单击“新建合并请求”，在“新建合并请求”页面的下方可以看到两条分支的文件差异对比详情、要合并分支的提交记录。

须知

分支名不支持以 - . refs/heads/ refs/remotes/ 开头，不支持空格 [\ < ~ ^ : ? * ! ( ) ' " | 等特殊字符，不支持以. / .lock结尾。

9.3 在 Git Bash 创建分支并开发代码

步骤1 进入本地仓库目录，打开Git Bash。执行如下命令，基于master分支新建一条分支feature1，并切换到feature1分支。

```
git checkout -b feature1
```

步骤2 以下步骤模拟将字符串“hello CR”写入到名为hello_cr.txt的文件中。

```
echo 'hello CR' > hello_cr.txt
```

步骤3 将当前目录下所有修改过的文件添加到Git的暂存区中，准备提交到版本库。

```
git add .
```

步骤4 将当前修改的代码提交到本地代码仓库中，并添加一条提交信息。

```
git commit -m 'hello cr'
```

步骤5 查看最近一次提交的详细信息。

```
git log -1
```

步骤6 执行如下命令，将本地分支feature1推送到您远程仓库的origin分支，并将本地分支与远程分支建立追踪关系。

```
git push --set-upstream origin feature1
```

说明

- 执行步骤6时，如果提示“connect to host ********.com port 22: Connection timed out”，表示您的网络被限制，无法访问代码托管服务，请求助您本地所属网络管理员。
- 检查**IP白名单**。注意，在未配置白名单时，全部IP均会放行，如果配置了则只允许名单内的IP访问。

步骤7 进入要新建合并请求的代码仓库首页，选择“合并请求” > “新建”，选择要发起合并请求的源分支和目标分支。在“新建合并请求”页面的下方可以看到两条分支的文件差异对比详情、要合并分支的提交记录信息。

----结束

须知

- 提交本地代码到CodeArts Repo，需要使用git push命令，git commit只是将本地仓库中的修改保存到本地仓库中，每次commit都会生成一个新的commit记录，记录了修改的内容、作者、时间等信息。commit操作只会将代码更改保存到本地仓库中，并不会将修改同步到远程仓库。
- 向CodeArts Repo推送代码时，提示“You are not allowed to push code to protected branches on this project”。原因是该分支为受保护分支，您没有权限推送代码到这个分支。解决方案：仓库所有者或者项目管理员进入代码仓库详情页，选择“设置 > 策略设置 > 保护分支”，单击  ，解除对该分支的保护。
- 向CodeArts Repo推送代码时，提示“src refspec master does not match any”。原因是您没有使用git add、git commit命令依次将文件从工作区加入暂存区。解决方案：在执行git push命令之前，请先使用git add、git commit将修改后的文件提交至暂存区中，再使用push命令推送至云端代码仓库中。
- 向CodeArts Repo推送代码时，提示“error: failed to push some refs to 'https://codehub”。原因是CodeArts Repo的该仓库与本地仓库代码不一致，所以从本地提交代码的操作被拒绝。解决方案：先使用git pull命令拉取CodeArts Repo远端仓的代码，与本地代码仓库合并，再使用git push命令推送代码到CodeArts Repo。
- 使用git pull命令拉取代码失败，提示“Merge branch "master’ of https://codehub/testMaven Please enter a commit message to explain why this merge is necessary”。原因是CodeArts Repo的代码仓库与您本地仓库内容不一致，拉取代码时会跟本地代码进行合并（merge），弹框是提示是否确认本次merge操作，并提交备注信息。解决方案请参考**使用git pull拉取代码失败，报错"Merge branch 'master' of https://xx.com Please Enter a commit"**。
- 执行步骤4时，如果报错“unable to auto-detect email address”，原因是未设置用户名、邮箱。您可以执行如下命令配置您的个人信息。
  ```
  git config --global user.name {你的名字}
  git config --global user.email {你的邮箱}
  ```
- 执行步骤6时，如果报错“'origion' does not appear to be a git repository...”，原因是远程不存在origion这个仓库名称，具体解决方案请参考.**执行git push 命令时，报错'origion' does not appear to be a git repository...**。
- 执行步骤3时，如果报错“Not a git respository”，原因是您不在当前代码仓目录下，需要使用cd命令进入代码仓库目录下。
- 提交合并请求时，如果报错"failed to push some refs to '....git'"，请参考**解决合并请求冲突**。

9.4 在 Git 客户端使用 git-crypt 传输敏感数据

git-crypt 简介

git-crypt是一款第三方开源软件，可以用于对Git仓库中的文件进行透明化的加密和解密。git-crypt可对指定文件、指定文件类型等进行加密存储。开发者可以将加密文件（如机密信息或敏感数据）与可共享的代码存储在同一个仓库中，并且该仓库可以同普通仓库一样被拉取和推送，只有持对应文件密钥的人才能查看到加密文件的内容，但不会限制参与者对非加密文件读写。

在 Windows 中使用密钥对方式进行加密、解密

步骤1 下载并安装最新的Windows Git客户端，下载最新基于Windows的git-crypt，把下载到的exe文件放到Git安装目录下的“cmd”文件夹中。

步骤2 执行如下命令，在本地生成密钥对。

1. 打开“Git Bash”，并进入本地代码仓库。
2. 执行如下命令，在Git代码仓库中创建“.git-crypt”文件夹，文件夹包含加密文件所需的密钥和配置文件。
   ```
   git-crypt init
   ```
3. 执行如下命令，把密钥文件导出到C:/test目录并命名为KeyFile。
   ```
   git-crypt export-key /c/test/keyfile
   ```
4. 执行完上述步骤，您可以到密钥导出的文件路径进行验证，确认是否已成功生成密钥。持有这个密钥文件的计算机，可以解密对应的加密文件。

步骤3 执行如下命令，为代码仓库配置加密范围。

1. 在仓库的根目录下新建一个名为“.gitattributes”的文件。
2. 打开“.gitattributes”文件，设置加密范围，语法如下。
   ```
   文件名或文件范围 filter=git-crypt diff=git-crypt
   ```
   下面给出四个示例。
   ```
   FT/file01.txt filter=git-crypt diff=git-crypt   #将特定文件加密，这里加密的是FT文件夹下的file01.txt
   *.java filter=git-crypt diff=git-crypt   #将 .java类型文件加密
   G* filter=git-crypt diff=git-crypt       #将 文件名为 G 开头的文件加密
   ForTest/** filter=git-crypt diff=git-crypt #将 ForTest 文件夹下的文件加密
   ```

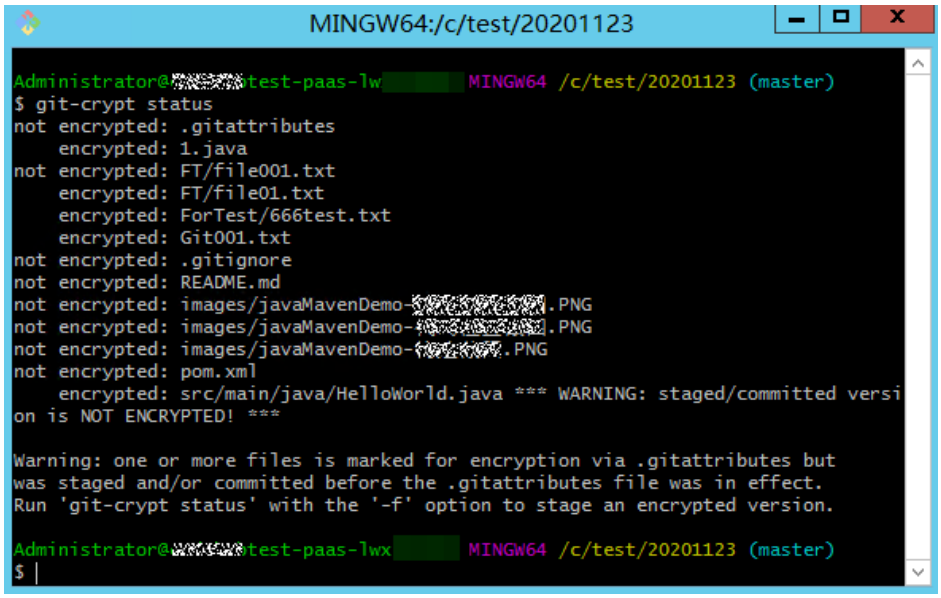
说明

- 如果创建.gitattributes文件时提示“必须键入文件名”，可以将文件名填写成“.gitattributes.”即可创建成功，如果使用Linux指令创建文件，则不会出现此问题。
- 注意不要将.gitattributes保存成txt文件，否则配置会无效。

步骤4 **进行文件加密**。

在仓库根目录打开Git bash，执行如下指令即可完成加密，加密后可看到目前文件的加密状态。

```
git-crypt status
```

加密执行后，在您的本地仓库仍能明文方式打开和编辑这些加密文件，这是因为您本地仓库有密钥存在。

这时你可以使用add 、commit、push组合将仓库推送到代码托管仓库，此时加密文件将一同被推送。

加密文件在代码托管仓库中将以加密二进制方式存储，无法直接查看。如果没有密钥，就算将其下载到本地，也无法解密。

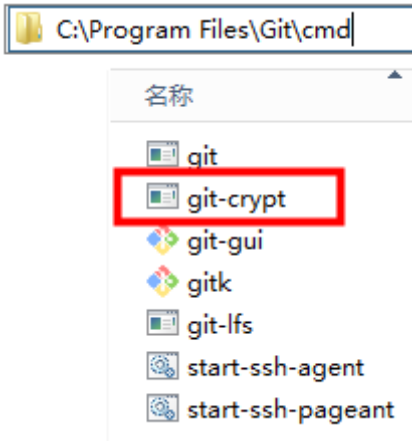
说明

“git-crypt status”只会加密本次待提交的文件，对本次未发生修改的历史文件不会产生加密作用，Git会对此设定涉及的未加密文件做出提示（见上图中的Warning），如果想将仓库中的对应类型文件全部加密，请使用“git-crypt status -f”。

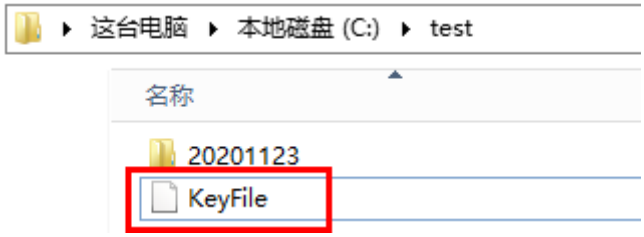
在让团队合作中 -f （强制执行）具有一定的风险，请谨慎使用。

步骤5 进行文件解密。

1. 确认本机器Git安装路径下存在git-crypt文件。



2. 将仓库从代码托管克隆到本地。
3. 获取加密此仓库的密钥文件，并存储于本地计算机。



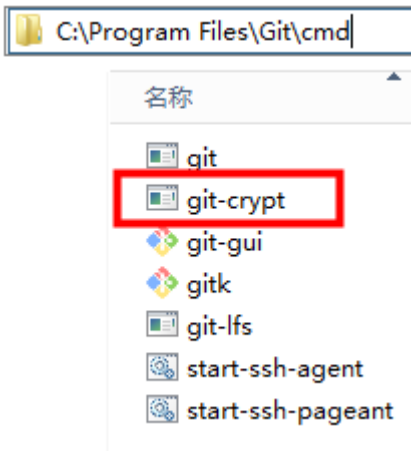
4. 进入仓库目录，右键打开Git bash。
5. 执行解密指令，执行后无回显，则为执行成功。
   ```
   git-crypt unlock /C/test/KeyFile     #请将 /C/test/KeyFile 更换为您实际的密钥存储路径
   ```

----结束

在 Windows 中使用 GPG 方式进行加密、解密

步骤1 安装并初始化Git。

步骤2 下载最新基于**Windows的git-crypt**，将下载到的exe文件放到Git安装目录下的“cmd”文件夹中，下图以“Windows Server 2012 R2 标准版 64”的默认Git Bash安路径为例。



步骤3 **下载GPG**最新版本，当提示您捐赠此开源软件时，选“0”,即可跳过捐赠环节。

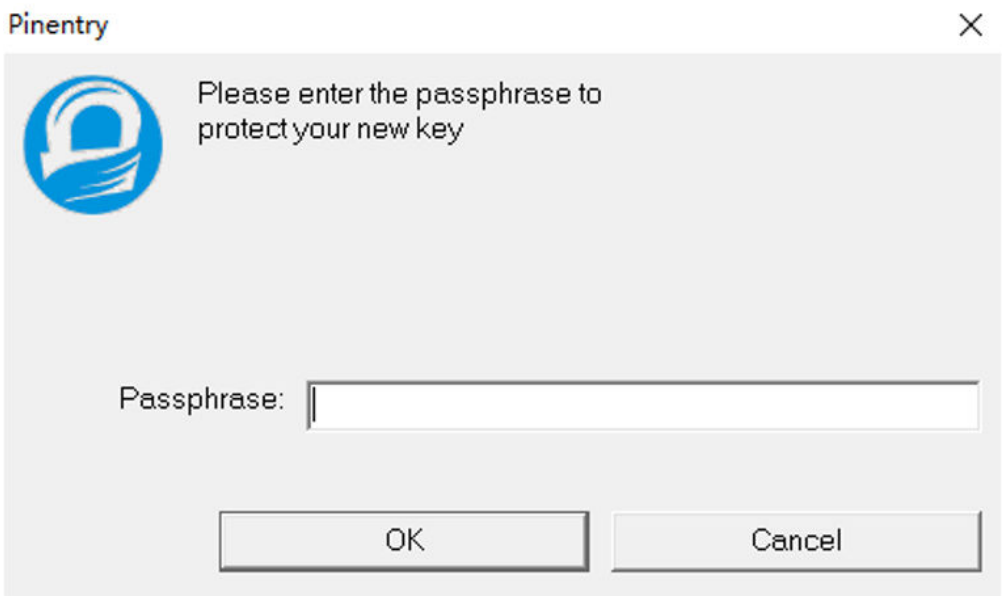
OS	Where	Description
Windows	**Gpg4win**	Full featured Windows version of *GnuPG*
	download sig	Simple installer for the current *GnuPG*
	download sig	Simple installer for *GnuPG 1.4*
OS X	**Mac GPG**	Installer from the gpgtools project
	GnuPG for OS X	Installer for *GnuPG*
Debian	**Debian site**	GnuPG is part of Debian
RPM	**rpmfind**	RPM packages for different OS
Android	**Guardian project**	Provides a GnuPG framework
VMS	**antinode.info**	A port of GnuPG 1.4 to OpenVMS
RISC OS	**home page**	A port of GnuPG to RISC OS

双击进行安装，单击“下一步”，即可完成安装。

步骤4 使用GPG方式生成密钥对。

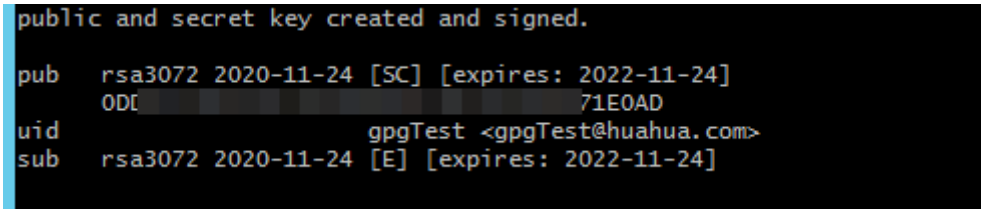
1. 任意位置打开Git Bash，执行如下指令。
   ```
   GPG --gen-key
   ```
2. 根据提示，输入名称、邮箱。

3. 确认无误后，按提示输入 “o”,并回车，此时会弹出输入密码窗口和确认密码窗口。



密码可以为空，出于信息安全考虑，建议设置新的密码。

4. 显示如下图返回内容，则为GPG密钥对生成成功。



步骤5 进行仓库加密初始化设置。

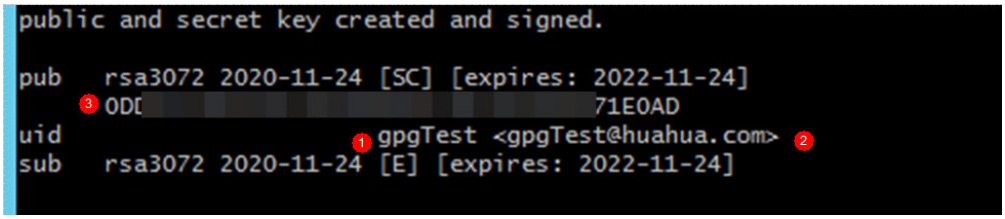
1. 仓库根目录打开Git bash，执行如下指令进行初始化。

```
git-crypt init
```

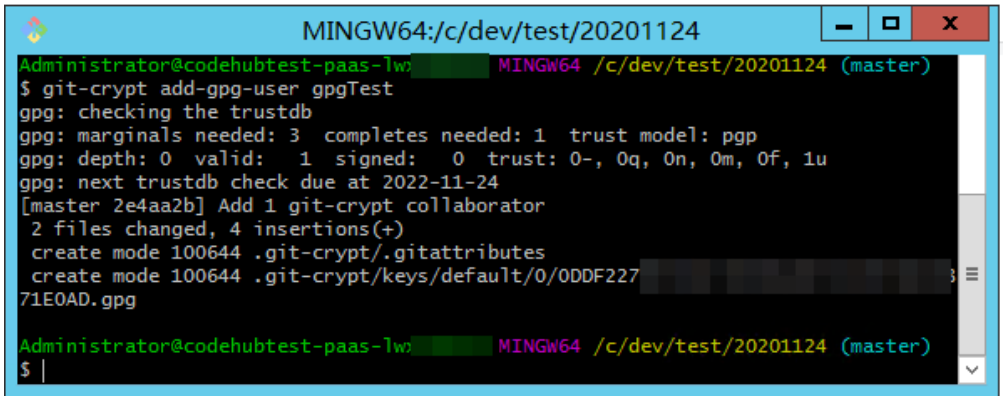
2. 将密钥副本添加到您的仓库，该副本已使用您的公共GPG密钥加密，指令如下。

```
git-crypt add-GPG-user USER_ID
```

此处的“USER_ID”可以是生成此密钥时的名称（①）、邮箱（②）或指纹（③）这三种是可以唯一标识此密钥的凭证。



执行后会提示您创建了.git-crypt文件夹以及其中的两个文件。



步骤6 为仓库配置加密范围。

1. 进入仓库下的.git-crypt文件夹。
2. 打开.gitattributes文件，设置加密范围，语法如下。

   ```
   文件名或文件范围 filter=git-crypt diff=git-crypt
   ```

   下面给出四个示例。

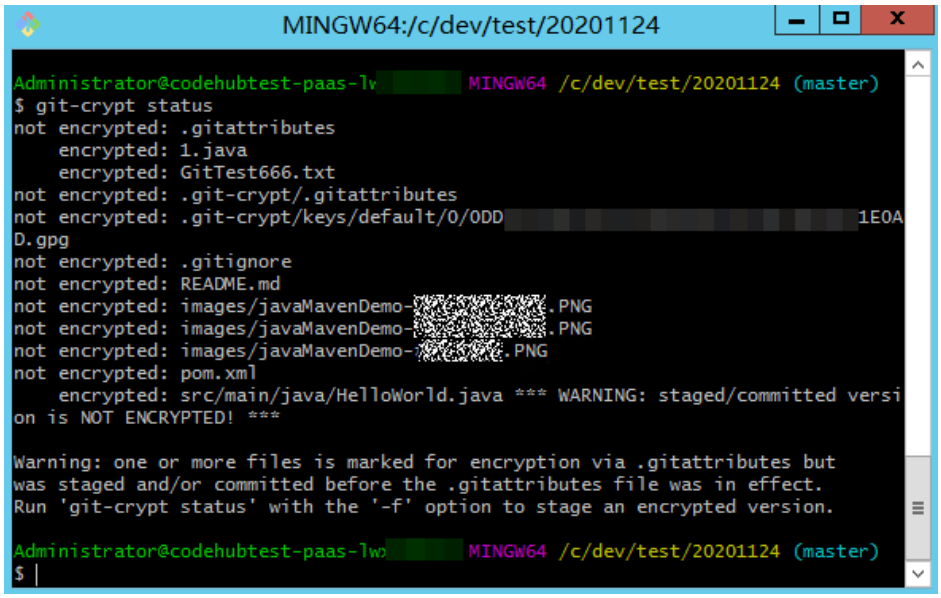
   ```
   FT/file01.txt filter=git-crypt diff=git-crypt   #将 特定文件加密，这里加密的是FT文件夹下的file01.txt
   *.java filter=git-crypt diff=git-crypt   #将 .java类型文件加密
   G* filter=git-crypt diff=git-crypt       #将 文件名为 G 开头的文件加密
   ForTest/** filter=git-crypt diff=git-crypt #将 ForTest 文件夹下的文件加密
   ```

3. 将.gitattributes文件复制到仓库的根目录下。

步骤7 进行文件加密。

仓库根目录打开Git bash，执行如下指令即可完成加密，并会看到目前文件的加密状态。

```
git-crypt status
```

加密执行后，在您的本地仓库仍能明文方式打开和编辑这些加密文件，这是因为您本地仓库有密钥存在。

这时你可以使用add 、commit、push组合将仓库推送到代码托管仓库，此时加密文件将一同被推送。

加密文件在代码托管仓库中将以加密二进制方式存储，无法直接查看。如果没有密钥，就算将其下载到本地，也无法解密。

说明

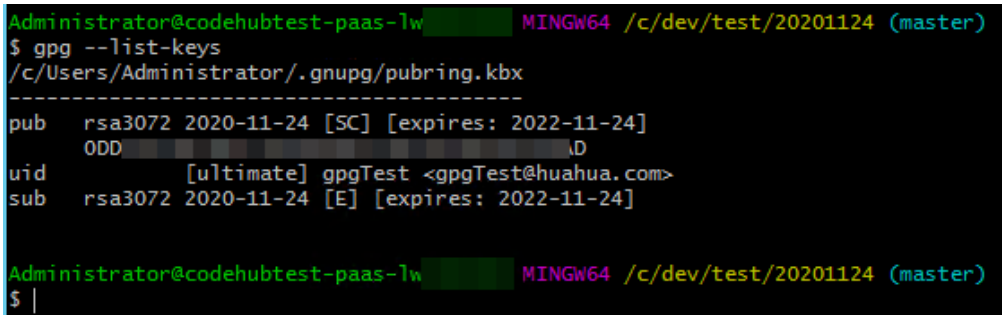
“git-crypt status”只会加密本次待提交的文件，对本次未发生修改的历史文件不会产生加密作用，Git会对此设定涉及的未加密文件做出提示（见上图中的Warning），如果想将仓库中的对应类型文件全部加密，请使用“git-crypt status -f”。

在让团队合作中 -f （强制执行）具有一定的风险，可能会对合作伙伴的工作产生不变，请谨慎使用。

步骤8 将密钥导出。

1. 列出当前可见密钥，可以看到每个密钥的名称、邮箱、指纹信息。

```
GPG --list-keys
```



2. 使用GPG --export-secret-key指令导出密钥，本示例中将名称为“GPGTest”的密钥导出到“C盘”，并命名为“Key”。

```
GPG --export-secret-key -a GPGTest > /c/key
```

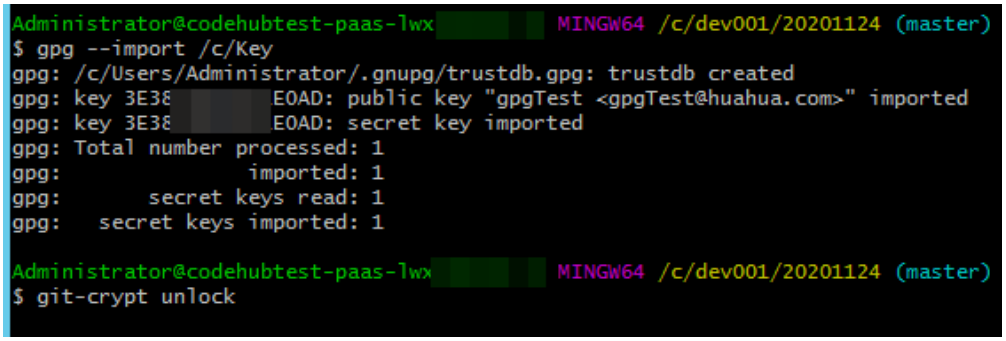
执行时会提示您输入此次密码，正确输入即可。

执行后无回显，到对应目录（本示例中为C盘）可看到密钥文件。

3. 将生成的密钥发送给团队内需要共享秘密文件的伙伴。

步骤9 将密钥导入并解密文件。

1. 想要在另一台机器解密文件，首先也需要基于Git，下载安装git-crypt、GPG。
2. 将对应仓库Clone到本地。
3. 取得对应加密文件的密钥，密钥的导出请参见**步骤8**，本示例中将取得的密钥放在C盘。
4. 进入仓库，打开Git Bash使用import指令导入密钥。导入时会提示您输入密钥的密码。
   ```
   GPG --import  /c/key
   ```
5. 使用unlock指令，解密文件。
   ```
   git-crypt unlock
   ```
   解锁时会弹窗提示您输入此密钥的密码，正确输入后无回显，则为解锁成功。



步骤10 解密完成后，查看文件可以看到文件内容已经不是加密状态。

----结束

git-crypt 加密在团队合作中的应用

很多时候，团队需要在代码仓库中存储**限制公开**的文件，这时可以优先考虑使用“CodeArts Repo” + “Git” + “git-crypt”的组合，来实现部分文件在仓库分布式开源中的加密。

通常，直接使用**密钥对方式的加密**就能满足限制部分文件访问的需要。

当团队需要将加密文件设置不同的秘密级别时，可以使用**GPG方式加密**，这种方式支持您对同一个仓库的不同文件使用不同的密钥加密，将不同密级的密钥分别随仓库共享给组织内的伙伴，即可实现文件的定向分级限制访问。

Linux、Mac 平台的 git-crypt、GPG 安装

Linux平台安装git-crypt、GPG

- Linux安装依赖环境。

Software	Debian/Ubuntu package	RHEL/CentOS package
Make	make	make
A C++11 compiler (e.g. gcc 4.9+)	g++	gcc-c++

Software	Debian/Ubuntu package	RHEL/CentOS package
OpenSSL development files	libssl-dev	openssl-devel

- Linux环境下，使用源码编译方式安装git-crypt。

  下载源码

  ```
  make
  make install
  ```

  安装到指定目录。

  ```
  make install PREFIX=/usr/local
  ```

- Linux环境下，使用源码编译方式安装GPG。

  下载源码

  ```
  ./configure
  make
  make install
  ```

- 使用Debian包安装git-crypt。

  下载源码

  Debian打包可以在项目Git仓库的“debian”分支中找到。

  软件包是用“git-buildpackage”构建的，如下所示。

  ```
  git checkout debian
  git-buildpackage -uc -us
  ```

- Debian环境下使用构建包安装GPG。

  ```
  sudo apt-get install gnupg
  ```

MAC平台安装git-crypt、GPG

- macOS上安装git-crypt。

  使用 brew 软件包管理器，只需运行如下命令。

  ```
  brew install git-crypt
  ```

- macOS上安装GPG。

  使用 brew 软件包管理器，只需运行如下命令。

  ```
  brew install GPG
  ```

9.5 在 Eclipse 提交代码并创建合并请求

如果您本地的Eclipse安装了EGit，可以把本地Git代码仓库代码提交到远程CodeArts Repo，CodeArts Repo当前仅支持Ecplise 4.4及以上版本。

说明

- 如果是首次提交：
  1. 在本地计算机建立一个仓库，称本地仓库。
  2. 在本地进行**Commit**，将更新提交到本地仓库。
  3. 将服务器端的更新**Pull**到本地仓库进行合并，最后将合并好的本地仓库**Push**到服务器端，即进行一次远程提交。
- 如果非首次提交：
  1. 将修改的代码**Commit**更新到本地仓库。
  2. 将服务器端的更新**Pull**到本地仓库进行合并，最后将合并好的本地仓库**Push**到服务器端。

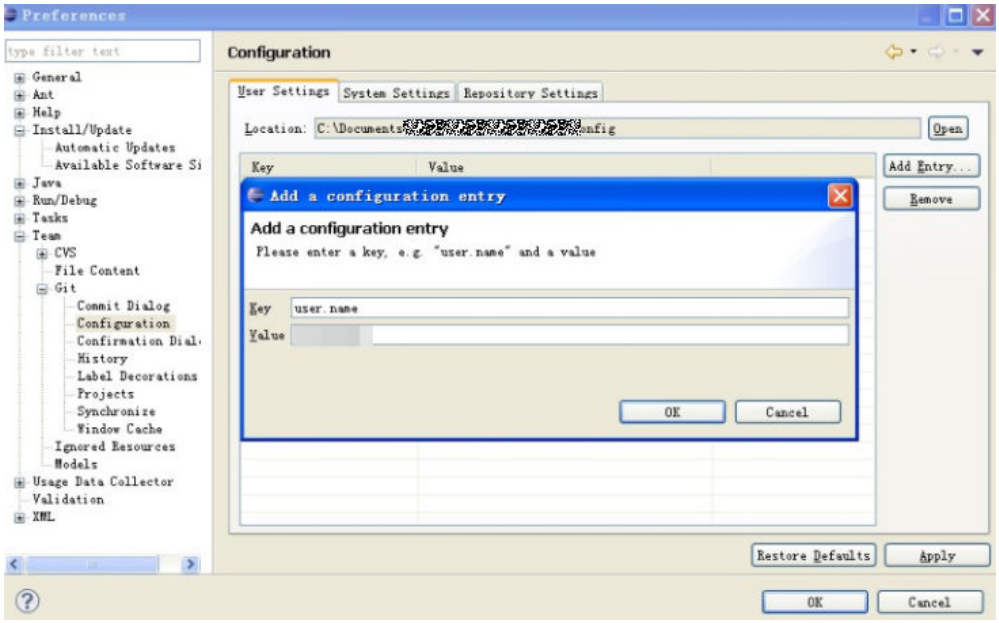
步骤一：在 Eclipse 上安装 EGit 插件

执行如下步骤，安装Eclipse的4.4版本：

在Eclipse上方工具栏选择“Help > Install New Software...”，弹出的“Install”窗口中，单击“Add...”，“Name”栏填写“EGit”，“Location”栏填写**EGit插件地址**，单击“OK”按钮，随后连续单击“Next >”默认安装，安装完成后重启Eclipse。

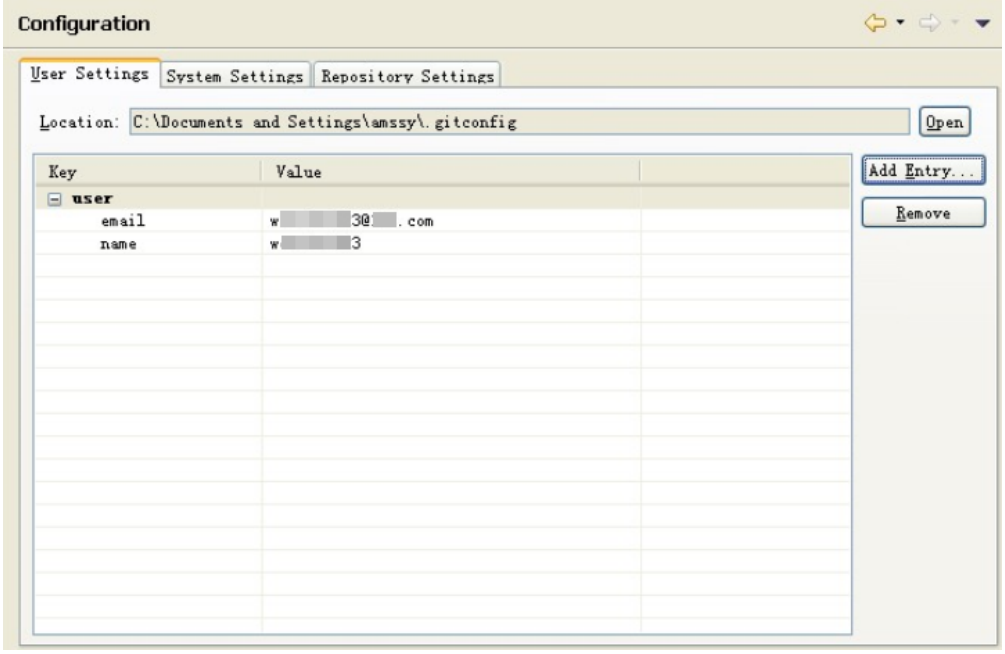
步骤二：在 Eclipse 中配置 EGit

1. 在Eclipse上方工具栏选择“Window > Preferences > Team > Git > Configuration”，填写“User Settings”信息。



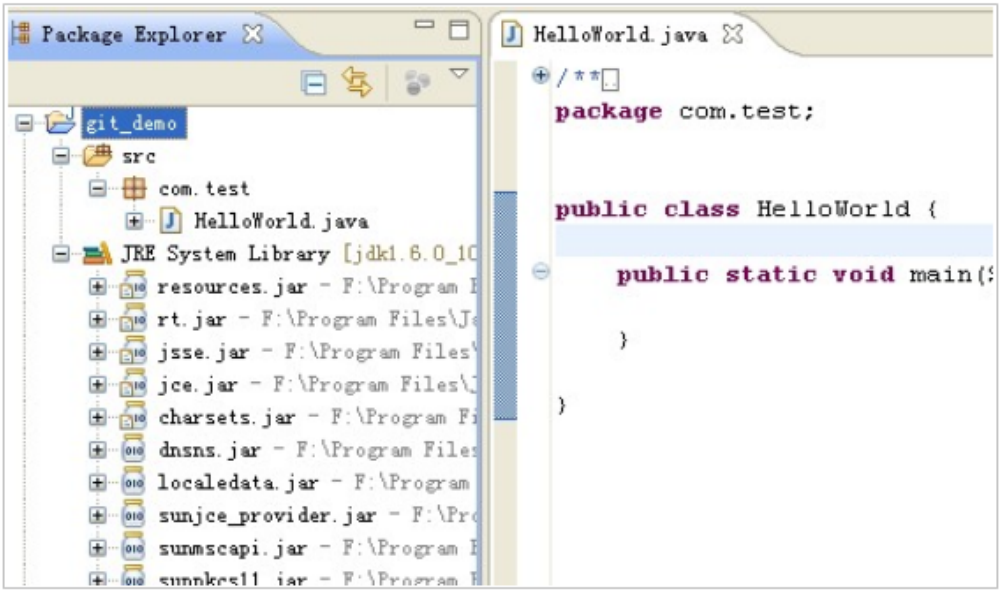
2. 单击“OK”，如下图所示。

   “user.email”为已绑定的邮箱。在这里配置“user.name”即可。

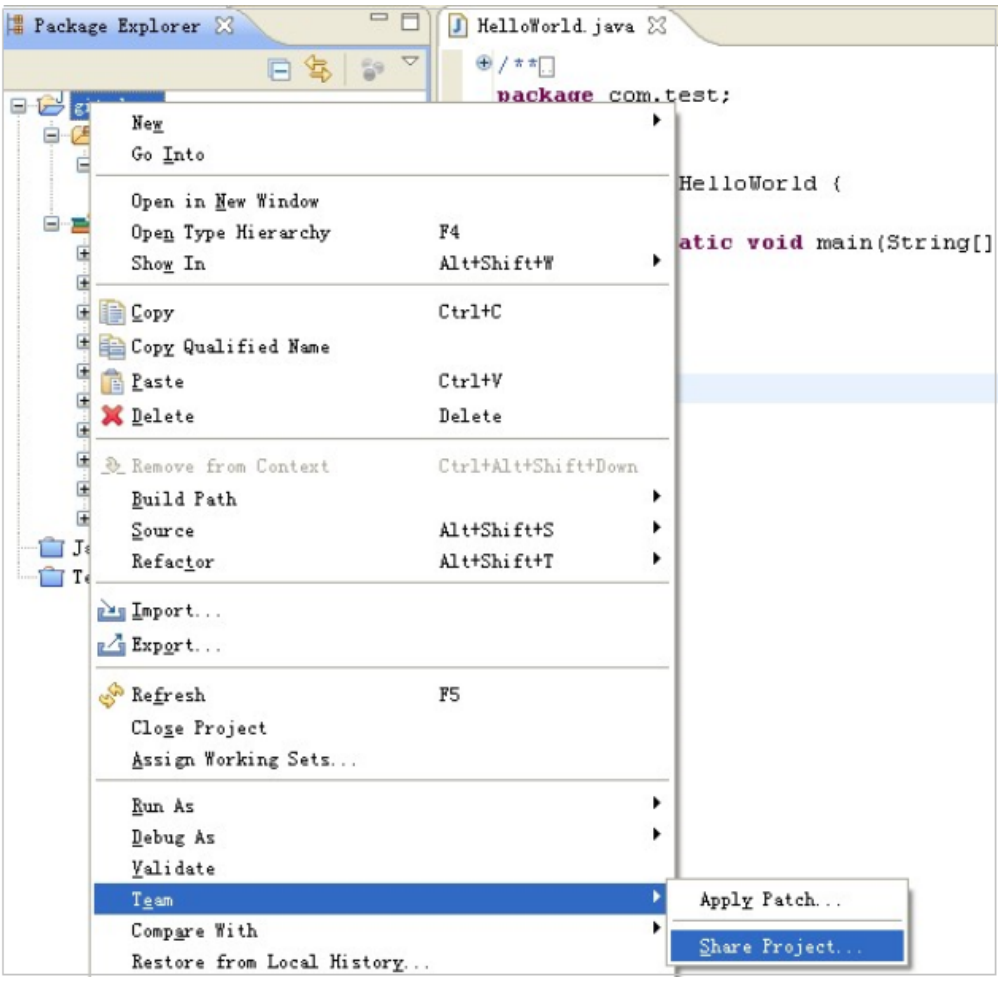


步骤三：新建项目，并将代码提交到本地的 Git 仓库中

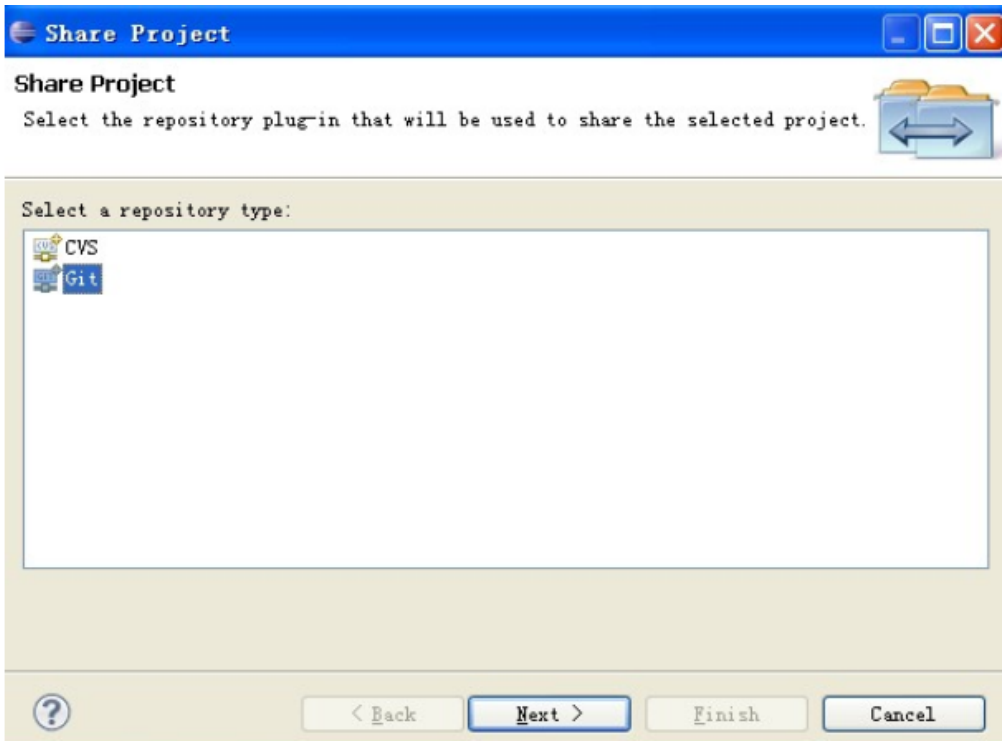
1. 新建项目“git_demo”，并新建“HelloWorld.java”类，如下图所示。



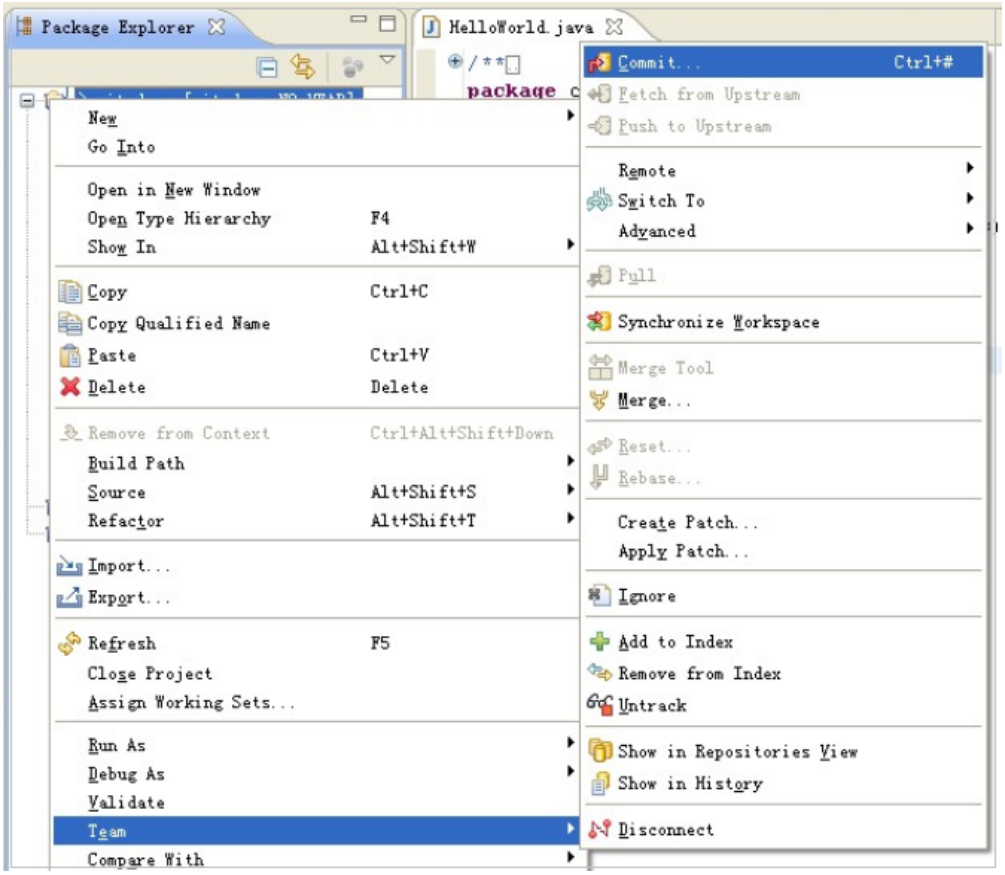
2. 将“git_demo”项目提交到本地仓库，如下图所示。



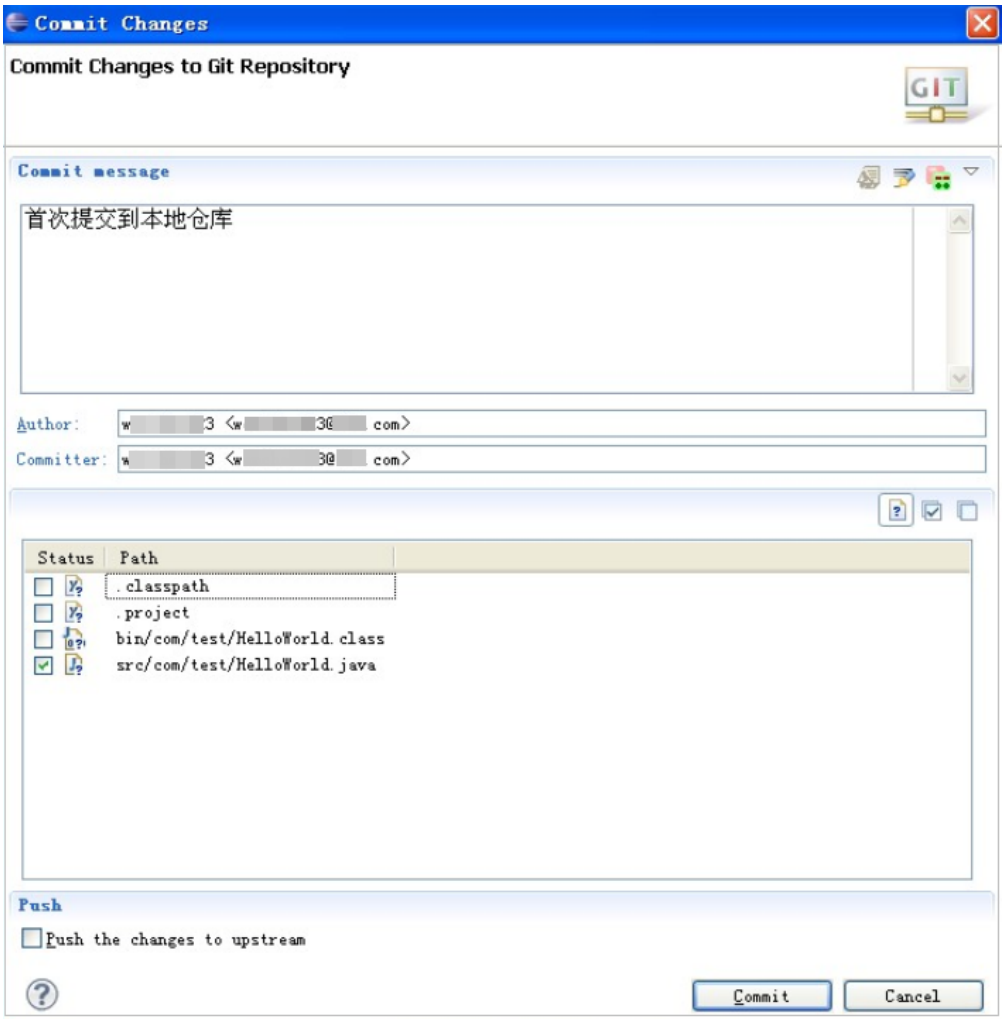
3. 在弹出的“Share Project”窗口中，选中“Git”，如下图所示。



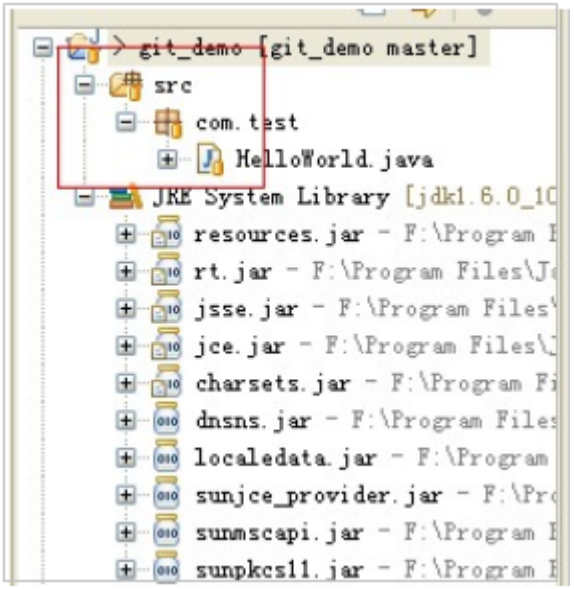
4. 单击“Next >”，弹出“Configure Git Repository”，勾选“Use or create repository in parent folder of project”，单击“Create Repository”。
5. 单击“Create Repository”，成功创建Git仓库。
   文件夹此时处于“untracked”状态（文件夹中以符号“?”表示）。
   此时需要提交代码到本地仓库，如下图所示开始提交。



6. 弹出“Commit Changes”窗口，设置提交信息，如下图所示。

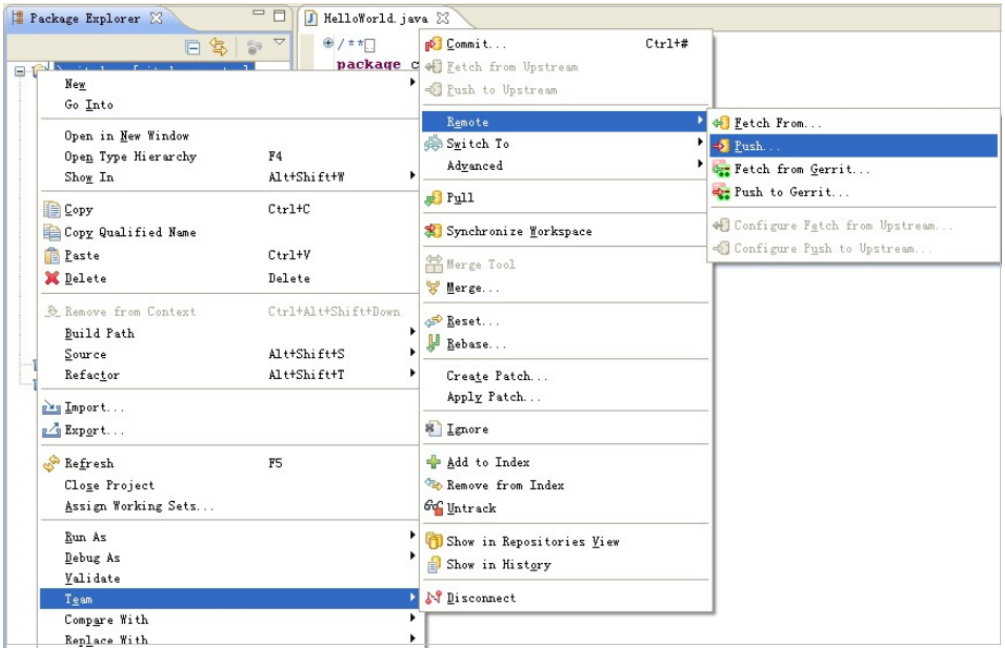


7. 单击“Commit”，代码提交到本地仓库，如下图所示。

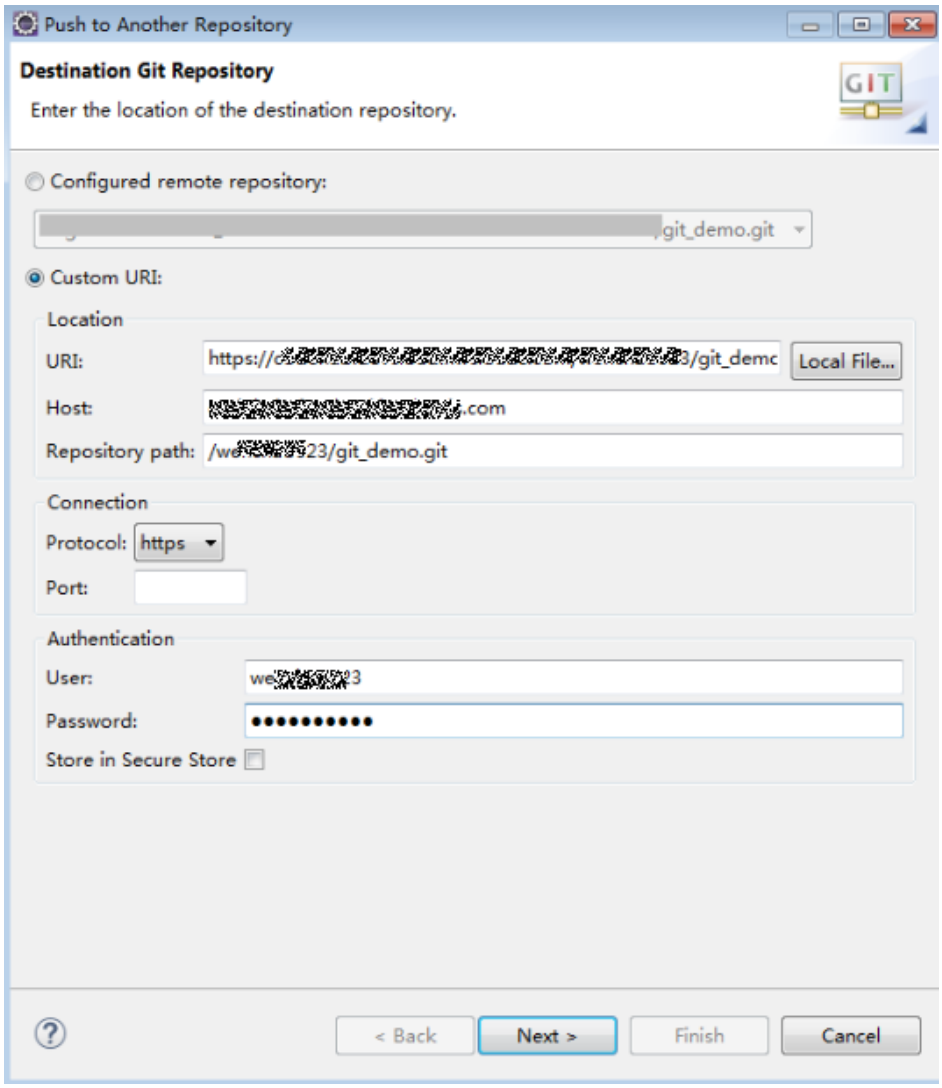


步骤四：将本地仓库代码提交到远程的 Git 仓库中

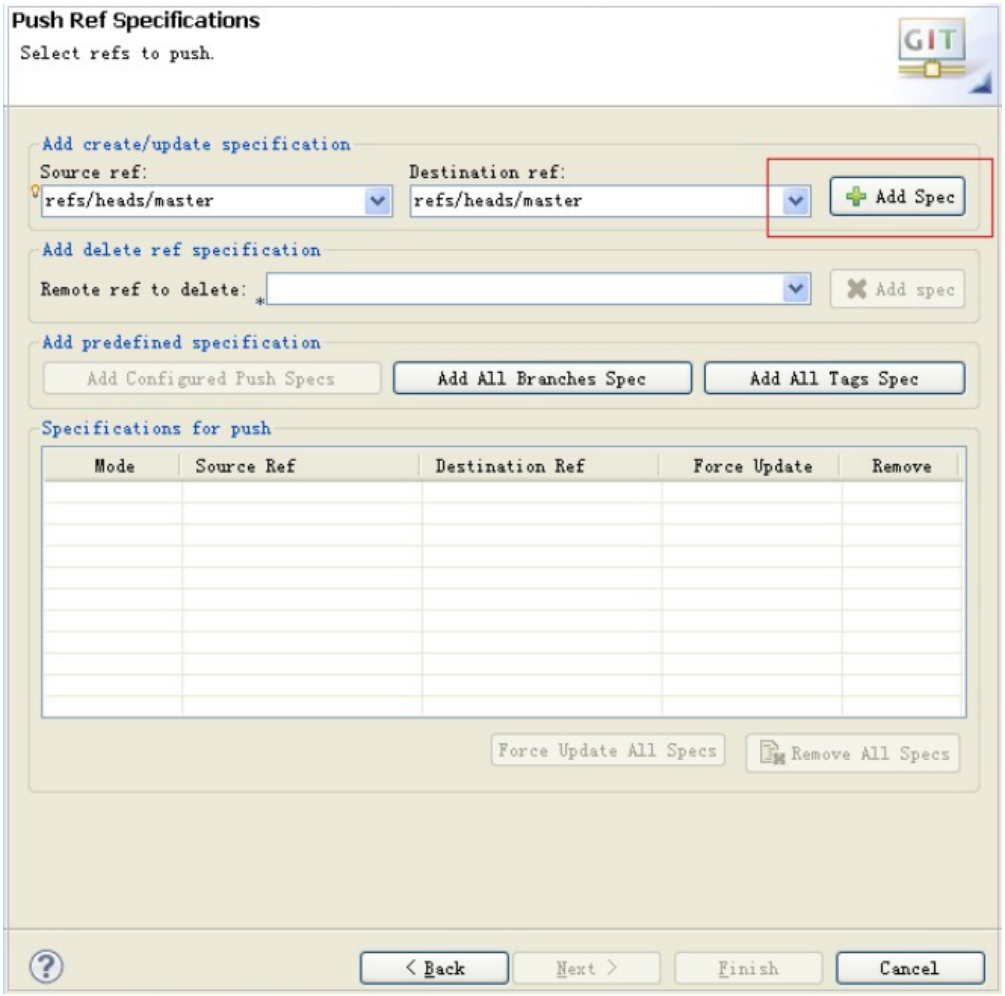
1. 在代码托管服务中**创建仓库**。
   创建好远程仓库后，进入远程代码仓库详情页面，可以复制远程仓库地址。
2. 选择Push菜单，开始将代码提交到远程仓库，如下图所示。



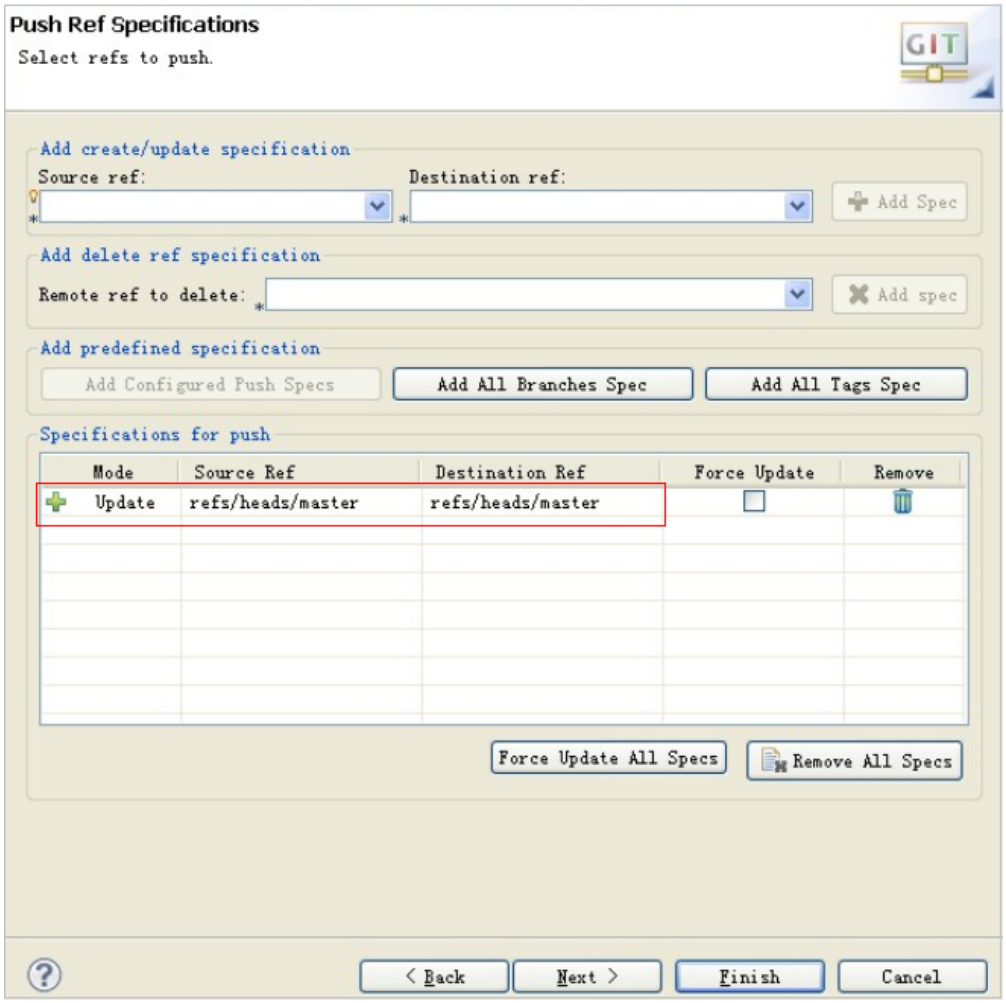
3. 在弹出的“Push to Another Repository”窗口中，设置相应参数，如下图所示。



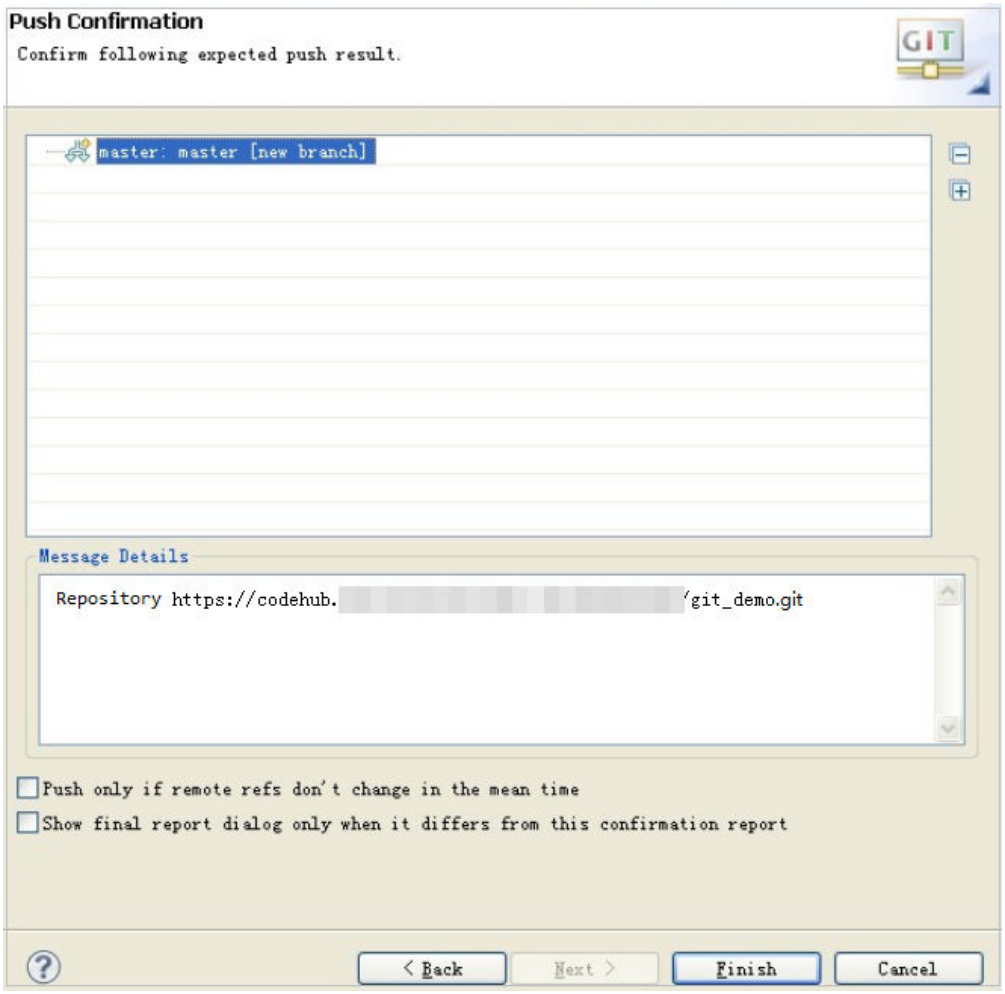
4. 单击“Next”，弹出“Push Ref Specifications”，如下图所示。



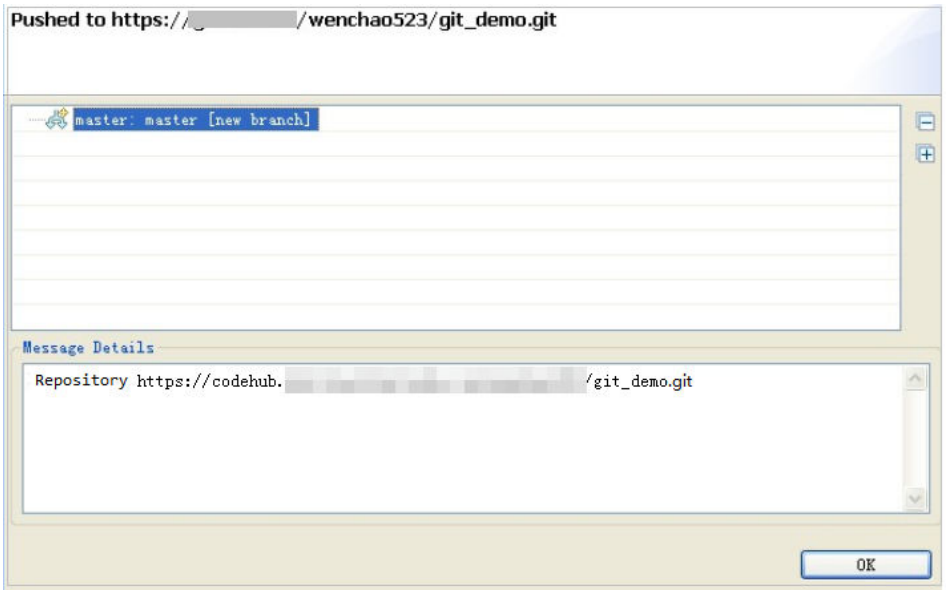
5. 单击“Add Spec”，成功添加，如下图所示。



6. 单击“Next”，弹出“Push Confirmation”窗口，如下图所示。



7. 单击“Finish”提交本地代码，如下图所示。



8. 单击“OK”，完成代码提交远程仓库。
   登录远程仓库地址，核对提交的代码。

说明

在Eclipse使用HTTPS方式连接CodeArts Repo的代码仓库时，提示“Transport Error: cannot get remote repository refs. XXX.git: cannot open git-upload-pack”，这是由于Eclipse中Egit插件的配置问题。解决方案：在Eclipse中，右键选择“Windows > Preferences > Team > Git > Configuration > User Settings”。单击“Add Entry”，“Key”填写内容“http.sslVerify”，“Value”填写内容“false”。

步骤五：在 CodeArts Repo 新建合并请求

进入要新建合并请求的代码仓库首页，选择“合并请求” > “新建”，选择要发起合并请求的源分支和目标分支。在“新建合并请求”页面的下方可以看到两条分支的文件差异对比详情、要合并分支的提交记录信息。

9.6 配置 CodeArts Repo 的合并请求通知设置

CodeArts Repo支持通过邮件或者企业微信的方式推送关于代码仓库和合并请求通知，您可以根据需要打开其中一种通知或者同时打开两种通知，代码仓库内的成员可以查看该页面，仅拥有代码仓库“设置”权限的角色可以配置代码仓库的通知设置。

- 配置邮件通知设置请参考**配置代码仓库的邮件通知设置**。
- 配置代码仓库的企业微信通知设置请参考**表9-4**。

配置代码仓库的邮件通知设置

表 9-3 邮件通知设置的参数说明

参数	说明
仓库	该参数非必填。根据您想收到的邮件通知设置即可，包含四个选项，默认勾选“冻结仓库”和“关闭仓库”，且不可更改，如果仓库出现冻结或者关闭，邮件将通知仓库所有者和项目管理员。另外两个选项如下，并且您可以选择想要邮件通知的对象： ● 如果勾选“删除仓库”，表示有成员删除仓库时，系统将通过邮件通知的方式告知您。 ● 如果勾选“容量预警”，表示超过设置的容量阈值，系统将通过邮件通知的方式告知您，并且您可以下拉选择阈值：60%、80%和90%。

参数	说明
合并请求	该参数非必填，根据您的需要勾选对应选项即可，包含如下选项： • 开启合并请求。表示有合并请求开启时(包括新建和重开合并请求)，会邮件通知到您勾选的角色，默认勾选的角色：评审人、审核人、检视人和合并人。 • 更新合并请求。表示更新合并请求关联分支的代码时，会推送更新邮件，默认勾选的角色：评审人、审核人和检视人。 • 合并合并请求。表示合并请求时，会推送邮件，默认勾选的角色：MR创建人。还可以勾选“合并人”。 • 检视合并请求。表示会推送邮件通知检视合并请求，默认勾选角色：MR创建人。 • 审核合并请求。表示会推送邮件通知审核合并请求，默认勾选角色：MR创建人。 • 新建评审意见。表示会将新建的评审意见推送给选中角色，默认勾选角色：MR创建人。 • 解决评审意见。表示会推送邮件给选中角色，让其解决评审意见，默认勾选角色：MR创建人。

说明

如果在CodeArts Repo已打开邮件通知设置，但仍未收到相关邮件通知，请前往**CodeArts的消息设置**，检查邮箱配置、邮件通知是否开启。

配置代码仓库的企业微信通知设置

表 9-4 企业微信通知设置的参数说明

参数	说明
Webhook地址	该参数必填。用于识别CodeArts Repo成员组所添加机器人的Webhook地址，长度上限为500字符。

参数	说明
仓库	该参数非必填。根据您想收到的微信通知设置即可，包含两个个选项，默认勾选如下两个选项，并且您可以选择想要邮件通知的对象： • 如果勾选“删除仓库”，表示有成员删除仓库时，系统将通过邮件通知的方式告知您。 • 如果勾选“容量预警”，表示超过设置的容量阈值，系统将通过邮件通知的方式告知您，并且您可以下拉选择阈值：60%、80%和90%。
合并请求	该参数非必填，根据您的需要勾选对应选项即可，包含如下选项： • 合并请求状态变更。表示开启、更新或者合并请求状态时，会通过微信机器人的方式推送通知。默认勾选的状态：开启、合并。 • 合并请求检视审核。包括“检视”和“审核”两种状态。 • 合并请求评审意见。默认勾选“新建”状态，还可以根据需要，勾选是否要通知“解决”状态。

9.7 解决评审意见并合入代码

通过评审意见门禁

如果为目标仓库开启了合并请求门禁，即勾选“评审问题全部解决才能合入”。合并请求的检视人或审核人可在合并请求的“文件变更”中，将鼠标置于要提检视意见的代码行，单击图标添加评审意见，也可在合并请求的“详情 > 评审意见”中直接添加评审意见。

当您已解决评审意见后，在合并请求的“详情 > 评审意见”中将评审意见的状态由“未解决”切换成“已解决”，此时门禁将显示为“评审意见门禁已通过”，如图1所示，表示发起合并请求的人将所有评审意见解决，可合入该合并请求。

图 9-1 评审意见门禁已通过示意图

通过流水线门禁

如果为目标仓库开启了流水线门禁，即勾选“开启流水线门禁”。执行如下步骤通过流水线门禁：

步骤1 进入目标仓库首页。选择左侧导航栏“持续交付 > 流水线”，进入流水线服务。

步骤2 单击“新建流水线”，填写以下信息后，单击“下一步”，根据您的需求，选择目标模板。

- 名称：自定义名称。
- 流水线源：选择“Repo”。
- 代码仓：选择需要创建合并请求的目标代码仓。
  默认分支：选择合并请求的目标分支。

步骤3 任务创建成功后会自动跳转任务详情中的“任务编排”页签，切换到“执行计划”页签。

步骤4 开启“合并请求时触发”，根据实际情况勾选以下的一种触发事件。

- 新建：合并请求创建时触发。
- 更新：合并请求内容或设置更新时触发。
- 合并：合并请求合入时触发，该事件会同时触发代码提交事件。
- 重新打开：合并请求重新打开时触发。

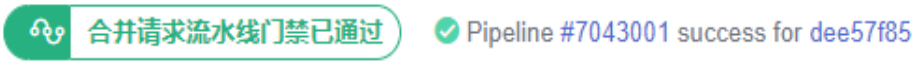
步骤5 完成流水线任务其他信息配置，单击“保存”。

步骤6 返回CodeArts Repo，触发“执行计划”中已勾选的事件让代码仓库执行流水线任务即可。

----结束

进入合并请求详情页，出现图2所示，表示最新commit/预合并commit成功拉起流水线。

图 9-2 合并请求流水线门禁通过示意图



通过 E2E 单号关联门禁

如果为目标仓库开启了E2E单号关联，即勾选“必须与CodeArts Req关联”。执行如下步骤，完成E2E单号关联：

步骤1 进入目标仓库，切换到“合并请求”页签，单击目标合并请求名称，进入目标合并请求。

步骤2 单击“详情”页中“关联工作项”旁的图标，搜索并选择目标工作项。

步骤3 单击**“确定”**，完成E2E单号关联。

----结束

当合并请求成功关联工作项时显示，如下图所示。

图 9-3 E2E 单号关联门禁通过示意图

10 新建并管理代码组

10.1 新建代码组

代码组概述

代码组是由一个或多个仓库组成的群体。您可以为代码组下的仓库或子代码组进行统一的仓库规则配置管理操作，包含提交规则、成员权限配置等。

新建代码组

进入**项目或父组织中**，单击 

图标下拉框选择“**新建代码组**”，进入新建代码组页面，根据下表填写基本信息，单击“**确定**”，完成代码组的新建，代码组最多支持三层目录。

表 10-1 新建代码组参数说明

字段说明	是否必填	备注说明
归属项目	是	• 代码组必须存在项目下。 • 如果账号下没有项目请在项目选择框中选择“新建项目”会先弹出“新建项目”页面，这时建立的项目是Scrum。 **说明** 只有通过“代码托管”首页入口新建代码组时，才能新建项目。
代码组路径	否	代码组路径对应建仓接口的参数groupId。如果groupId为空，则表示创建项目下的仓库，没有对应的代码组。您可根据自己实际需求选择代码组路径。代码组路径范围是所有首层代码组、子代码组的根组织路径。
代码组名称	是	请以大小写字母、数字、下划线开头，可包含大小写字母、数字、中划线、下划线、英文句点，但不能以.git、.atom或.结尾。代码组和仓库总长度限制为256字符。
描述	否	为您的代码组填写描述，限制2000字符。

字段说明	是否必填	备注说明
是否公开	是	可选择私有和公开只读，默认选择私有。 • 私有 仅对代码组成员可见。私有代码组下的子代码组和仓库的公开性只能为“私有”。 • 公开只读 代码组对所有访客公开可读，但不出现在访客的代码组列表及搜索中。公开代码组下的子代码组和仓库的公开性支持私有和公开只读两种。

10.2 使用代码组

10.2.1 查看代码组列表

您可以通过以下方式进入代码托管服务代码组列表页。

进入软件开发生产线首页，单击“**服务**”图标下的“**代码托管**”，默认展示我参与的仓库列表页，单击“代码组”下的任一菜单，即可进入代码托管服务代码组列表页。

在这里您可以完成新建代码组、配置代码组等操作。

-  ：单击该图标，可进入新建代码组页面。
- ：单击该图标，关注代码组。可在我关注的代码组中查看该代码组。
-  ：单击代码组所在行右侧的该图标，可进入子代码组首页。
- ：单击父代码组后的该图标，可展示“仓库”、“成员”、“设置”和“新建子代码组”图标。
  -  ：单击该图标，可直接进入代码仓（组）列表页面。
  -  ：单击该图标，可直接进入代码组成员列表页面。
  -  ：单击该图标，可直接进入代码组“设置”页签下 的代码组信息页面。
  -  ：单击该图标，可直接进入新建子代码组页面。

个人首页：支持查看“**我关注的**”、“**我参与的**”及“**我创建的**”等分类的代码组。右上角支持查看“**最近创建**”和“**最近更新**”的代码组。

10.2.2 查看代码组详情

在代码组列表中单击代码组名称可进入该代码组的详情页面，代码托管服务提供了丰富的控制台操作，详情如下。

表 10-2 页签说明

功能说明	页签说明
代码仓（组）	用于展示代码组的数量、仓库数量、开启中的MR的数量和成员数量等信息。同时您也可以新建仓库和查看未锁定的仓库。
成员	代码组成员管理页面，支持添加成员，调整代码组成员角色。
设置	此代码组的设置入口，代码组所有成员均可查看，但是仅支持项目管理员或代码组所有者修改。

另外代码组详情页框架上还提供以下功能的快捷入口：

获取仓库地址：可通过单击“SSH”或“HTTPS”后的图标，获取仓库地址。

☆：单击该图标，可关注该代码组。

···：在代码组下单击该图标，可查看仓库数目，并进入每个仓库，查看代码组成员并进行代码组成员设置，创建新的子代码组或仓库，按照模板创建新的仓库，以及导入外部仓库。在仓库下单击该图标，可进行关联工作项、成员管理和删除仓库操作。

10.2.3 查看代码组首页

代码组首页用于展示代码组的基础情况。

表 10-3 字段说明

字段	说明
子代码组	统计代码组数量。
仓库	统计仓库数量。
开启中的MR	统计开启中MR数量。
成员	统计代码组中成员数量，单击图标支持跳转至“成员”页签，进行成员管理。
新建仓库	单击图标支持进入“新建仓库”页面，新建仓库。
所有仓库	所有仓库，支持统计锁定仓库和未锁定仓库。

10.2.4 管理代码组合并请求

在代码组详情“合并请求”页签中，可以看到合并请求列表页面。

- 可以切换和查看不同状态的合并请求。
- 通过单击请求标题可以进入合并请求详情页。
- 可以查看请求的简要信息，包括：涉及的分支、创建时间、创建人。
- 提供了多条件维度的查找功能。

说明

- **开启中**：代表该请求已进入检视或合并阶段，分支未合并。
- **已合并**：代表该请求已经完成审核，并完成分支合并的动作。
- **已关闭**：代表该请求被取消，分支未产生实际合并。
- **所有**：显示所有状态的合并请求。

10.2.5 查看代码组评审记录

在代码组详情中的“**评审记录**”页签，可以查看代码组源自于合并请求与Commit的评审信息，可根据选择具体筛选条件进行筛选记录。

表 10-4 评审记录参数说明

参数项	参数说明
状态	评审记录分为“未解决”、“已解决”、“无需解决”三种状态。
评审意见	评审人提出的意见内容。
评审人	提出该评审意见的评审人。
评审日期	评审人提交评审意见的日期。
指派给	指派给系统默认人员或指定人员。

添加“源自合并请求的评审记录”方式

- 在**“合并请求”**页签中“**文件变更**”子菜单下，新建检视意见。

  进入“**文件变更**”子菜单，单击代码行图标，在“文本框”输入评审意见，选择**“严重程度”**和**“指派给”**，如**“严重程度”**为“一般”，**“指派给”**为“MR创建者”，在下拉框选择“**意见分类**”和“**意见模块**”，单击**“确定”**完成检视意见添加。

- 在**“合并请求”**页签中合并请求详情页，新建评论。

  进入“**合并请求**”页签，单击“待检视的合并请求”，进入合并请求详情页。单击“**评审记录**”，在评审记录下输入评审意见，单击“**确定**”完成评论的添加。

添加“源自 Commit 的评审记录”方式

- 方式一：

  在“**代码**”页签下“**文件**”子菜单，新建检视意见。

  进入“**文件**”子菜单，单击“待评审文件”，单击代码行图标，在“文本框”输入评审意见，选择**“严重程度”**和**“指派给”**，如**“严重程度”**为“一般”，**“指派给”**为“MR创建者”，在下拉框选择“**意见分类**”和“**意见模块**”，单击**“确定”**完成检视意见添加。

- 方式二：

  在“**代码**”页签下“**提交**”子菜单，新建检视意见。

  进入“**提交**”子菜单，单击“**提交记录**”下“待检视的文件”，单击代码行图标，在“文本框”输入评审意见，选择**“严重程度”**和**“指派给”**，如**“严重程度”**为“一般”，**“指派给”**为“MR创建者”，在下拉框选择“**意见分类**”和“**意见模块**”，单击**“确定”**完成检视意见添加。

- 方式三：

  在**“提交”**页面中，单击某个提交，切换**“评论”**界面，即可新建评论。

10.2.6 查看代码组动态

在代码组详情中的“动态”页签，可以查看截止当前代码组的全部动态。

- 全部：展示截止当前该代码组的所有操作记录。
- 推送：展示截至当前该代码组所有的推送操作记录，例如推送代码、新建/删除分支等。
- 合并请求：展示截至当前该代码组所有合并请求的操作记录，单击合并请求的序号可查看详情，例如新建/关闭/重开/合入合并请求等。
- 检视意见：展示截至当前该代码组所有检视意见记录，单击提交号可查看详情，例如添加/删除检视意见等。
- 成员：展示截至当前该代码组所有成员的管理记录，例如添加/移除成员、编辑成员权限等。

10.2.7 代码组成员管理

代码托管服务支持在代码组中**添加成员**或**成员组**。

- **“成员列表”**、**“成员组列表”****“待审核”**和**“添加成员”**位于代码组详情的“**成员**”页签下。

- “**成员列表**”用于展示代码组中所有成员“**用户名**”、“**用户来源**”、“**项目角色**”、“**代码组角色**”和“操作”。
- “**成员组列表**”用于展示代码组中所有成员组的“**成员组名称**”、“**成员组数量**”、“**描述**”和“**操作**”。
- “**待审核**”用于展示即将加入代码组中**待审核**成员，包括“**用户名**”、“**项目角色**”、“**代码组角色**”和“**操作**”。“**待审核**”成员可被拥有“**添加成员**”权限的人设置为“**同意**”或“**拒绝**”。
- “**添加成员**”用于代码组**添加成员**或**添加成员组**。

说明

父代码组下的成员无条件继承到子代码组或子仓库，且不允许被删除。

当项目角色发生变化时，如果项目角色和仓库角色一致，则仓库角色同步更新。关于代码组继承成员或成员组添加的成员角色优先级，以最近一次更新为准。

仓库所有者在本仓库中作为管理员角色，享有仓库所有权限，且不可被移除和编辑。

项目管理员为项目下最高权限成员，将同步加入仓库并作为管理员角色，享有仓库所有权限，且不可被移除或编辑。

代码组创建者享有本代码组以及子代码组/仓库的最高权限，且不可被移除和编辑。

该成员通过成员组添加如需删除该成员，请前往所在成员组进行删除操作。

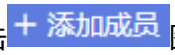
该成员继承于上层代码组，如需删除该成员，请在上层代码组删除即可。

在代码组中添加成员或成员组

步骤1 进入软件开发生产线首页，单击目标项目名称，进入项目。

步骤2 单击菜单“**服务 > 代码托管**”，进入代码托管服务。

步骤3 找到代码组父组织，进入代码组首页。

步骤4 单击菜单“**成员**”，单击 

图标，弹出“**添加成员**”弹框。

步骤5 在“**添加成员**”弹框，单击菜单“**成员**”，您可搜索需添加的成员，选择成员后，单击“**确定**”按钮，添加当前页面的成员进入仓库。

步骤6 在“**添加成员**”弹框，单击菜单“**成员组**”，在下拉框中选择需添加的成员组，单击“**确定**”按钮，添加当前页面的成员组进入仓库。

----结束

10.3 配置代码组

10.3.1 代码组信息

代码组信息可在代码组详情的“**设置 > 基本设置 > 代码组信息**”查看和修改。

此设置只针对被设置的代码组生效。

代码组下所有成员都能查看这个页面，项目管理员和代码组创建者能看到这个页面且有设置权限。

代码组名称默认不可修改。

代码组描述用于描述代码组相关信息。

10.3.2 仓库设置

仓库设置位于代码组详情中的“**设置 > 仓库管理 > 仓库设置**”。

默认分支会作为进入本代码组时，默认选中的分支，也会作为创建合并请求时，默认的目标分支。代码组新建时，master分支将被作为默认分支，可以随时手动调整。

此设置只针对被设置的代码组生效。

仓库内的仓库成员可以查看该页面，仓库成员是否具有仓库设置权限，请参考“权限管理”页面。设置完成后单击**“提交”**即可生效。

表 10-5 参数说明

参数项	说明
MR预合并	默认不勾选，勾选后，服务端会自动生成MR预合并的代码，相比客户端使用命令做预合并操作更高效简洁、构建结果更准确，适用于对构建实时性要求严格的场景。
分支名规则	所有分支名都必须匹配正则表达式。如果此字段为空，则允许任何分支名。需满足基本的分支命名规则，限制500个字符。示例：^feature-[0-9a-zA-Z]+ ● 至多500个字符。 ● 创建分支不支持以“**-**”、“**refs/heads/**”、“**refs/remotes/**”开头，不支持空格[\<~^：？*！（）'‘’|等特殊字符，不支持以“**./**”或“**.lock**”结尾。 ● 新建的分支不可以和原有的分支/tag名重复。
Tag名规则	所有Tag名都必须匹配正则表达式。如果此字段为空，则允许任何Tag名。需满足基本的Tag命名规则，限制500个字符。示例：^TAG*$ ● 不能超过500个字符。 ● 创建tag不支持以“**-**”、“**refs/heads/**”、“**refs/remotes/**”开头，不支持空格[\<~^：？*！（）'‘’|等特殊字符，不支持以“**./**”或“**.lock**”结尾。 ● 新建的tag不可以和原有的分支/tag名重复。

说明

- 字节（byte）：指一小组相邻的二进制数码，是计算机重要的数据单位，通常用大写B表示，1B（byte）= 8bit（位）。
- 字符：表示数据和信息的字母、数字或其他符号。

配置“MR 预合并”

当MR创建后，您可自定义WebHook、流水线等下载插件的脚本，即下载代码内容可以由您自己控制。

- 如果勾选“**MR预合并**”，则服务端会帮助您生成一个隐藏分支，表示该MR代码已经合入，进而您可以直接下载已经存在在隐藏分支的代码。

- 如果未勾选“**MR预合并**”，您需要在客户端本地做预合并，即分别下载MR源分支、MR目标分支的代码，并在构建执行机自己做合并动作。

操作命令

服务端预合并命令如下：

```
git init
git remote add origin ${repo_url克隆/下载地址}
git fetch origin +refs/merge-requests/${repo_MR_iid}/merge:refs/${repo_MR_iid}merge
```

如果未勾选，则可以通过客户端做预合并操作，本地新建干净的工作目录，命令如下：

```
git init
git remote add origin ${repo_url克隆/下载地址}
git fetch origin +refs/heads/${repoTargetBranch}:refs/remotes/origin/${repoTargetBranch}
git checkout ${repoTargetBranch}
git fetch origin +refs/merge-requests/${repo_MR_iid}/head:refs/remotes/origin/${repo_MR_iid}/head
git merge refs/remotes/origin/${repo_MR_iid}/head --no-edit
```

功能优势

对于构建实时性要求高的场景，如：一个MR可能拉起几十或上百台服务器的构建，本地/客户端做预合并可能会与服务端产生的结果不一致，导致构建代码获取不够准确、构建结果不准确等问题。使用服务端预合并可以解决该实时性问题，并且构建脚本命令更简单，开发人员或CIE更好上手。

10.3.3 风险操作

风险操作位于代码组详情中的“**设置 > 风险操作**”。

代码组所有成员均可查看，但是仅支持项目管理员或代码组所有者修改。

目前有如下操作：

- **删除代码组**：删除代码组将导致所有子代码组和资源被删除。删除的代码组无法复原。您只能删除一次，并且无法恢复，请再三确认！
- **更改代码组名称**：将同步修改代码组路径和仓库路径，修改后原路径不可用，请谨慎操作！更改代码组名称可能会引起超出预期的情况。

  说明

  - 更改代码组名称会影响仓库克隆地址，需检查和更新相关配置，否则影响使用，请谨慎操作。
  - 如果代码组下仓库配置了相关流水线，修改代码组名称后，流水线将无法触发，需同步更新流水线相关配置（执行计划和流水线源），具体可参考的“配置流水线”章节。
  - 更改代码组名称后，代码构建、代码检查、部署、CodeArts IDE等服务也需要排查和修改相关配置。

10.3.4 权限管理

代码组的**权限管理**位于代码组详情中“**设置**”页签下。

您可根据下表给各角色配置权限。

说明

代码组权限矩阵仅支持项目管理员及各层代码组的所有者修改。

如果该仓库成员是从代码组下继承的，那么其角色默认为代码组角色，在仓库中修改该仓库成员的角色后，单击“成员列表”页签下仓库成员所在行对应操作列的 按钮时，则该角色权限会改为之前代码组角色。

表 10-6 代码组角色权限

角色/功能	操作权限	项目经理	Committer	开发人员	系统工程师	测试经理、测试人员、参与者、运维经理和产品经理	浏览者	自定义角色
代码组	新建	B	B	B	B	C	D	C
	删除	B	D	D	D	D	D	C
	设置	B	D	D	D	D	D	C
仓库	新建	B	B	B	B	C	D	C
	Fork	B	B	B	B	C	D	C
	删除	B	D	D	D	D	D	C
	设置	B	D	D	D	D	D	C
代码	提交	B	A	A	A	C	D	C
	下载	B	A	A	A	C	D	C
成员	添加	B	D	D	D	D	D	C
	修改	B	D	D	D	D	D	C
	删除	B	D	D	D	D	D	C
分支	新建	B	B	B	B	C	D	C
	删除	B	B	B	B	C	D	C
Tag	新建	B	B	B	B	C	D	C
	删除	B	C	C	C	C	D	C
MR	新建	B	B	B	B	C	D	C
	编辑	B	B	C	C	D	D	C
	评论	B	B	B	B	C	C	C
	检视	B	B	B	B	D	C	C

角色/功能	操作权限	项目经理	Committer	开发人员	系统工程师	测试经理、测试人员、参与者、运维经理和产品经理	浏览者	自定义角色
	审核	B	B	C	C	D	D	C
	合并	B	B	C	C	D	D	C
	关闭	B	B	C	C	D	D	C
	重开	B	B	C	C	D	D	C

说明

- A：表示该角色默认拥有该权限且不可被移除。
- B：表示该角色默认拥有该权限且可被移除。
- C：表示该角色可分配到该权限。
- D：表示该角色不可分配到该权限。

仓库级权限管理位于仓库详情中“设置”页签下。

您可根据下表给各角色配置权限。

表 10-7 仓库级角色权限

角色/功能	操作权限	项目经理	Committer	开发人员	系统工程师	测试经理、测试人员、参与者、运维经理和产品经理	浏览者	自定义角色
代码组	删除	B	D	D	D	D	D	C
	设置	B	D	D	D	D	D	C
仓库	Fork	B	B	B	B	C	D	C
	删除	B	D	D	D	D	D	C
	设置	B	D	D	D	D	D	C
代码	提交	B	A	A	A	C	D	C
	下载	B	A	A	A	C	D	C
成员	添加	B	D	D	D	D	D	C

角色/功能	操作权限	项目经理	Committer	开发人员	系统工程师	测试经理、测试人员、参与者、运维经理和产品经理	浏览者	自定义角色
	修改	B	D	D	D	D	D	C
	删除	B	D	D	D	D	D	C
分支	新建	B	B	B	B	C	D	C
	删除	B	B	B	B	C	D	C
Tag	新建	B	B	B	B	C	D	C
	删除	B	C	C	C	C	D	C
MR	新建	B	B	B	B	C	D	C
	编辑	B	B	C	C	D	D	C
	评论	B	B	B	B	C	C	C
	检视	B	B	B	B	D	C	C
	审核	B	B	C	C	D	D	C
	合并	B	B	C	C	D	D	C
	关闭	B	B	C	C	D	D	C
	重开	B	B	C	C	D	D	C

说明

- A：表示该角色默认拥有该权限且不可被移除。
- B：表示该角色默认拥有该权限且可被移除。
- C：表示该角色可分配到该权限。
- D：表示该角色不可分配到该权限。

11 CodeArts Repo 的安全管理

CodeArts Repo为保证代码仓库的安全性，支持添加IP白名单、支持更改代码仓库所有者、删除代码仓库、更改代码仓库名称、增加水印设置、锁定仓库、记录审计日志，具体可参考如下章节。且这些操作只有具有代码组或者代码仓库“设置”权限的人员可执行，代码仓库的“设置”权限可参考**配置代码仓库级的权限**。

为代码仓库配置部署密钥

为保证代码仓的安全性，有些代码仓库只支持克隆/下载，不支持合入代码等其它变更代码仓库操作，代码仓库处于只读模式，此时需要为该代码仓库配置部署密钥。配置部署密钥位于代码仓库详情中的“设置 > 安全管理 > 部署密钥”，进入部署密钥页面，单击“添加部署密钥”，本地生成SSH密钥可参考**配置SSH密钥**的步骤1~步骤3。

说明

- 多个仓库之间可以使用同一个部署密钥，一个仓库最多可以添加10个不同的部署密钥。
- SSH密钥与仓库部署密钥有区别：前者与用户/计算机关联，后者与代码仓库关联；SSH密钥对仓库有读写权限，部署密钥对仓库是只读权限。
- 此设置只针对被设置的仓代码库生效。

CodeArts Repo 的风险操作

CodeArts Repo支持更改代码仓库所有者、删除代码仓库和更改代码仓库名称，但该操作存在风险，请谨慎操作。

风险操作位于代码仓库详情中的 “设置 > 安全管理 > 风险操作”。支持如下三个操作：

- 移交仓库所有者：可以将当前代码仓库移交给仓库内的其他人（不能移交给浏览者）。
- 删除仓库：一旦您删除该代码仓，代码仓库内所有内容都将会被永久删除。这是一个不可恢复的操作，请谨慎操作。
- 更改仓库名：更改仓库名称将导致仓库的访问和克隆地址改变，在此之前的地址将失效，请谨慎操作。

为 CodeArts Repo 的代码仓库增加水印设置

CodeArts Repo支持为代码仓库增加水印，以此保护代码仓库的知识产权。

水印设置位于代码仓库详情中的 “设置 > 安全管理 > 水印设置” 。

打开水印设置按钮，该代码仓库将展示如下的水印内容：账户+时间。

锁定 CodeArts Repo 的代码仓库

CodeArts Repo支持锁定代码仓库，以此防止任何人破坏即将发布版本的代码仓库。

锁定仓库设置位于代码仓库详情中的 “设置 > 安全管理 > 锁定仓库” 。拥有修改“设置”权限的仓库成员可以执行此操作。

打开水印设置按钮，表示锁定该代码仓库，锁定后将完全只读，任何人无法向任何分支提交代码，也不能创建评论和其他相关新增的操作。

为 CodeArts Repo 代码仓库设置 IP 白名单

CodeArts支持通过设置IP白名单的IP范围和访问权限，限制用户的访问和上传下载权限，增强代码仓库的安全性。配置IP白名单仅支持可见范围为“私有”的仓库，可见范围为“公开只读”或“公开示例模板”的仓库均不能操作此设置。

IP白名单设置位于代码仓库详情中的 “设置 > 安全管理 > IP白名单” 。IP白名单支持IPv4和IPv6，有3种格式，如下表所示。

单击“新建IP白名单”，参考下表填写参数，如果您需要修改，单击IP白名单所在行的  即可。

表 11-1 新建 IP 白名单参数说明

参数	说明
IPv4	如果选择该参数，您指定IP、设置IP范围或者设置CIDR格式的路由，区别如下： • 指定IP，表示该IP将被添加到白名单中，例如将您的个人家庭电脑的IP添加到白名单中。 • 指定IP范围，当您拥有不止一台服务器而且IP段是连续的，或者您的IP会在一个网段内动态变化，您可以添加一个IP白名单范围。示例：100.*.*.0 - 100.*.*.255。 • 设置CIDR格式的路由，当您的服务器在一个局域网内并使用CIDR路由时，您可以指定局域网的32位出口IP以及一个指定网络前缀的位数。从同一个IP发起的请求，只要网络前缀同您设定的前缀部分相同，即可视为来自同一授信范围从而被接受。
IPv6	如果选择该参数，您指定IP、设置IP范围，可参考**指定IP**和**设置IP范围**。
备注	非必填参数。

参数	说明
访问控制	非必填参数，根据您的需要，勾选对应选项即可： ● 允许访问仓库：勾选该选项后，白名单内的IP才可以访问该仓库，仓库所有者不受限制。 ● 允许下载代码：勾选该选项后，白名单内的IP允许在线下载、本地克隆代码。 ● 允许提交代码：勾选该选项后，白名单内的IP才可以在线修改、在线上传和本地提交代码；构建工程代码化编排，yaml文件同步功能不受访问控制。

说明

如果您需要对租户下的所有代码仓库统一设置IP白名单，登录您的代码托管服务仓库列表页，单击右上角昵称，单击“租户设置 > 代码托管 > 租户级IP白名单”，进入页面，，其配置规则与如上配置相同。

CodeArts Repo 记录审计日志

CodeArts Repo支持更改代码仓库属性，因此CodeArts Repo会将关于该代码仓的代码提交、合并请求等信息进行记录，每一条审计日志包含操作者、操作类型和操作内，您可以根据时间段进行筛选查看。

