代码托管

快速入门

文档版本 04

发布日期 2021-04-21

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默示的声明或保证。

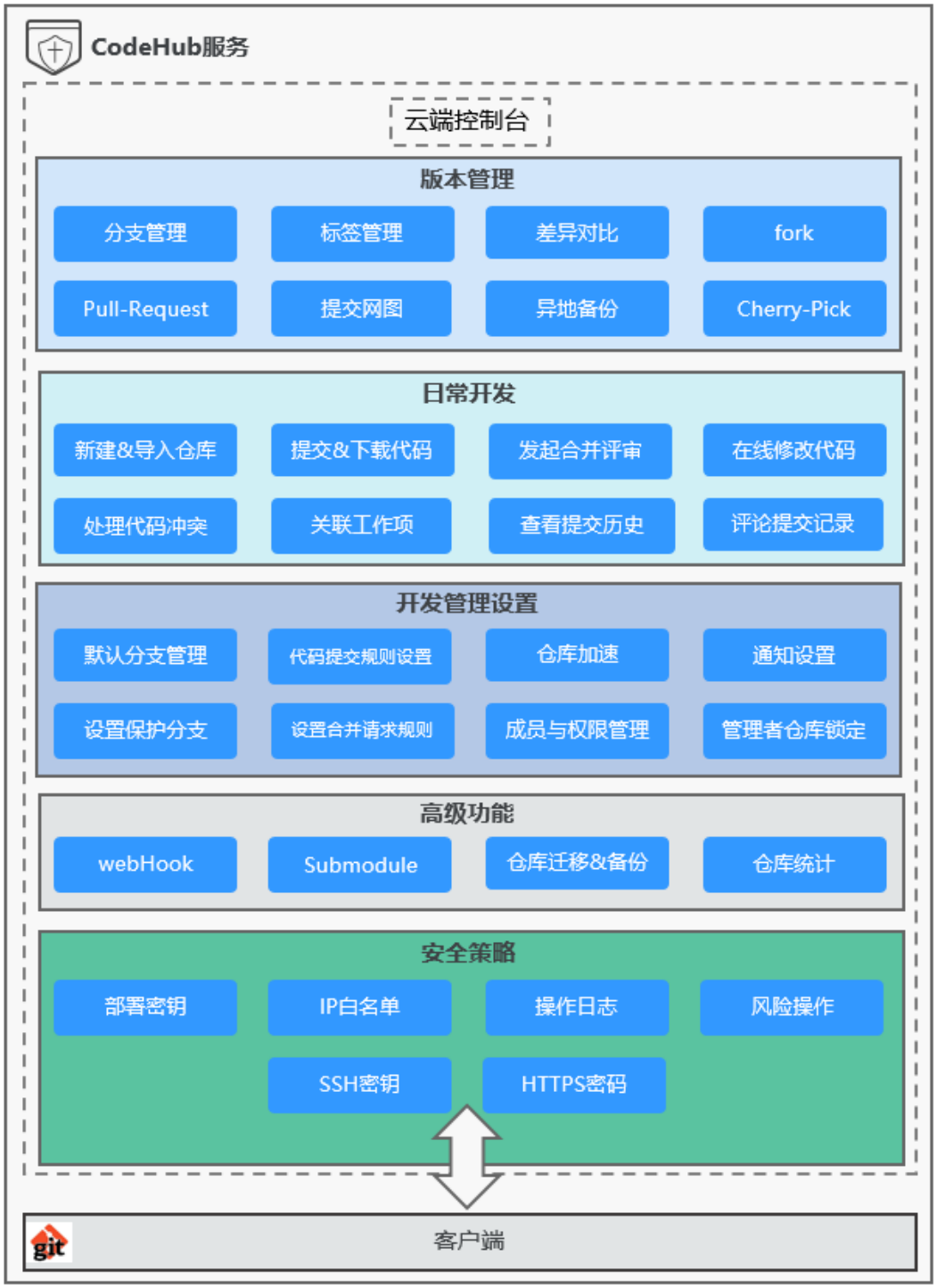
由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 代码托管控制台新手指引

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

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。



2 基于 Git 的代码托管入门

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。

代码托管提供基于Git的云端仓库，供用户存储管理代码。

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**新建项目**，或有可用的项目。
- **下载安装Git客户端**。
- **设置客户端与远程仓库的交互凭证**。
- 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@codehub.devcloud.huaweicloud.com
  ```

  如果返回内容含有“connect to host codehub.devcloud.huaweicloud.com port 22: Connection timed out”，如下图所示，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。

  ```
  $ ssh -vT git@codehub.devcloud.huaweicloud.com
  OpenSSH_8.1p1, OpenSSL 1.1.1d  10 Sep 2019
  debug1: Reading configuration data /etc/ssh/ssh_config
  debug1: Connecting to codehub.devcloud.huaweicloud.com [117.78.39.149] port 22.
  debug1: connect to address 117.78.      port 22: Connection timed out
  ssh: connect to host codehub.devcloud.huaweicloud.com port 22: Connection timed
  out
  ```

流程概览

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 进入仓库列表页（请确认项目所在区域并手动切换区域）。

步骤2 按模板“Java War Demo”创建新仓库。

1. 单击下图中图示1的，选择“按模板新建”。

2. 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。

3. 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3 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“文件”页面左侧的目录树中，打开“src/main/webapp/index.jsp”文件，单击“编辑”按钮，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。

在仓库列表中，单击您创建的仓库URL的“SSH”，复制此仓库的SSH地址到剪切板中，通过这个地址，可以在本地计算机连接云端仓库。

仓库名称	类型	仓库URL	合并...	仓库容量 (GIT | ...
test_Java_War_D...	私有	SSH | HTTPS	0	0.43M | 0.00M

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。常见的分支及命名请参见**Git工作模式**。

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/liuxin-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。

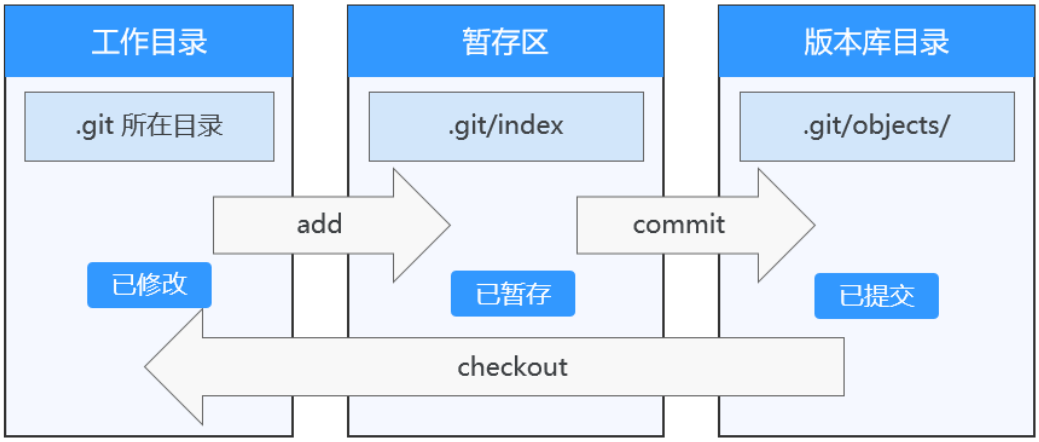
```
Administrator@andrew-windows12-01 MINGW64 /c/git-test/liuxin-demo-java (master)
$ git checkout dev
Switched to branch 'dev'

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

----结束

管理本地代码仓库版本

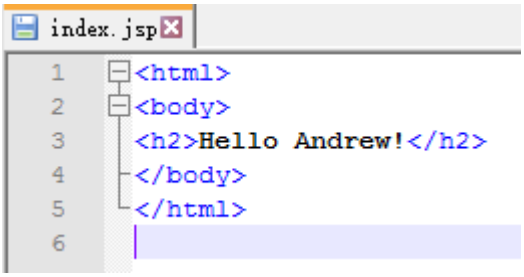
本节中，将修改本地仓库中“\src\main\webapp\index.jsp”文件里的内容，并通过add及commit命令将修改提交至本地仓库。

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。



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/liuxin-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/liuxin-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/liuxin-demo-java (dev)
$ git add src/main/webapp/index.jsp

Administrator@gittestcce MINGW64 /c/git-test/liuxin-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/liuxin-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"
```

```
Administrator@gittestcce MINGW64 /c/git-test/liuxin-demo-java (dev)
$ git commit -m dev分支的第一次提交
[dev e348160] dev分支的第一次提交
 1 file changed, 1 insertion(+), 1 deletion(-)
```

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/liuxin-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/liuxin-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/liuxin-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/liuxin-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 codehub.devcloud.huaweicloud.com:Andrew-test00001/liuxin-demo-java.git
   5e42dcf..e348160  master -> master

Administrator@gittestcce MINGW64 /c/git-test/liuxin-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 删除代码托管上的云端仓库。

1. 进入项目，从上方导航栏“代码 > 代码托管”进入项目的**仓库列表页**（请确认项目所在区域并手动切换区域）。
2. 单击 ⋮ 按钮，在展开选项中，单击“删除”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 删除项目（可选）。

如果仓库所属的项目仅是用于学习本教程，并无其它在用内容，建议在完成本教程后删除项目。

1. 进入**项目列表页**。
2. 鼠标悬停在需要删除的项目上，单击出现的 ⚙ 按钮，进入项目基本信息页面，单击“删除项目”按钮，按提示输入项目名称后单击“删除”按钮，即完成项目删除。

步骤3 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

