代码托管

最佳实践

文档版本	01
发布日期	2023-07-05

华为技术有限公司

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。

目 录

1 Git on CodeArts Repo

1.1 概述

文档目的

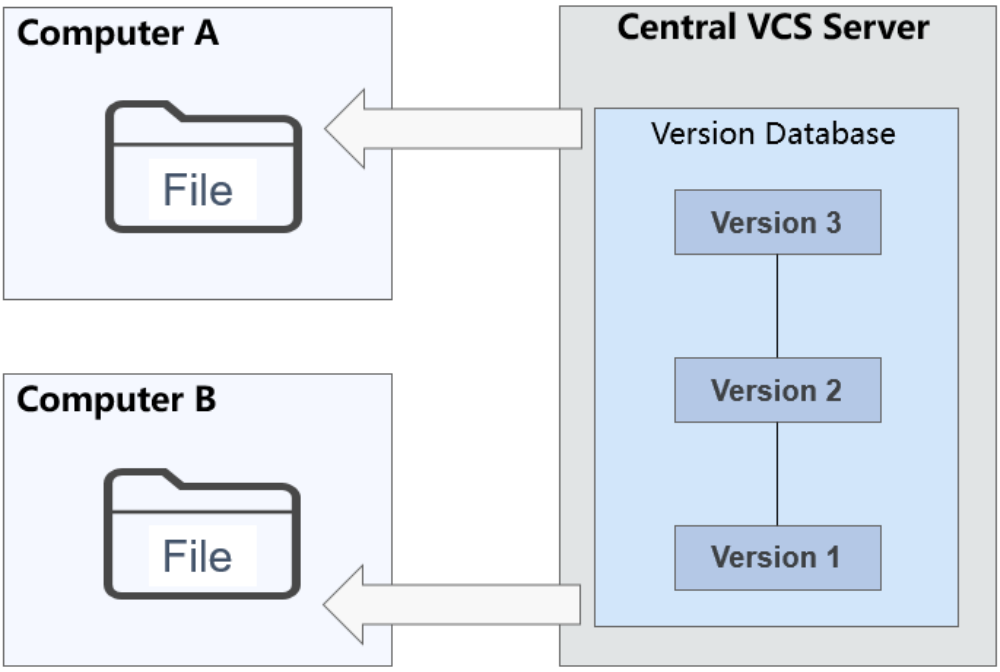
基于CodeArts实践所编写，用于帮助已经掌握或想要掌握Git的开发者，更好的应用Git，以及更好的将Git与CodeArts结合应用。

Git 概述

从狭义上来说，版本控制系统是软件项目开发过程中管理代码所有修订版本的软件，能够存储、追踪文件的修改历史，记录多个版本的开发和维护，事实上我们可以将任何对项目有帮助的文档交付版本控制系统进行管理。版本控制系统（Version Control Systems）主要分为两类，集中式和分布式。

集中式版本控制系统

集中式版本控制系统的特点是只有一台中央服务器，存放着所有研发数据，而其它客户端机器上保存的是中央服务器最新版本的文件快照，不包括项目文件的变更历史。所以，每个相关人员工作开始前，都需要从这台中央服务器同步最新版本，才能开始工作，如下图所示。



常见的集中式版本控制系统为CVS、VSS、SVN、ClearCase。

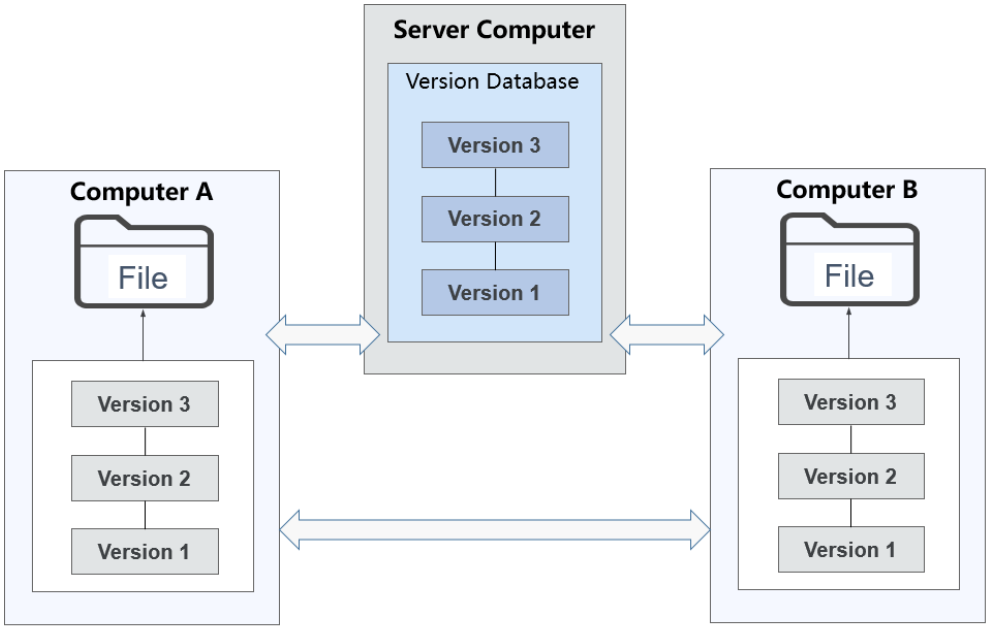
集中式版本控制系统的优点与缺点如下表所示。

表 1-1 集中式版本控制系统描述

优点	缺点
• 操作简单，使用没有难度，可轻松上手。 • 文件夹级权限控制，权限控制粒度小。 • 对客户端配置要求不高，无需存储全套代码。	• 网络环境要求高，相关人员必须联网才能工作。 • 中央服务器的单点故障影响全局，如果服务器宕机，所有人都无法工作。 • 中央服务器在没有备份的情况下，磁盘一旦被损坏，将丢失所有数据。

分布式版本控制系统

分布式版本控制系统的特点是每个客户端都是代码仓库的完整镜像，包括项目文件的变更历史。所有数据分布的存储在每个客户端，不存在中央服务器。可能有人会问，我们公司使用Git分布式存储工具，也有“中央服务器”啊？其实，这个所谓的“中央服务器”仅仅是用来方便管理多人协作，任何一台客户端都可以胜任它的工作，它和所有客户端没有本质区别，如下图所示。



常见的分布式版本控制系统为Git、Mercurial、Bazaar、Bitkeeper。

分布式版本控制系统的优点与缺点如下表所示。

表 1-2 分布式版本控制系统描述

优点	缺点
• 版本库本地化，版本库的完整克隆，包括标签、分支、版本记录等。 • 支持离线提交，适合跨地域协同开发。 • 分支切换快速高效，创建和销毁分支廉价。	• 学习成本高，不容易上手。 • 只能针对整个仓库创建分支，无法根据目录建立层次性的分支。

1.2 CodeArts Repo 云端操作

准备工作

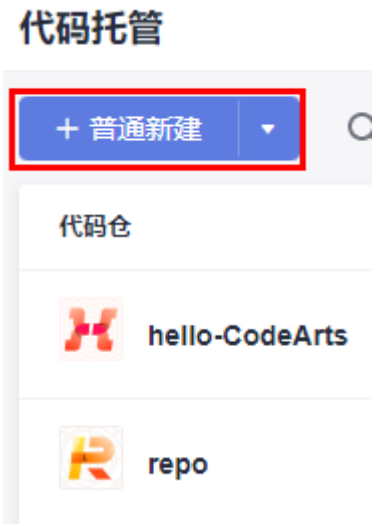
- 成为代码托管（CodeArts Repo）用户
- **已有Git客户端**
- **已创建好的项目**

云端仓库功能

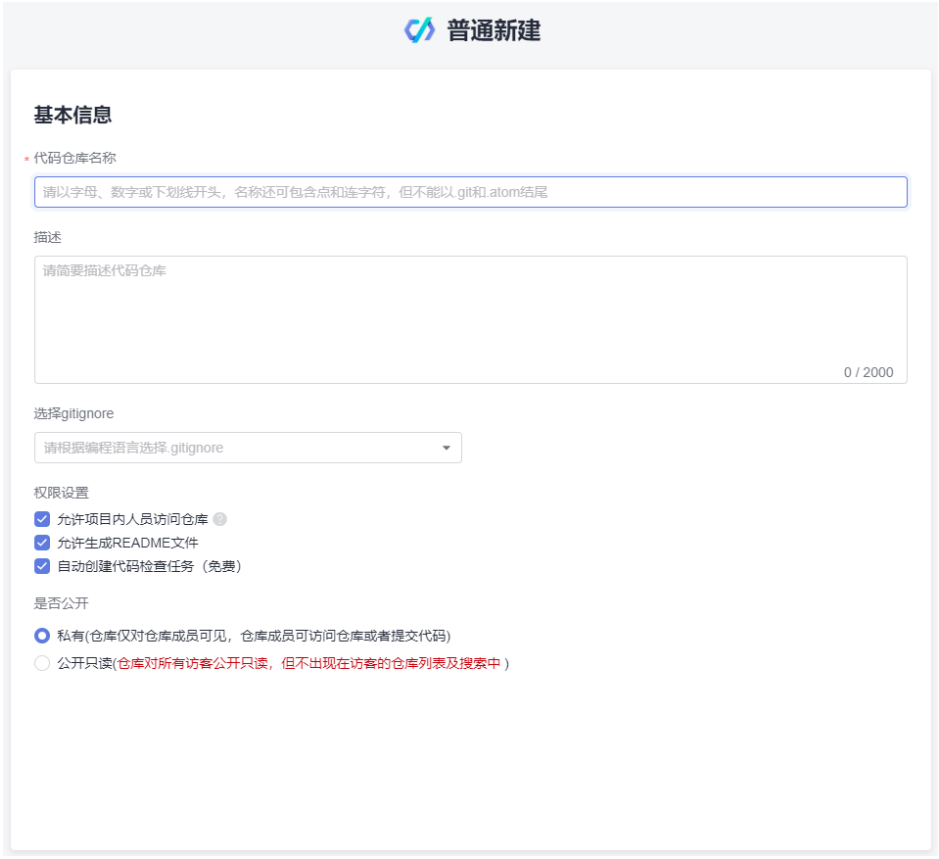
云端仓库功能支持新建仓库、仓库克隆、分支管理、标签管理、提交代码、拉取代码、推送代码、代码阅读、在线修改、仓库成员管理、密钥管理等，更多仓库功能介绍请参见**产品概述**。

新建空仓库

1. 在目标项目下的代码托管服务中，单击“普通新建”按钮，如下图所示。



2. 填写仓库的基本信息，下图所示。



3. 单击“确定”按钮，完成仓库新建，跳转到仓库列表。

设置 SSH 密钥/HTTPS 密码

后续需要在本地客户端进行代码仓库的克隆/推送，SSH密钥和HTTPS密码是客户端和服务端交互的凭证，需要先对它们进行设置。

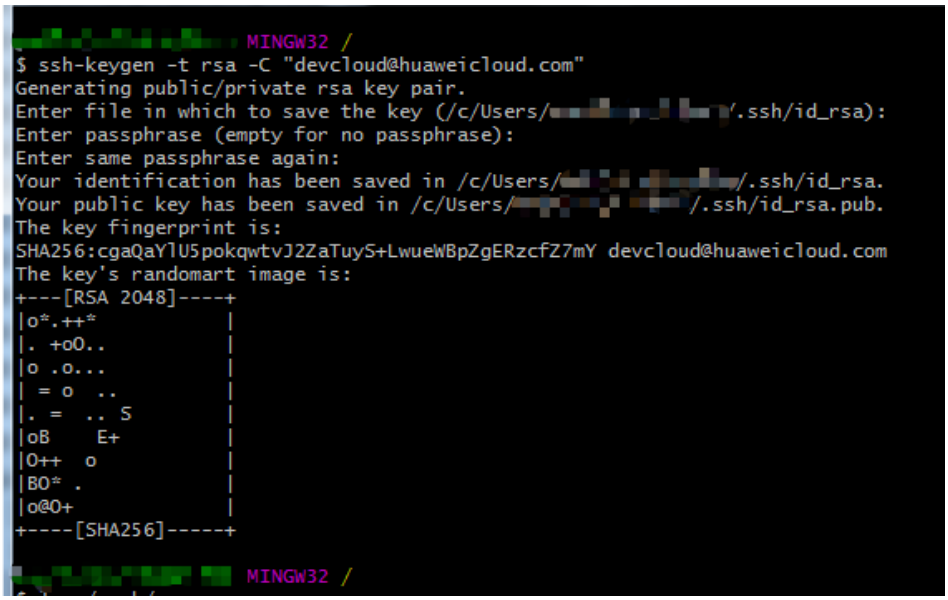
设置SSH密钥

SSH密钥是使用SSH协议和代码托管服务端交互的凭证，如果您使用windows下的Git Bash客户端并在其中已经生成，此步骤可以略过。

步骤1 打开Git客户端（Git Bash或linux的命令行窗口），输入以下命令行：

```
ssh-keygen -t rsa -C "<您的邮箱>"
```

然后输入3个回车（Enter键）即可，生成的SSH秘钥对默认在“~/.ssh/id_rsa、~/.ssh/id_rsa.pub”位置，如下图所示。

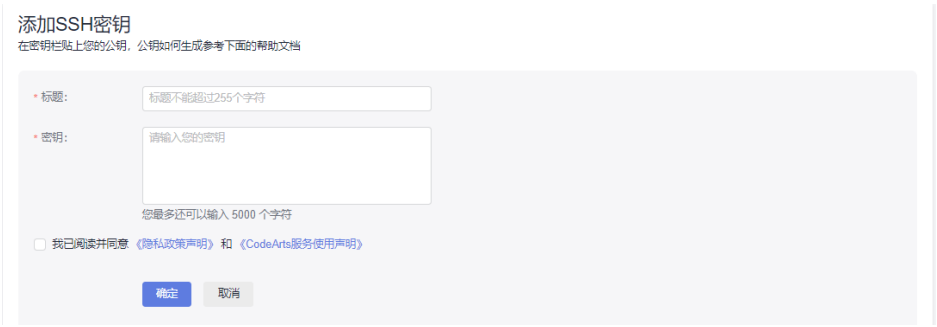


步骤2 添加SSH密钥到代码托管服务端：

打开Git客户端（Git Bash或linux的命令行窗口），将SSH密钥“~/.ssh/id_rsa.pub”的内容打印出来。

步骤3 复制上述的SSH密钥内容，登录您的代码托管服务仓库列表页，单击右上角昵称，单击“个人设置 > SSH密钥管理”，进入页面。

步骤4 在“SSH密钥管理”页面，单击“添加SSH密钥”，弹出“添加SSH密钥”页面，填写下图中信息，单击**“确定”**，页面会提示您操作成功。



您已经设置好了SSH密钥，您可以继续设置HTTPS密码。

----结束

设置HTTPS密码

HTTPS密码是使用HTTPS协议和代码托管服务端交互的凭证，设置步骤如下：

步骤1 登录您的代码托管服务仓库列表页，单击右上角昵称，单击“个人设置 > HTTPS密钥管理”，进入页面。

步骤2 单击“自行设置密码”，再单击“修改”进入“重设密码”页面。（如果您之前自主设置过HTTPS密码并正在使用，直接单击“修改”）

步骤3 填写新密码与邮箱验证码，勾选**“我已阅读并同意《隐私政策声明》和《CodeArts服务使用声明》”**，单击**“保存”**，页面会提示您操作成功。

----结束

1.3 Git 本地研发场景

背景介绍

在CodeArts云端创建一个README文件的空仓库，然后架构师或者项目负责人需要把本地框架代码推送到这个空仓库，最后，其他开发人员将云端架构代码克隆到本地，进行增量应用开发。

说明

- Git代码传输支持SSH和HTTPS两种传输协议，本节基于SSH传输协议进行的操作。
- 如果想使用HTTPS方式，直接下载HTTPS密码，当克隆、推送代码时直接输入HTTPS用户名密码即可。
- 同一仓库SSH和HTTPS的地址不同。

推送架构代码

1. 打开本地框架代码，确保根目录名与云端创建的代码仓库名一致，在根目录下右键打开**Git bash**终端。
2. 推送本地代码到云端。

   在当前**Git Bash**终端依次输入如下命令：

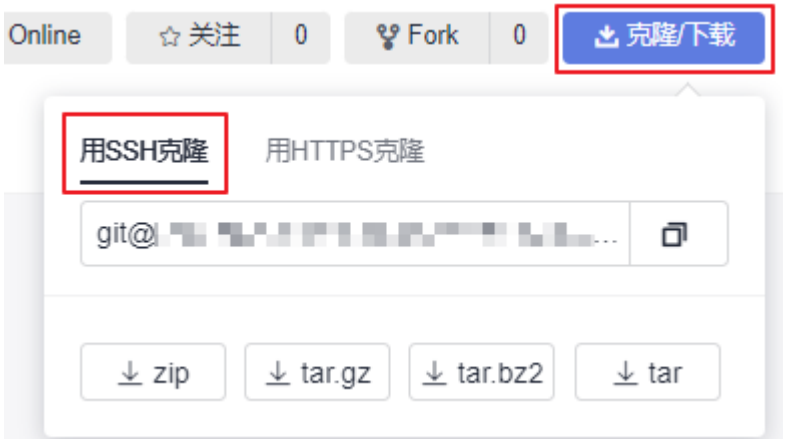
   a. 初始化本地代码仓库，执行该命令后，在“D:/code/repo1/”下多了一个“.git”文件夹。

   ```
   $ git init
   ```

   b. 关联云端代码仓库。

   ```
   $ git remote add origin repoUrl
   ```

   仓库地址如下图进入仓库详情页，单击”**克隆/下载**“，所示单击红色方框处获取。

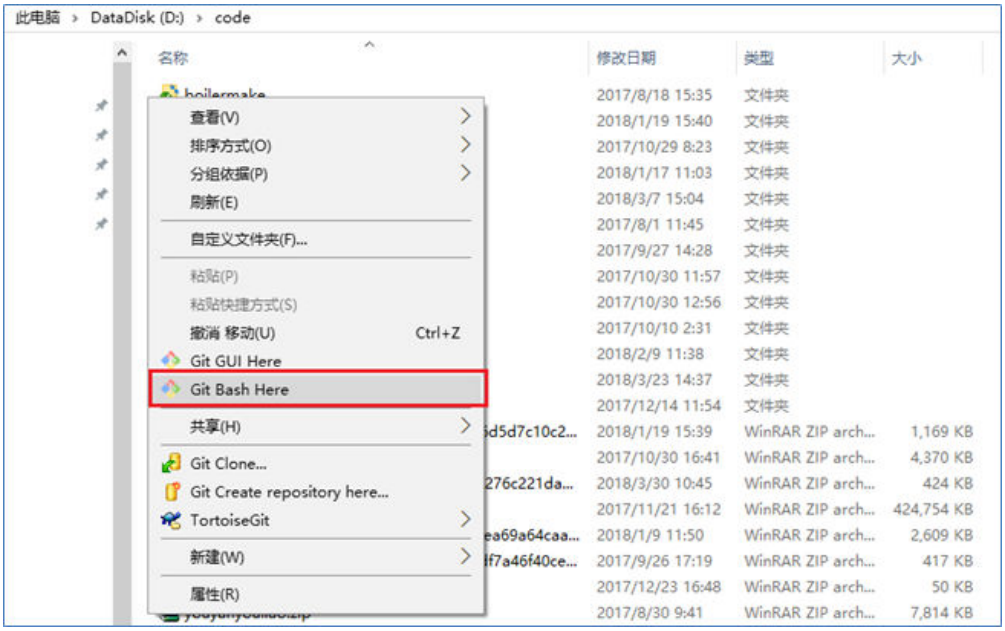
   c. 推送代码到云仓库。

   ```
   $ git  add .
   $ git  commit  -m "init project"
   $ git  branch --set-upstream-to=origin/master master
   $ git  pull  --rebase
   $ git  push
   ```

克隆代码

开发人员在本地准备克隆云端架构代码。

1. 在准备把代码克隆到的目标文件夹下，右键打开**Git bash**终端，如下图所示。



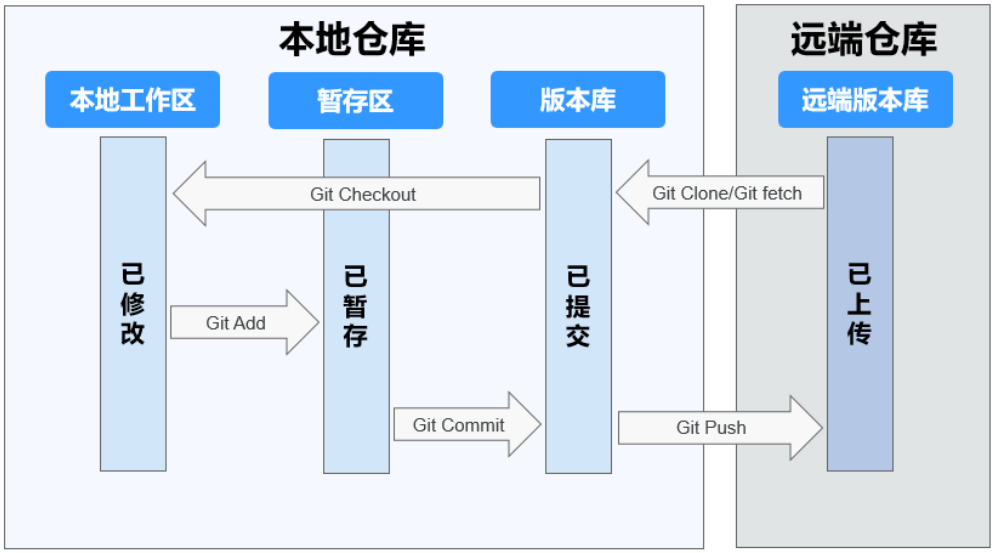
2. 克隆仓库，URL地址如下图所示单击红色方框处获取。

```
$ git clone repoUrl  //将代码从远端仓库clone到本地
```

代码提交

一次修改被成功提交到远端仓库会历经四个阶段：“1本地工作区 > 2缓存区 > 3版本库 > 4远端版本库”。

通过执行相应的Git命令，文件在这四个区域跳转，并呈现不同的状态，如下图所示。



主要涉及如下三步操作：

1. **#git add/rm filename** *//将新增、修改或者删除的文件增加到暂存区*
2. **#git commit –m “commit message”** *//将已暂存的文件提交到本地仓库*
3. **#git push** *//将本地代码仓库修改推送到远端仓库*

分支操作

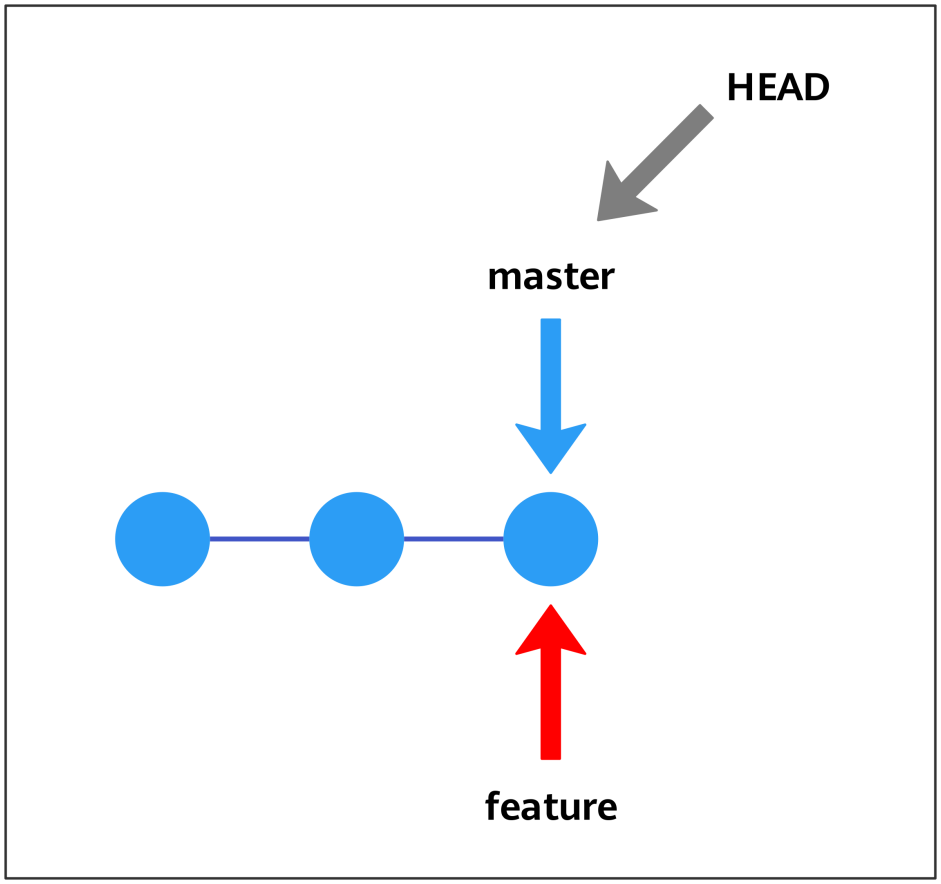
- 新建分支

  Git新建分支的本质就是创建一个指向最后一次提交的可变指针，所以，Git分支的创建不是复制版本库的内容，仅仅是新建了一个指针，它以40个字符长度SHA-1字串形式保存在文件中。

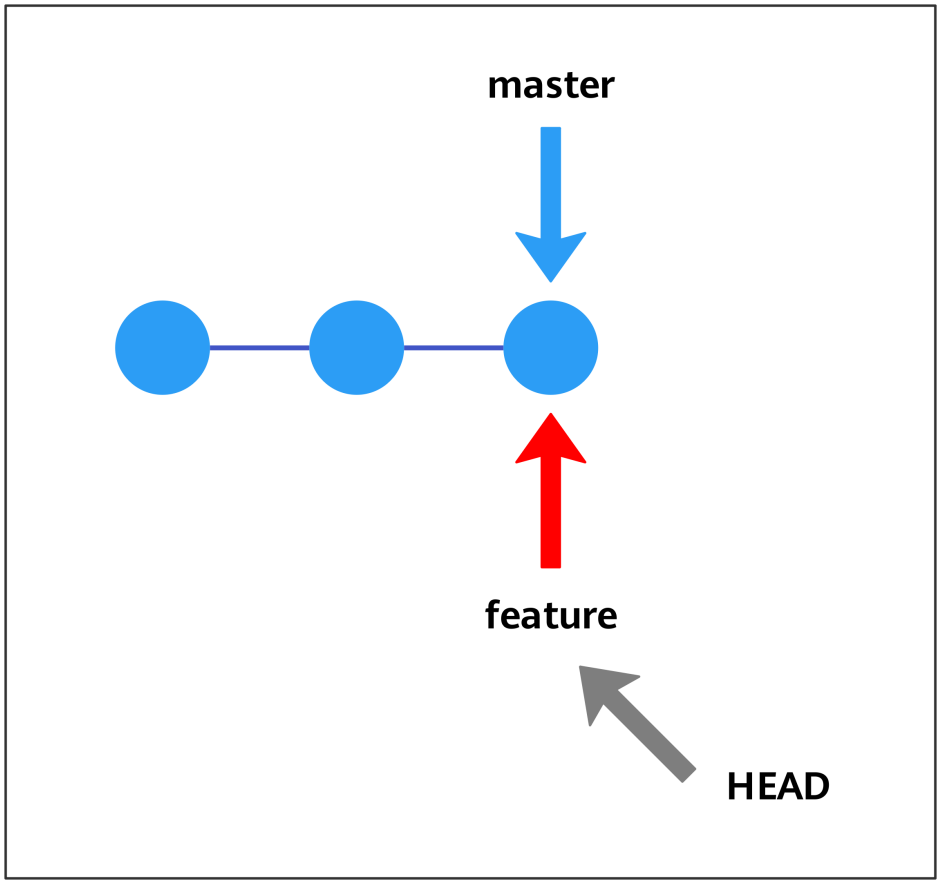
  ```
  #git branch branchName commitID
  ```

  基于commitID即某一个全球版本号拉出新分支，如果没有commitID则基于当前分支的HEAD拉出新分支。

  例如，新建feature分支，执行的命令为**git branch feature**，如下图所示。



- 切换分支
  命令如下。
  ```
  #git checkout branchName
  ```
  例如，切换到feature分支，执行的命令为**git checkout feature**，如下图所示。



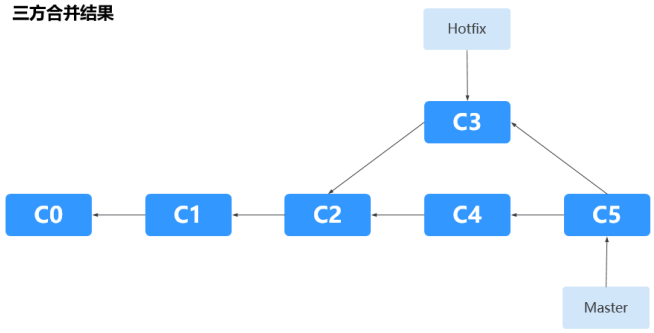
- 分支合并

  无论哪种工作流都会涉及到分支合并（把一个分支中的修改整合到当前分支），主要有两种方法：三方合并（merge）和衍合（rebase）。通过对同一种场景进行不同操作体会两种合并方法的区别。

  场景：master分支新增了C4节点， hotfix分支新增了C3节点，现将hotfix分支合并到master分支：

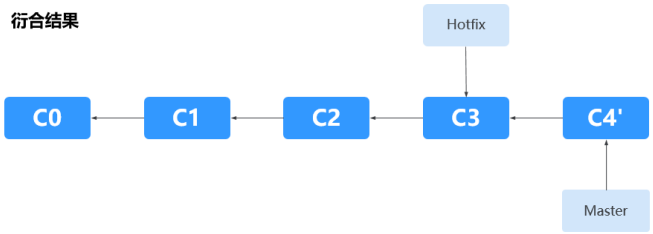
  a. 三方包括hotfix新增节点C3，master新增节点C4，以及两者的共同祖先节点C2。这种合并操作简单，但新增合并节点C5，形成了环形，版本记录可读性差，如下图所示。

  ```
  #git checkout master
  #git merge hotfix
  ```

  b. 衍合先将master分支新增节点C4以补丁形式保存在.git/rebase目录中，然后同步hotfix分支最新代码，再应用补丁C4’，如下图所示。


```
#git checkout master
#git rebase hotfix
```



- 冲突解决
  a. 场景一：两个合并分支修改了同一行代码

```
$ cat doc/README.txt
User1 hacked.
<<<<<<< HEAD
Hello, user2. #当前分支修改方法
=======
Hello, user1. #源分支修改方法
>>>>>>> a123390b8936882bd53033a582ab540850b6b5fb
User2 hacked.
User2 hacked again.
```

解决方法：

i. 分析哪种修改方法正确，手动合并。
ii. 提交修改。

  b. 场景二：文件被重命名为不同的名字

解决方法：

i. 确认哪个名字是正确的，删除错误的。
ii. 提交修改。

2 将仓库迁移至代码托管平台

本实践将展示如何将您自有的**本地仓库**或**云端仓库**迁移至代码托管服务。

应用场景

随着**“代码上云”**浪潮的推动，仓库的自主迁移趋于常态化，代码托管针对准备进行仓库迁移的用户提供完整的操作指导，帮助您将仓库迁移至代码托管服务。

实现原理

代码托管服务（CodeArts Repo）结合仓库的存储方式提供以下迁移方案：

- **HTTP在线导入**

  通过HTTP协议直接将您的远程仓库导入到代码托管中，全程线上操作，但导入仓库的时长会受到网络条件及仓库容量的影响。

  说明

  仓库容量相对较大的云端仓库推荐使用**Git客户端推送**的方式进行迁移。

- **Git客户端推送**

  通过使用Git客户端将本地仓库中的代码文件推送至代码托管服务。

  - 将项目文件存放在本地计算机的用户，建议先将本地项目文件初始化成Git仓库，再使用Git客户端进行迁移。
  - 新建仓库对于仓库容量相对较大的云端仓库，建议先将云端仓库克隆或下载到本地，再使用Git客户端进行迁移。

前提条件

- 已有可用项目，如果没有，请先新建项目。
- 已有可用仓库，如果没有，请先新建仓库。
- 在仓库迁移的过程中，要保持网络稳定畅通。
- 被迁移仓库的容量不能超过2GB，否则新建的代码托管仓库会被冻结，不可使用。

HTTP 在线导入

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**“代码 > 代码托管”**，进入代码托管服务。

步骤3 在代码托管仓库列表页，单击**“普通新建”**旁的图标，在下拉列表中单击**“导入外部仓库”**。

步骤4 在**“填写外部仓库信息”**页面，根据实际情况填写以下参数。

表 2-1 外部仓库参数说明

参数项	是否必填	说明
源仓库路径	是	填写源仓库的仓库地址，以（http://）或（https://）开头，以（.git）结尾。
源仓库访问权限	是	• **不需要用户名/密码**：如果源仓库是开源仓库（公开仓），请勾选该选项。 • **需要用户名/密码**：如果源仓库是私有仓库，请勾选该选项，并填写HTTPS克隆代码时的用户名及密码。

步骤5 勾选**“我已阅读并同意《隐私政策声明》和《CodeArts服务使用声明》”**，单击**“下一步”**。

步骤6 在**“创建仓库”**页面，根据实际情况填写以下参数。

表 2-2 字段说明

字段名称	是否必填	备注说明
代码仓库名称	是	请以字母、数字、下划线开头，名称还可包含点和连字符，但不能以.git、.atom结尾，限制200个字符。
描述	否	为您的仓库填写描述，限制2000个字符。
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，也会自动同步到已经存在的仓库中。 • **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。

字段名称	是否必填	备注说明
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，可以选择开源许可证作为备注。
分支设置	是	可选择同步源仓库的**默认分支**或**全部分支**。
增加定时同步	否	勾选“**增加定时同步**”功能： • 每天自动从源仓库导入仓库的默认分支。 • 仓库将成为只读镜像仓库，不能写入，并且只同步当前创建仓库的默认分支对应的第三方仓库的分支。

步骤7 单击“**确定**”，完成仓库导入，跳转到仓库列表页。

----结束

Git 客户端推送（以 Git Bash 为例）

说明

在进行Git Bash客户端推送之前请确保客户端**已关联代码托管服务**。

步骤1 进入目标代码托管服务。

步骤2 将本地仓库初始化为Git仓库，用于与代码托管仓库进行关联。

在您的仓库中打开Git Bash客户端，执行以下命令：

```
git init
```

初始化成功如下图，此时当前文件夹已经是本地Git仓库了。

步骤3 将本地仓库与代码托管仓库进行绑定。

1. 进入代码托管仓库，获取仓库地址。
2. 在本地使用remote命令，将本地仓库与代码托管仓库进行绑定。

   ```
   git remote add 仓库别名 仓库地址
   ```

   示例为：

   ```
   git remote add origin git@*****/java-remote.git  #复制使用时注意换成您自己的仓库地址
   ```


说明

- 一般用origin作为仓库别名，因为当您从远程仓库clone到本地时，默认产生的别名就是origin，当然您也可以使用任意别名。
- 如果提示仓库名重复，更换一个即可。
- 无回显即为绑定成功。

步骤4 将代码托管仓库master分支拉取到本地仓库。

此步骤主要是避免冲突。

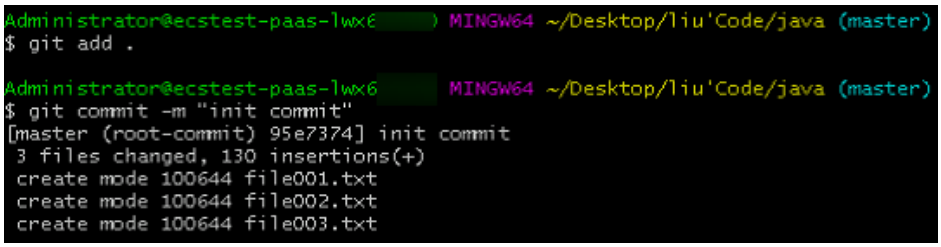
```
git fetch origin master  #复制使用时 注意是否需要将origin替换为您仓库的别名
```

步骤5 将本地代码文件提交到master分支。

依次执行：

```
git add .
git commit -m "您的提交备注"
```

下图为成功的执行。



步骤6 将本地master分支与代码托管master分支进行绑定。

```
git branch --set-upstream-to=origin/master master  #复制使用时 注意是否需要将origin替换为您仓库的别名
```

成功执行如下图所示，提示您已经将合并后的仓库放在工作区与版本库。

步骤7 合并代码托管仓库与本地仓库的文件，并存储在本地。

```
git pull --rebase origin master  #复制使用时 注意是否需要将origin替换为您仓库的别名
```

成功执行如下图所示，提示您已经将合并后的仓库放在工作区与版本库。

步骤8 将本地仓库推送覆盖代码托管仓库。

因为之前已经进行了绑定，直接push即可。

```
git push
```

成功后，再直接拉取pull，验证代码托管仓库与本地仓库版本相同，如下图。

----结束

