天翼云

代码托管

快速入门

发布日期 2023-06-30

目　录

1 新手指引

如果您是初次接触 Git，可以前往基于 Git 的代码托管入门了解 Git 与代码托管服务的工作原理。

如果您有基于 Git 进行版本管理的经验，下面将快速帮您了解代码托管服务的功能。

新建一个仓库

下面以**创建模板仓库**为例。

步骤 1 进入目标项目下的代码托管服务。

步骤 2 单击“普通新建”旁的图标，在扩展框中选择“按模板新建”，跳转到**“选择模板”**页面。

步骤 3 **“选择模板”**页面支持模糊查询，根据您的需求选中某个模板。

步骤 4 单击“下一步”按钮，进入**“基本信息”**页面，填写仓库基本信息。

表1-1 按模板新建仓库的参数说明

字段名称	是否必填	备注说明

字段名称	是否必填	备注说明
代码仓库名称	是	以字母、数字、下划线开头，名称还可包含点和连字符，但不能以.git、.atom 或.结尾，限制 200 字符。
归属项目	是	• 仓库必须存在项目下。 说明 如果在项目内新建仓库则默认选择该项目，页面会隐去“归属项目”这个字段。
描述	否	为您的仓库填写描述。
权限设置	否	• **允许项目内人员访问仓库。** 选择后会自动将项目中的项目经理设为仓库管理员，开发人员设为仓库普通成员。当项目新增这两个角色时，会自动将新增成员同步到代码仓成员中，可通过成员列表查看。 • **自动创建代码检查任务（免费）。** 仓库创建完成后在代码检查任务列表中，可看到对应仓库的检查任务。（注意切换到仓库所在区域）
是否公开	是	可选择： • **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，可以选择添加开源许可证作为备注。

步骤 5 单击“确定”按钮，完成仓库的新建并返回仓库列表。

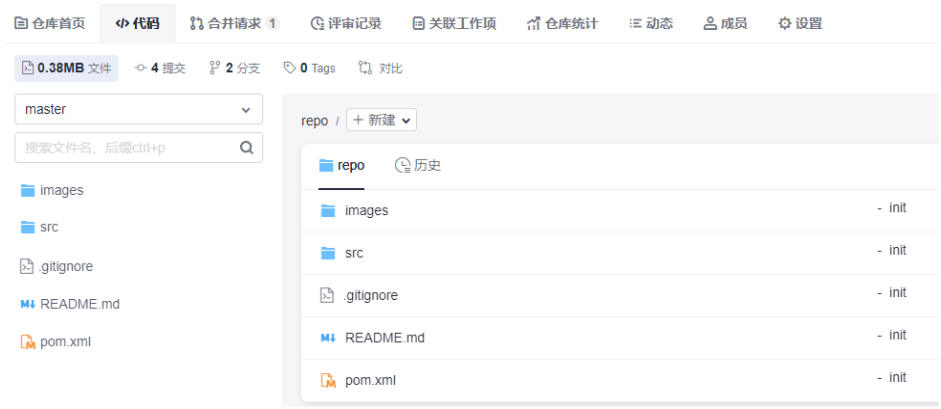
----结束

您已新建了一个仓库，去新建一个分支吧！

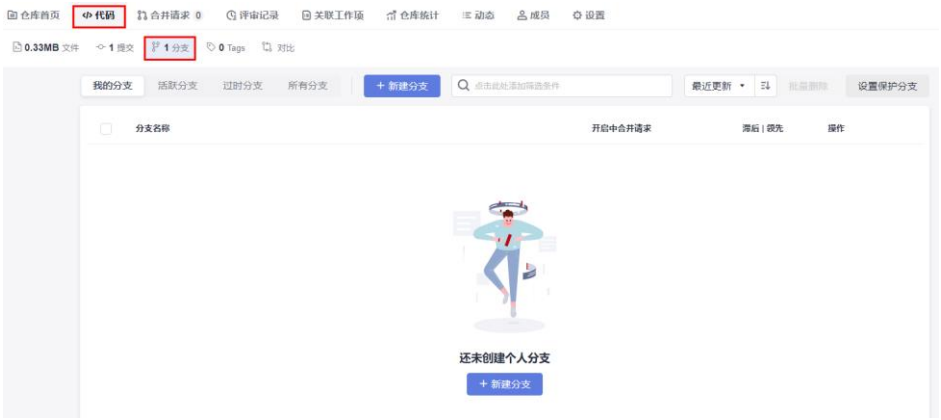
新建一个分支

分支是版本管理工具中最常用的一种管理手段，使用分支可以把项目开发中的几项工作彼此隔离开来使其互不影响，当需要发布版本之前再通过**分支合并**将其进行整合。

步骤 1 单击仓库名称进入仓库详情。



步骤 2 切换到“代码”页签下的**“分支”**子页签，在这里可以看到目前仓库中的分支列表。



步骤 3 单击“新建分支”按钮，在弹出的窗口中选择要基于哪个版本（分支或标签）进行创建，填写新分支的名称，并且可关联现有工作项。

表1-2 参数说明

名称	是否必填	备注说明
基于	是	基于已有某个分支或标签进行新建分支。
分支名称	是	新建分支的名称。
描述	否	针对于新建的分支的描述。
关联工作项	否	支持关联工作项

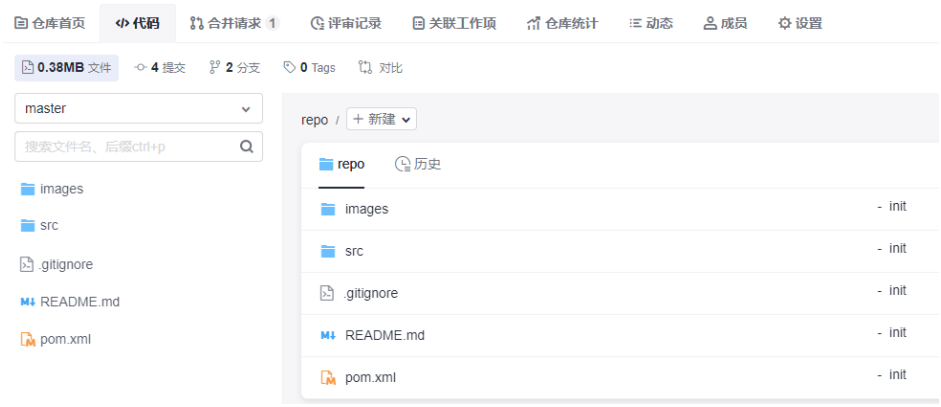
步骤 4 单击“确定”按钮，即可完成分支的新建并返回分支列表。

----结束

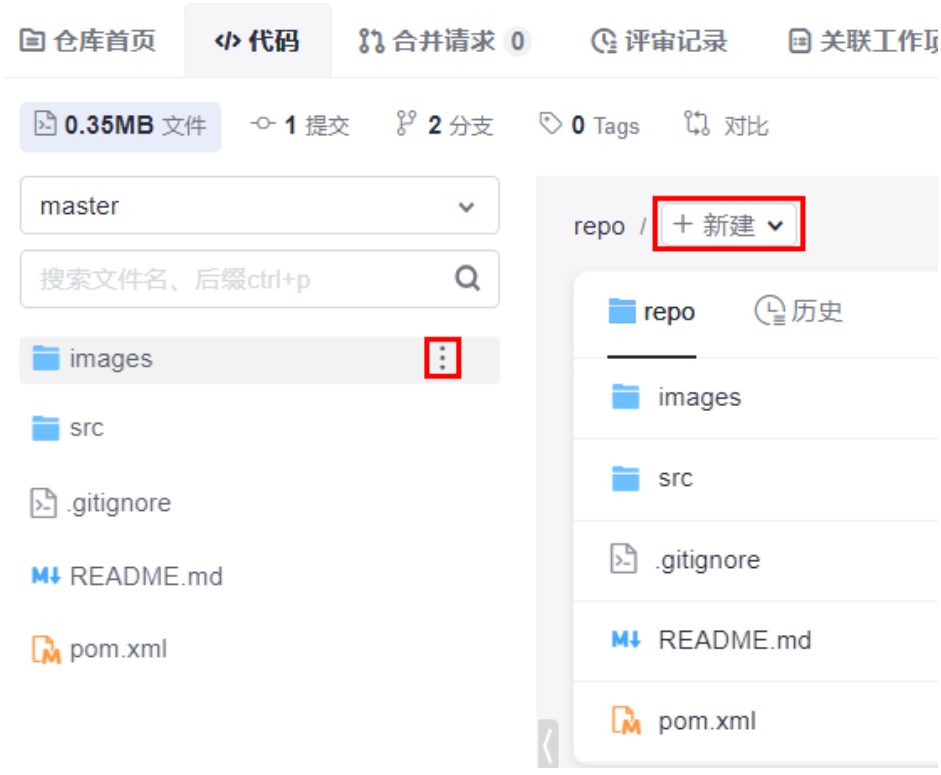
您已新建了一个分支，去新建一个文件吧！

新建一个文件

步骤 1 单击仓库名称进入仓库详情。



步骤 2 鼠标停留在文件夹名称处，单击显示的 ⋮ 图标或单击 + 新建 图标，单击**“新建文件”**。



步骤 3 根据情况填写以下参数。

表1-3 参数说明

名称	是否必填	备注说明
文件名称	是	新建文件的名称。
空白文件（不使用模板）	是	支持选择多种模板类型，默认选择空白文件（不使用模板）。
text/base64	是	支持 text 与 base64 编码类型，默认 text 编码类型。
文件内容	否	新建文件的内容。
提交信息	是	自动同步【新建文件 您的文件名称】，支持自定义内容，可关联工作项。

步骤 4 单击**“确定”**按钮，即可完成文件的新建并返回文件列表。

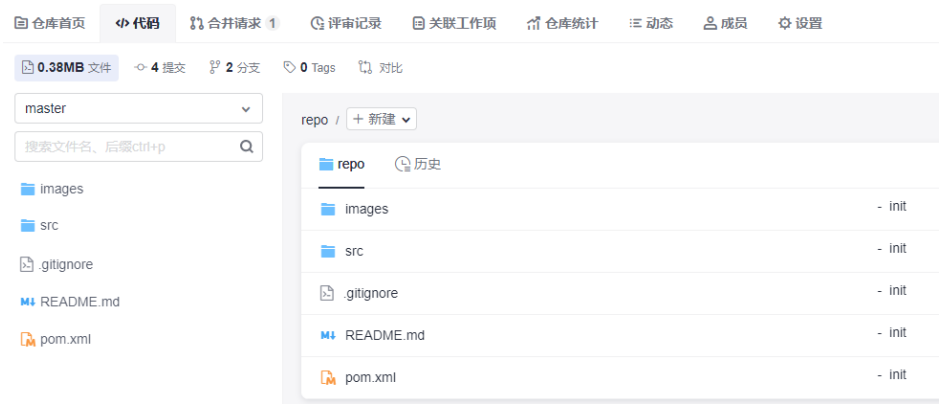
----结束

您已新建了一个文件，接下来您可以对两个分支新建一个合并请求了！

新建一个合并请求

代码托管服务支持多分支开发，并为分支合并建立了可配置的审核规则，当一个开发者发起一次合并请求时，可选择部分仓库成员参与到代码审视中，以确保合并代码的正确性。

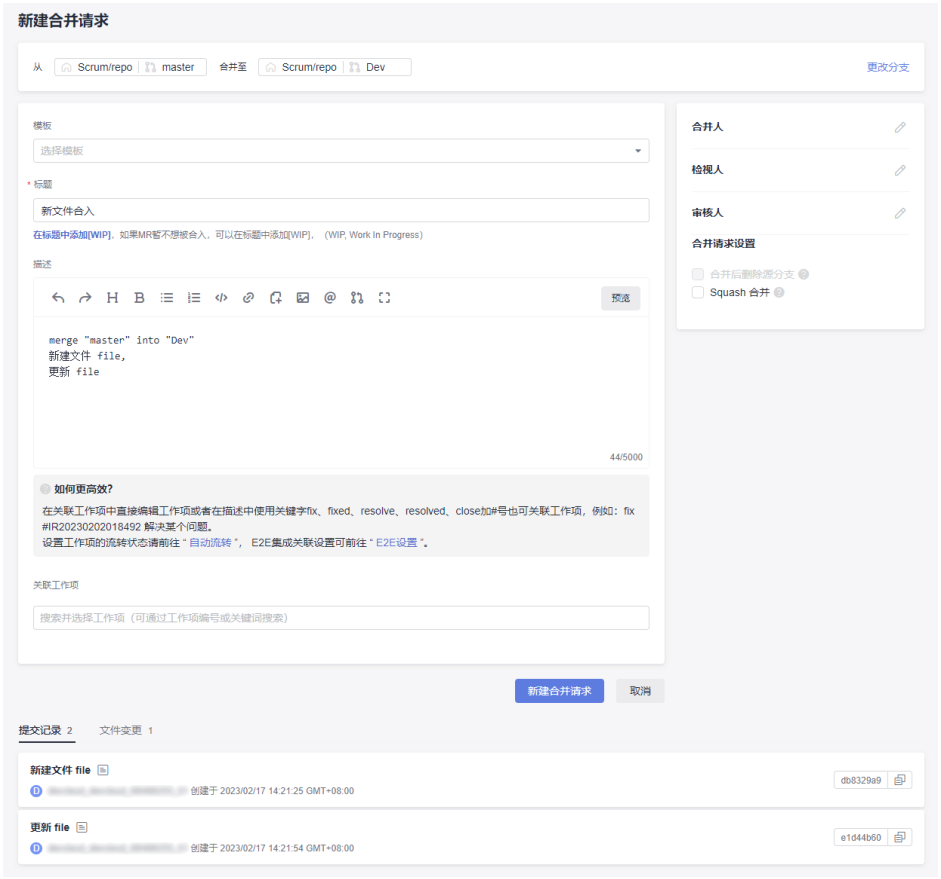
步骤 1 单击仓库名称进入仓库详情。



步骤 2 切换到“合并请求”页签，单击“新建”按钮，选择要合并的分支。

步骤 3 单击“下一步”按钮，此时系统会检测两条分支是否有差异。

- 如果分支没有差异，系统会做出提示，且不能新建合并请求。
- 如果分支存在差异，则进入如下“新建合并请求”页面。



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。

步骤 4 根据下表参数说明，填写页面信息。

表1-4 参数说明

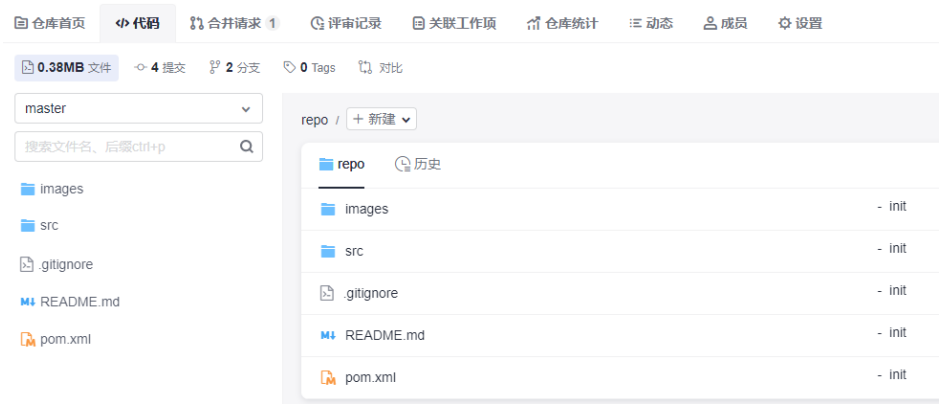
参数	说明
更改分支	单击可返回上一步更改需要合并的分支。
模板	若仓库管理员或所有者已为该仓库创建合并请求模板，您可以直接选择使用模板。 **说明** **该功能仅支持“专业版套餐”及“铂金版套餐”用户。**
标题	输入合并请求的标题。
描述	会结合分支合并情况与要合并分支的提交（commit）备注生成默认值，您可以根据项目情况进行修改。
关联工作项	可选择将合并动作关联到某个工作项，以起到自动改变工作项状态的作用。
合并人	在合并请求满足合入要求时，一般是所有审核人审核通过、所有问题都被解决（可设置不解决也能合并），合并人有权限执行合并操作（单击按钮）、也有权限关闭合并请求。
检视人	被指定参与合并分支检视，可以提出问题给发起人。
评审人	被指定参与合并分支评审，可以给出审核意见（审核通过、拒绝），也可以提出问题给发起人。
合并后删除源分支	可选择是否合并后删除源分支，初始会带入合并请求设置中预设状态。
Squash 合并	Squash 合并是将合并请求的所有变更提交信息合并为一个，并保留干净的历史记录。当用户在处理功能分支只关注当前提交进度，而不关注提交信息时，可使用 squash merge。 开启 Squash 合并，可使基本分支的历史记录保持干净，并带有有意义的提交消息，而且在必要时可以更简单地恢复。

步骤 5 单击“新建合并请求”按钮，可以完成合并请求的提交，页面会跳转到该“合并请求详情页”。

----结束

合入一个合并请求

步骤 1 单击仓库名称进入仓库详情。



步骤 2　切换到“合并请求”页签，单击目标合并请求名称，进入合并请求详情页。

步骤 3　检视人、审核人对合并请求进行检视、审核操作。

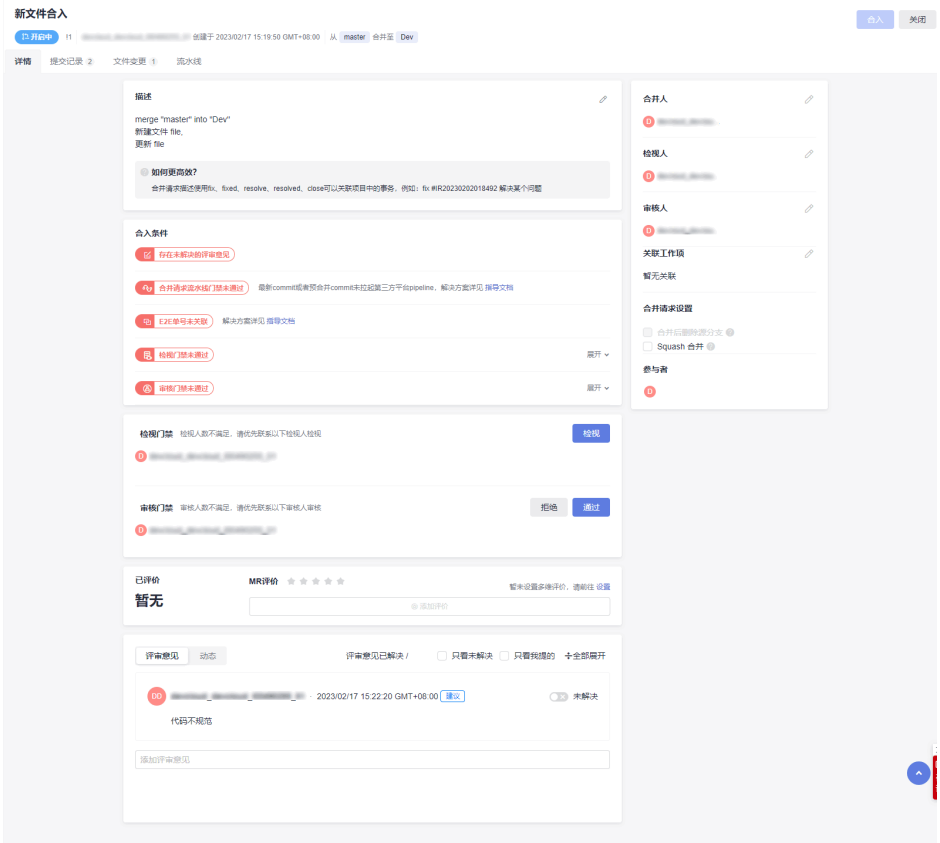


表1-5 合入条件说明

合入条件	说明

合入条件	说明
代码合并冲突	当源分支代码与目标分支代码产生合并冲突时，需要先解决冲突才可进行下一步操作，解决代码冲突可参考《用户指南》中“使用代码托管仓库 > 管理合并请求 > 解决合并请求的代码冲突”。
评审意见门禁	当发起人解决所有检视人或审核人的评审意见后，门禁显示通过。 说明 需要在**“设置 > 策略设置 > 合并请求”**中勾选**“评审问题全部解决才能合入”**后，门禁功能生效。
流水线门禁	当最新 commit 或者预合并 commit 拉起并执行成功 pipeline 时，门禁显示通过，解决方案详见流水线服务《用户指南》“流水线管理 > 配置流水线”章节。
E2E 单号未关联	当合并请求关联工作项后，门禁显示通过。 说明 需要在**“设置 > 策略设置 > 合并请求”**中勾选**“必须与 CodeArts Req 关联”**后，门禁功能生效。
检视门禁	当已检视的检视人数达到最小检视人数时，门禁显示通过。
审核门禁	当已审核的审核人数达到最小审核人数时，门禁显示通过。

步骤 4 当发起人通过以上合入条件后，合并人单击页面右上角“合入”按钮进行合入，反之，合并人可单击**“关闭”**将请求关闭。

----结束

您已经学完了新手教程，去探索一下更多的功能吧！

2 基于 Git 的代码托管入门

代码托管（CodeArts Repo）是面向软件开发者的基于 **Git** 的在线代码托管服务，是具备安全管控、成员/权限管理、分支保护/合并、在线编辑、统计服务等功能的云端代码仓库，旨在解决软件开发者在跨地域协同、多分支并发、代码版本管理、安全性等方面的问题。通过本章，您可以快速掌握 Git 和代码托管的基本使用方法，如果您对 Git 很熟悉，可以跳过本章。

在本章中，将模拟软件项目开发场景，使用代码托管提供的 Java War Demo 模板创建一个云端仓库，通过 SSH 方式将云端仓库克隆到本地 Git 环境中，在本地修改代码内容并推送修改至云端仓库。

前提条件

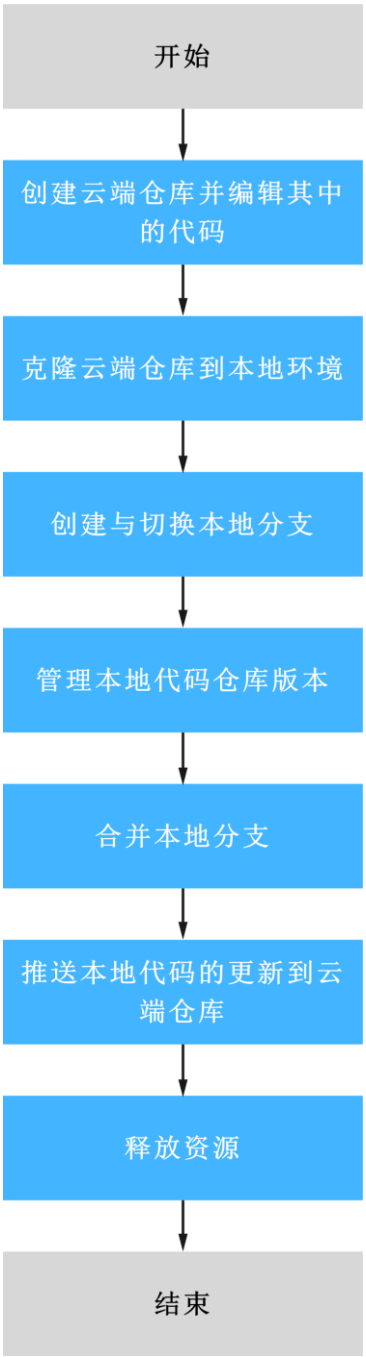
- 已经可用的项目，若没有，需新建项目。
- 下载安装 Git 客户端，详情请参考《用户指南》“Git 客户端安装配置”章节。
- 设置客户端与远程仓库的交互凭证，详情请参考《用户指南》“设置代码托管仓库的 SSH 密钥/HTTPS 密码”章节。
- 确保您的网络可以访问代码托管服务。

  请在 Git 客户端使用如下测试指令验证网络连通性。

  ```
  ssh -vT git@XXXXXXXX.com
  ```

  如果返回内容含有“connect to host XXXXXXXX.com port 22: Connection timed out”，则您的网络被限制，无法访问代码托管服务，请求助您本地所属网络管理员。

流程概览



所涉及到的操作或知识如下：

1. 创建云端仓库并编辑其中的代码
2. 克隆云端仓库到本地环境
3. 创建与切换本地分支
4. 管理本地代码仓库版本
5. 合并本地分支
6. 推送本地代码的更新到云端仓库

7. 释放资源

创建云端仓库并编辑其中的代码

如果您已有可用的云端仓库，可以跳过本节。

在本节中，您将使用已有模板“Java War Demo”快速创建一个新的仓库。“Java War Demo”模板是开发者们都熟悉的“Hello World”小程序的模板，您可以使用此模板创建的仓库体验代码托管的功能。

步骤 1 进入目标项目下的代码托管服务。

步骤 2 单击“普通新建”旁的图标，在扩展框中选择“按模板新建”，跳转到**“选择模板”**页面。

步骤 3 在“选择模板”页的搜索框中输入“Java War Demo”，在搜索结果中选择该模板，单击“下一步”。

说明

模板查询条件中的区域选择指的是模板文件存放的区域，不会影响使用模板新建仓库的所在区域，新建的仓库仍与仓库所隶属的项目在同一区域中。

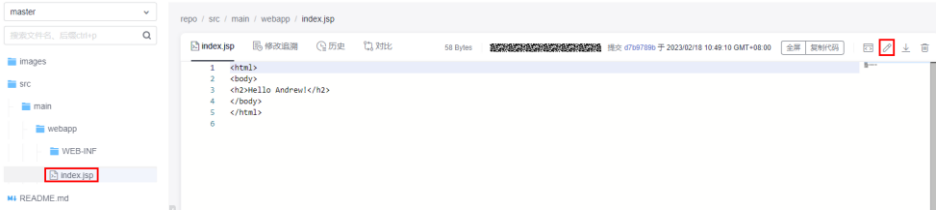
步骤 4 在“基本信息”页，填写仓库名称等信息，单击“确定”完成仓库创建。

创建完成后，可在代码托管服务首页中看到已创建的仓库，单击仓库名称进入仓库，可以查看仓库已有文件。

步骤 5 代码托管服务提供了线上编辑功能，开发者可以直接在云端修改仓库内的代码。

为了标识代码的唯一性，请跟随此步骤修改云端仓库的代码。

1. 在仓库列表页面，找到新创建的仓库，单击“仓库名称”进入仓库。
2. 在仓库**“代码”**页签下左侧的目录树中，打开“src/main/webapp/index.jsp”文件，单击图标，将“Hello World！”修改为任意内容，填写备注信息，并单击“确定”保存修改。



----结束

克隆云端仓库到本地环境

通过本节，您可以将克隆云端仓库到本地环境中，以下以使用 Git Bash 客户端为例。

步骤 1 获取仓库地址。

进入仓库详情，单击“克隆/下载”按钮获取 SSH 地址

步骤 2 打开 Git Bash 客户端。

在本地计算机上新建一个文件夹用于存放代码仓库，本案例中将其命名为“git-test”，进入文件夹，在空白处单击鼠标右键，打开 Git Bash 客户端。

说明

克隆仓库时会自动初始化，无需执行 init 命令。

步骤 3 输入如下命令，克隆云端仓库。

```
git clone 仓库地址
```

命令中“仓库地址”即本节第一步中获取的 SSH 地址。

第一次与云端仓库互动时，会询问是否保存指纹，需输入“yes”，才能进行通信。

执行成功后，进入“git-test”文件夹，您会看到多出一个与您在云端新建的仓库同名的文件夹，并且其中有一个隐藏的.git 文件夹，则说明克隆仓库成功。

步骤 4 此时您位于仓库上层目录，执行如下命令，进入仓库目录。

```
cd 仓库名称
```

进入仓库目录，可以看到此时 Git 默认为您定位到 master 分支。

```
Administrator@gittestcce MINGW64 /c/git-test
$ cd test_War_Java_Demo

Administrator@gittestcce MINGW64 /c/git-test/test_War_Java_Demo (master)
$
```

----结束

创建与切换本地分支

master 是仓库创建后默认的主分支，建议代码开发、发布、问题修复等在独立的分支开发，完成后合入主分支，保证主分支代码随时可用。本节将在本地环境中新建一个名为“dev”的分支，并切换到该分支上。

步骤 1 创建分支。

打开 Git Bash，进入仓库目录，执行如下命令，在本地环境新建一个名为“dev”的分支。

```
git branch dev
```

命令执行后无回显表示创建分支成功。

步骤 2 查看分支（可选）。

执行如下命令查看本地仓库分支。

```
git branch
```

```
Administrator@andrew-windows12-01 MINGW64 /c/git-test/      -demo-java (master)
$ git branch
  dev
* master
```

可以看到当前有 master、dev 两条分支，并且目前处于 master 分支，可以理解为本地有 master、dev 两套内容一样的代码。

步骤 3 切换分支。

执行如下命令，切换当前分支至“dev”分支。

```
git checkout dev
```

命令执行后，可以看到当前路径后的分支为“(dev)”即表示分支切换成功。分支切换后，对本地仓库的所有修改将保存在当前分支上。

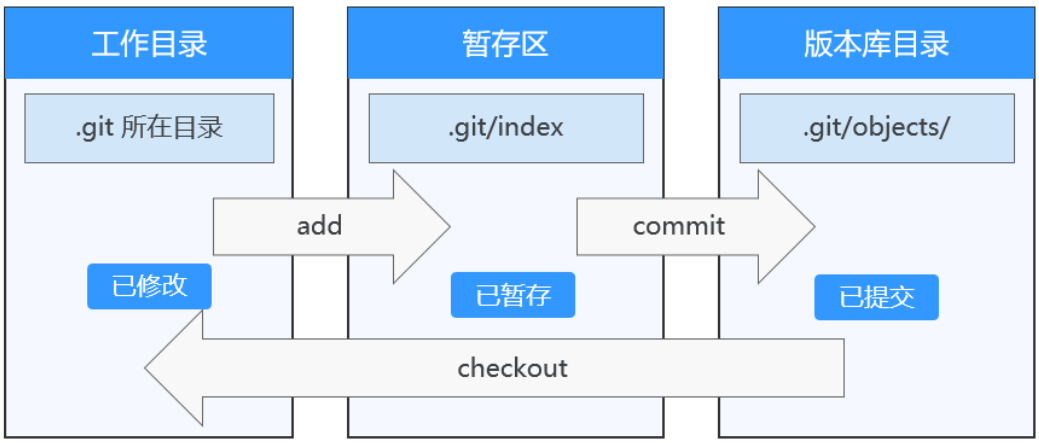
```
Administrator@andrew-windows12-01 MINGW64 /c/git-test/      -demo-java (master)
$ git checkout dev
Switched to branch 'dev'

Administrator@andrew-windows12-01 MINGW64 /c/git-test/      -demo-java (dev)
$
```

----结束

管理本地代码仓库版本

本节中，将修改本地仓库中“\src\main\webapp\index.jsp”文件里的内容，并通过 add 及 commit 命令将修改提交至本地仓库。

Git 本地仓库中的数据有三种状态，分别是“已修改”、“已暂存”和“已提交”。当您对仓库中的文件做出修改后，该文件状态为“已修改”，您可以通过 **add** 命令将该修改追加到本地的暂存区，此时状态为“已暂存”，再通过 **commit** 命令将修改提交到本地版本库进行管理，每次提交都会生成对应的版本和版本号，通过版本号可以进行版本的切换、回滚，下图为 Git 本地仓库的基本工作示意图。在同一版本中还可以同时存在多个分支，每个分支又相当于独立的版本。



步骤 1　修改 dev 分支的代码。

在之前章节已经克隆云端仓库到本地环境，并且切换到了 dev 分支，现在要对 dev 分支的代码进行修改，打开本地仓库文件夹找到 index.jsp 文件（仓库文件夹\src\main\webapp\index.jsp），使用任意文本编辑软件打开，可以看到在创建云端仓库并编辑其中的代码时修改的内容，此时本地的两个仓库分支（dev、master）与云端仓库的版本内容是一样的。

```
<html>
<body>
<h2>Hello Andrew!</h2>
</body>
</html>
```

将内容修改为“Hello git!!!”并保存、关闭文件，因为之前已经切换到了 dev 分支，所以此时的修改仅仅将被记录在 dev 分支中。

步骤 2　查看修改记录（可选）。

使用 status 命令查看当前分支与暂存区的差异。

```
git status
```

```
Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (dev)
$ git status
On branch dev
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/main/webapp/index.jsp

no changes added to commit (use "git add" and/or "git commit -a")

Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (dev)
$
```

如上图，git 识别到了您的修改并提示您还没有将修改加入暂存区和提交到本地版本库。

步骤 3 将修改内容追加到本地暂存区中。

使用 add 指令将修改加入本地暂存区。

```
git add .
```

或

```
git add src/main/webapp/index.jsp
```

使用“git add .”意味着将全部修改加入暂存区，您也可以使用文件的路径来单独将某个修改的文件加入暂存区，如果没有任何回显，就是执行成功了，此时可以再次使用 status 命令，如下图可以看到此时修改内容已经进入暂存区等待提交。

```
Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (dev)
$ git add src/main/webapp/index.jsp

Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (dev)
$ git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/main/webapp/index.jsp


Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (dev)
$
```

步骤 4 将已暂存的内容提交到本地版本库。

使用 commit 指令将暂存区的内容提交到版本库，-m 后面跟本次提交的标签。

```
git commit -m "本次提交的标签"
```

本示例中，看到返回“1 file changed”则表示提交成功，此时本地的 master 分支与 dev 分支已经锁定了两个版本的代码，您可以使用 checkout 命令切换分支然后在仓库文件夹中查看\src\main\webapp\index.jsp 文件的内容，会发现当处于不同分支时，看到的是不同的文件版本。

----结束

合并本地分支

在前面的章节中，新建了 dev 分支，并修改了分支中的文件内容，在实际开发中，一般会有多条开发（dev）分支同时存在，所以在将代码提交到远程仓库前，一般将已经

完成修改的分支都合并到 master 分支，以保证 master 分支是本地最全最新的可提交代码版本。

步骤 1 使用如下命令切换到 master 分支。

```
git checkout master
```

```
Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (dev)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (master)
$
```

步骤 2 使用 merge 命令将 dev 分支的修改合并到 master 分支。

```
git merge dev
```

```
Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (master)
$ git merge dev
Updating 5e42dcf..e348160
Fast-forward
 src/main/webapp/index.jsp | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

----结束

推送本地代码的更新到云端仓库

使用 push 命令将本地 master 分支提交到远端仓库。

```
git push origin master
```

```
Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (master)
$ git push origin master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 902 bytes | 902.00 KiB/s, done.
Total 12 (delta 2), reused 0 (delta 0), pack-reused 0
To                                    .com:Andrew-test00001/      -demo-java.git
   5e42dcf..e348160  master -> master

Administrator@gittestcce MINGW64 /c/git-test/      -demo-java (master)
$
```

上图是推送成功的示例，此时去代码托管服务的仓库列表，单击对应仓库名称，查看文件\src\main\webapp\index.jsp，可以看到您在本地仓库修改的内容，并能发现“更新时间”和“备注”的变化。

至此完整进行了一次修改远程代码托管仓库的操作。

释放资源

在本节中，将删除您在本教程中创建的代码托管云端仓库和项目，以免产生仓库存储空间使用费用。

> **⚠ 注意**
>
> 删除的项目和仓库无法恢复。

步骤 1 删除云端仓库。

1. 进入目标项目下的代码托管服务。
2. 单击 ··· 图标，在展开选项中，单击“删除仓库”按钮，按提示输入仓库名后单击“确认”按钮，即完成仓库删除。

步骤 2 删除本地仓库（可选）。

如果您不再需要本地仓库，可以将其删除以释放存储空间，直接删除仓库文件夹即可。

----结束