git文档

维护纪录

维护事项	维护人	维护时间	维护版本
初始版本	双星级	2017/5/23	Ver 1.0
修改发布流程	双星级	2017/08/22	Ver 1.1

---

1. 环境搭建

   git官网下载安装git： https://git-scm.com/downloads

   安装完成之后，输入git --version可以查看安装的git版本

2. 环境配置

   SSH key 提供了一种与github通信的方式

   配置ssh key 步骤：

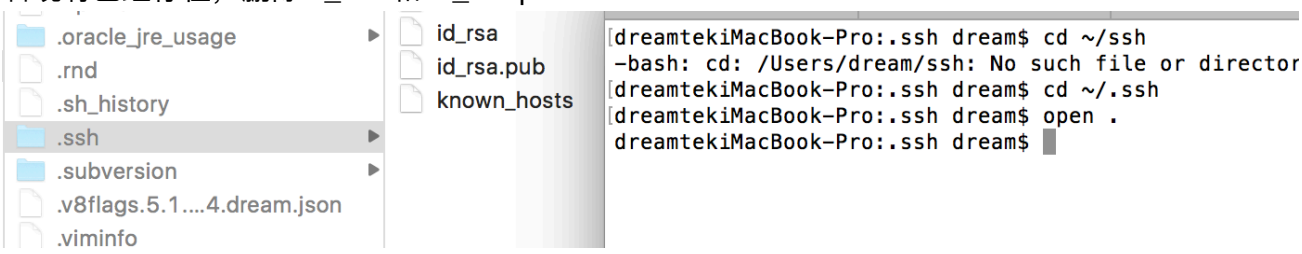
   - 检查ssh key是否存在

     控制台输入： ls -al ~/.ssh

     ```
     dreamtekiMacBook-Pro:.ssh dream$ ls -al ~/.ssh
     total 40
     drwx------   6 dream  staff   204  5 23 13:37 .
     drwxr-xr-x+ 39 dream  staff  1326  5 23 13:33 ..
     -rw-r--r--@  1 dream  staff  6148  5 23 11:19 .DS_Store
     -rw-------   1 dream  staff  1679  5 23 13:34 id_rsa
     -rw-r--r--   1 dream  staff   411  5 23 13:34 id_rsa.pub
     -rw-r--r--   1 dream  staff   989  5 23 13:37 known_hosts
     ```

     控制台输入： cd ~/.ssh 和 open . 打开ssh key的安装目录

     若现有已经存在，删除id_rsa 和 id_rsa.pub

   - 配置本地git的user name 和 email

> git config --global user.name "xjshuang"
>
> git config --global user.email "shuangxingji867@pingan.com.cn"
>
> git config --global gui.encoding utf-8 (可选，避免git的ui工具乱码)
>
> git config --global core.quotepath off（可选，避免git status显示中文文件名乱码）

配置完之后，可以使用 *git config --global --list* 查看配置状态

```
dreamtekiMacBook-Pro:.ssh dream$ git config --global  --list
user.name=xjshuang
user.email=shuangxingji867@pingan.com.cn
core.excludesfile=/Users/dream/.gitignore_global
core.quotepath=off
difftool.sourcetree.cmd=opendiff "$LOCAL" "$REMOTE"
difftool.sourcetree.path=
mergetool.sourcetree.cmd=/Applications/SourceTree.app/Contents/Resources/opendif
f-w.sh "$LOCAL" "$REMOTE" -ancestor "$BASE" -merge "$MERGED"
mergetool.sourcetree.trustexitcode=true
gui.encoding=utf-8
dreamtekiMacBook-Pro:.ssh dream$
```

- 生成ssh key密钥

> ssh-keygen -t rsa -C "shuangxingji867@pingan.com.cn"

一路enter，最终可以看到

```
dreamtekiMacBook-Pro:.ssh dream$ ssh-keygen -t rsa -C "shuangxingji867@pingan.co
m.cn"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/dream/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/dream/.ssh/id_rsa.
Your public key has been saved in /Users/dream/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:k9Lj2FQg5dNL8uO30cdjcynmwyXAHgkelHRc7ddx0IE shuangxingji867@pingan.com.cn
The key's randomart image is:
+---[RSA 2048]----+
|      ..=oo...++.|
|       o *.. E oo|
|        = B . . +|
|       . O *   .o|
|      . S = o   .|
|       * + o o o.|
|      . o . +o+=+|
|           .o=o.+|
|            ...  |
+----[SHA256]-----+
dreamtekiMacBook-Pro:.ssh dream$
```

- 最终在ssh key的安装目录下面，就可以看到id*rsa和id*rsa.pub

这个id_rsa.pub文件里面的内容，就是你的唯一标志，加入到git的仓库权限，就能进行代码库的操作了

3. 公司git仓库

    - 登录公司git仓库，注册（需用公司邮箱注册）

      git仓库地址(http://git.yqb.pub/h5)

      > 新建好的git帐号，发邮件给配管，加入到git的h5仓库组

发布&分支管理规范

```
目的:
为了规范化我们目前分支管理以及发布，我们需要做到
    i. 角色权限控制
    ii. 分支拉取和命名规范
    iii. 封板发布
    iv. 代码合并

1. 角色介绍
    每一个github仓库，有四个角色，权限从高到低
    i. owner: admin账号，配管控制，建立新仓库需要
    ii. master: 仓库的管理者，可以操作合并master代码，可以分配其它开发者权限，管理员
    iii. developer: 开发者，可以提交推送除了master,release之外的其他分支
    iv. guest: 只读权限，不具备提交代码功能

    !!master权限，只有分配给项目负责人，项目负责人控制release代码进行代码的封板和解封操作

2. 新仓库建立
    目前前端h5的代码总仓库地址为
    http://git.yqb.pub/h5
    新项目，需要让owner(配管)，在h5的代码仓库下，新建项目名xxx，并分配master权限

3. 代码提交规范
        git commit -m "{$行为} {$霸道id} {$耗时} {$信息}"
        $行为: do/ref/finish
        $霸道id: #8888
        $耗时: @1h/@30m
        $信息: fix xxx bug

    Exp: git commit -m "ref #76732 @4h 商城817版本苏宁易购库存需求"

4. 分支命名规范
    目前jekins已经将打包建构进行优化，stg3和stg5环境只能打包建构dev分支代码，stg2环境只能打包建构release代码，所以分支命名按以下规范进行
    常规版本:
    {ProjectName}_{发布日期}_dev
    release
    紧急生产故障版本:
```

{ProjectName}_{紧急生产故障日期}_hotfix_dev
{ProjectName}_{紧急生产故障日期}_hotfix_release

5. 开发和发布流程

分支含义:

master: 代码发布分支，已发布的版本需要打上tag标签备份，只有master以上权限能操作

dev: 代码开发分支，所有developer以上权限都能操作

release: 代码发布分支，封板后，只有master权限能合并

!!pdp系统地址(内网):www.pdp.pinganfu.net

!!所有已发布的版本，尽量在master分支上打上tag标签，tag标签为时间戳(version_20170622)

打上tag标签，是便于以后，拉分支可以拉出各个版本的分支

master管理员分支管理操作，基于master拉出版本release发布分支，在基于release分支，拉出对应版本的dev开发分支，设置release分支的提交权限

!!gitlab上切记设置分支权限

!!依赖于后台的版本，需要走pdp系统，进行提测和发布的常规流程

下面以商城为例，介绍一次发布流程

A. 不并行开发的流程:

商城0720已发布，下一个版本为0817，

* 0720发布上线后，将mall_20170720_dev的代码分支，合并回release，并在master打上tag标签(version_20170720)
* 基于release分支拉出开发分支mall_20170817_dev,在dev分支上进行开发
* 在gitlab上配置权限，只有release分支只有master角色能操作
* 0731提测日，在pdp系统，发起提测申请流程
* 0811封板，管理员将mall_20150817_dev合并回release分支，并进行封板操作
* 同时，在pdp系统，提交0817的发布单，在发布系统收集过程中，标明系统依赖和发布以来顺序
* 如果代码需要解封，解封需要走原有解封流程，邮件通知
* developer在dev提交的代码，可以通过gitlab上发起合并请求等操作
* 管理员在gitlab上处理合并操作，gitlab上可以回滚，解决冲突，提交等一系列操作
* 0817基于release发布完成之后，将代码合并回master分支
* master打上tag标签version_20170817

B. 并行开发流程（pdp提测流程同上）:

商城0511已发布，下一版本为0528和0608

* 基于release分支拉出mall_20170528_dev和mall_20170608_dev
* 常规流程，开发者mall_20170528_dev开发完成封板，release发布部署
* 将release代码合并回master,打上tag标签version_20170528
* 将release代码同步合并到mall_20170608_dev，完成代码同步
* 0608版本封板后，代码合并回release，发布部署后，将代码合并回master分支，并打上tag标签version_20170608
*

C. 紧急bug版本（不限制封板等操作，achain发布流程）:

商城0511已发布，下一版本为0528，紧急需要修复一个线上bug在0518

* 基于version_20170511的tag标签，拉出fix分支，mall_20170518_hotfix_dev

```
        * 开发基于hotfix分支操作，提测
        * 基于mall_20170518_hotfix_dev拉出mall_20170518_hotfix_release分支进行回归发布
        * 发布完成之后，将mall_20170518_hotfix_release分支合并回master，并打上标签version
_hotfix_20170518
        * 在将master的内容，merge到现有版本开发分支，release
        * 在将release的代码同步到mall_20170528_dev，完成全量同步
```

Jekins建构

```
各环境打包目录：
http://172.20.1.131:8089/
账号密码：h5-test/h5-test
各环境发布目录（内网云桌面打开）：
http://10.59.2.209:8082/view/h5/
账号密码：h5-test/h5-test

外网选到对应部署环境，可以看到git上所有的分支，选择对应的分支生成zip包
内网，选择对应环境，构建，将zip包发布到对应的stg环境
```

Git 常用指令

1. 创建克隆远程库
    git clone git@git.yqb.pub:h5/mall.git

2. 切换分支
    git checkout -b branch(相当于branch后，在checkout)
3. 合并代码
    切换到目的分支，执行git merge的操作，有冲突后，解决冲突
4.
    git pull origin master(分支名)
    git commit -m "提交注释"
    git push origin master(分支名)
    每一次push之前，要确保先要pull一次

5. 代码缓存
    git stash：备份当前的工作区的内容，从最近的一次提交中读取相关内容，让工作区保证和上次提交的内容一致。同时，将当前的工作区内容保存到Git栈中
    git stash list：显示Git栈内的所有备份，可以利用这个列表来决定从那个地方恢复。
    git stash clear：清空Git栈。
    git stash pop：从Git栈中读取最近一次保存的内容，恢复工作区的相关内容。由于可能存在多个Stash的内容，所以用栈来管理，pop会从最近的一个stash中读取内容并恢复。

    git stash save "name"
    git stash pop --index stash@{0}

6. 代码回滚