**Python1-1**

**Python的特点**

**高级：有高级的数据结构，缩短开发时间与代码量**

**面向对象：为数据和逻辑相分离的结构化和过程化编程添加了新的活力**

**可升级：提供了基本的开发模块，可以在它上边开发软件，实现代码的重用**

**可扩展：通过将其分离为多个文件或模块加以组织管理**

**可移植性：python是用C写的，又由于C的可移植性 使得python 可以运行在任何带有ANSI C编译器的平台上**

**易读:没有其他语言通常用来访问变量、定义代码块和进行模式匹配的命令式符号**

**内存管理器：内存管理是有python解释器负责的**

**安装python3**

**# yum install -y gcc gcc-c++ zlib-devel openssl-devel readline-devel libffi-devel sqlite-devel tcl-devel tk-devel**

**# tar xzf Python-3.6.4.tar.gz**

**# cd Python-3.6.4**

**# ./configure --prefix=/usr/local**

**# make && make install**

**设置pycharm**

**支持的功能有:**

**– 调试、语法高亮**

**– Project管理、代码跳转**

**– 智能提示、自动完成**

**– 单元测试、版本控制**

**• 下载地址:https://www.jetbrains.com/pycharm/download**

**Git简介 实现代码管理的工具**

**Git是一个开源的分布式版本控制系统,用于敏捷高效地处理任何或小或大的项目。**

**• Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。**

**• Git 与常用的版本控制工具 CVS, Subversion 等不同,它采用了分布式版本库的方式,不必服务器端软件支持。**

**# yum install -y git**

**配置用户的基本信息**

**# git config --global user.name "qian"**

**# git config --global user.email "zhangzg@tedu.cn"**

**# git config --global core.editor vim编写日志默认使用的编辑器是vim**

**查看**

**# git config --list**

**user.name=qian**

**user.email=qian@tedu.cn**

**core.editor=vim**

**# cat ~/.gitconfig**

**[user]**

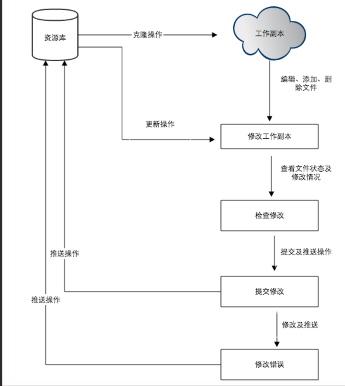
**name = qian**

**email = qian@tedu.cn**

**[core]**

**editor = vim**

**Git工作流程**

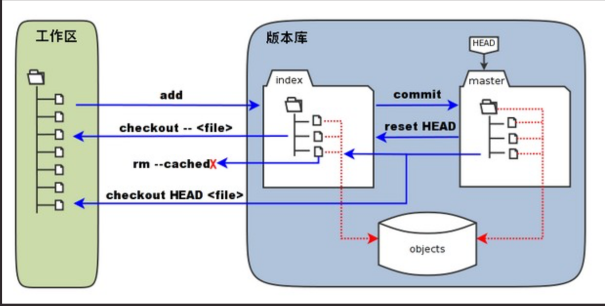
****

**工作区、暂存区和版本库**

**• 工作区:就是你在电脑里能看到的目录 编写代码的目录 工作区的文件不一定会保存到git里**

**• 暂存区:英文叫stage, 或index。一般存放在 ".git目录下" 下的index文件(.git/index)中,所以我们把暂存区有时也叫作索引(index) 工作区和版本库的缓冲地带 工作区的文件先提交到暂存区 确定可以提交到版本库 不确定可以删除**

**• 版本库:工作区有一个隐藏目录.git,这个不算工作区,而是Git的版本库**

****

**创建仓库**

**[root@room9pc01 ~]# git init devops**

**初始化空的 Git 版本库于 /root/devops/.git/**

**查看git状态**

**[root@room9pc01 devops]# git status**

**# 位于分支 master**

**#**

**# 初始提交**

**#**

**无文件要提交（创建/拷贝文件并使用 "git add" 建立跟踪）**

**在工作区编写程序**

**[root@room9pc01 devops]# cat hello.py**

**print('hello world!')**

**将当前目录中所有的文件放到暂存区**

**[root@room9pc01 devops]# git add .**

**[root@room9pc01 devops]# git status 详细信息**

**# 位于分支 master**

**# 初始提交**

**# 要提交的变更：**

**# （使用 "git rm --cached <file>..." 撤出暂存区）**

**# 新文件： hello.py**

**[root@room9pc01 devops]# git status -s 查看概要信息**

**A hello.py**

**将暂存区的文件提交到版本库**

**[root@room9pc01 devops]# git commit -m "project init"**

**[master（根提交） 74e4f83] project init**

**1 file changed, 1 insertion(+)**

**create mode 100644 hello.py**

**[root@room9pc01 devops]# git status**

**# 位于分支 master**

**无文件要提交，干净的工作区**

**继续添加代码**

**[root@room9pc01 devops]# echo "a=10" >> hello.py**

**提交工作i内容到版本库**

**[root@room9pc01 devops]# git commit -m "add new"**

**查看提交日志**

**[root@room9pc01 devops]# git log**

**查版本库中的文件**

**[root@room9pc01 devops]# git ls-files**

**hello.py**

**hha.py**

**删除文件**

**[root@room9pc01 devops]# git rm hha.py 确认删除需要git add . (commit)**

**rm 'hha.py'**

**撤销删除**

**[root@room9pc01 devops]# git reset HEAD hha.py**

**重置后撤出暂存区的变更：**

**D hha.py**

**[root@room9pc01 devops]# git checkout -- hha.py 克隆**

**[root@room9pc01 devops]# ls**

**hello.py hha.py 多看git status 有提示**

**配置GITLAB服务器 实现代码统一管理**

**pycharm汉化**

**1、百度搜索 pycharm2017.3汉化包**

**2、在百度网盘中下载resource\_cn.jar**

**3、把resources\_cn.jar拷贝到/root/bin/pycharm2017/lib/中**

**4、如果经过第3步，仍然是英文界面，可以将lib目录中的resources\_en.jar先剪切到别处**

**5、启动pycharm，将报错**

**6、关闭pycharm后，把resources\_en.jar再拷贝进去**

**git**

**1、它是实现代码管理的工具**

**2、安装**

**[root@node3 ~]# yum install -y git**

**3、配置用户基本信息**

**[root@node3 ~]# git config --global user.name zzg**

**[root@node3 ~]# git config --global user.email zhangzg@tedu.cn**

**# 如果需要编写日志，默认使用的编辑器是vim**

**[root@node3 ~]# git config --global core.editor vim**

**4、查看**

**[root@node3 ~]# git config --list**

**[root@node3 ~]# cat ~/.gitconfig**

**5、git的三个区域**

**工作区：指的就是我们编写代码的目录，工作区中的文件不一定会保存到git中**

**暂存区：它是工作区和版本库之间的缓冲地带。工作区中的文件先提交到暂存区，如果确定加入**

**git管理，可以提交到版本库；如果后悔了，可以将文件从暂存区中删除**

**版本库：在工作区有一个.git目录，它是程序员保存代码、交给git管理的区域**

**使用git管理代码**

**1、创建目录（项目的根目录）**

**[root@node3 ~]# git init devops**

**[root@node3 ~]# cd devops/**

**2、查看git状态**

**[root@node3 devops]# git status**

**3、在工作区编写程序**

**[root@node3 devops]# vim hello.py**

**print('Hello World!')**

**[root@node3 devops]# git status**

**4、将当前目录中所有的文件放到暂存区**

**[root@node3 devops]# git add .**

**[root@node3 devops]# git status**

**[root@node3 devops]# git status -s # 查看概要信息**

**5、将暂存区中的文件提交到版本库**

**[root@node3 devops]# git commit -m "project init"**

**[root@node3 devops]# git status**

**6、继续编写代码**

**[root@node3 devops]# echo "a = 10" >> hello.py**

**[root@node3 devops]# echo 'print("yes")' > new.py**

**7、将现有工作区内容提交到版本库**

**[root@node3 devops]# git add .**

**[root@node3 devops]# git commit -m "add new file"**

**8、查看提交日志**

**[root@node3 devops]# git log**

**9、查看版本库中的文件**

**[root@node3 devops]# git ls-files**

**10、删除文件**

**[root@node3 devops]# git rm new.py**

**11、撤销删除**

**[root@node3 devops]# git status**

**[root@node3 devops]# git reset HEAD new.py**

**[root@node3 devops]# git checkout -- new.py**

**配置GITLAB服务器，实现代码统一管理**

**1、创建虚拟机，内存至少4G，配置好yum、ip地址，安装docker并启动**

**2、将gitlab镜像导入到docker中**

**[root@node1 images]# docker load < gitlab\_zh.tar**

**3、将虚拟机的ssh端口改为2222**

**[root@node3 devops]# vim /etc/ssh/sshd\_config**

**Port 2222**

**[root@node3 devops]# systemctl restart sshd**

**4、使用gitlab镜像启动容器**

**[root@node1 ~]# docker run -d -h gitlab --name gitlab**

**-p 443:443 -p 80:80 -p 22:22 --restart always**

**-v /srv/gitlab/config:/etc/gitlab -v /srv/gitlab/logs:/var/log/gitlab**

**-v /srv/gitlab/data:/var/opt/gitlab gitlab\_zh:latest**

**-h指定容器的主机名，--name指定执行dock ps时看到的容器名**

**5、在物理机上通过浏览器访问gitlab**

**因为硬件限制，运行起容器后，gitlab在一段时间内无法访问，需要多等几分钟。**

**可以在虚拟机上执行docker ps查看状态，当状态是healthy的时候，就可以正常访问了。**

**6、gitlab中重要的概念**

**群组group：可以为每个团队创建一个group**

**项目project：每个团队都可能开发不只一个项目**

**成员member：每个团队成员，需要有相应的用户名登陆到git**

**gitlab用户免密登陆**

**1、生成密钥对**

**# ssh-keygen -t rsa -C "zhangzg@tedu.cn" -b 4096**

**2、复制~/.ssh/id\_rsa.pub中的公钥**

**3、粘贴到gitlab个人用户的ssh密钥中**

**用户将本地代码上传到gitlab服务器**

**1、进入本地git项目目录**

**[root@node3 ~]# cd devops/**

**2、将远程名称origin改名**

**[root@node3 devops]# git remote rename origin old-origin**

**如果出现以下错误，可以忽略**

**error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'**

**3、将本地git名称origin添加到远程**

**[root@node3 devops]# git remote add origin git@192.168.4.1:devops/core\_py.git**

**4、推送代码到gitlab服务器**

**[root@node3 devops]# git push -u origin --all**

**[root@node3 devops]# git push -u origin --tags**

**如果出现permission denied，需要执行以下操作：**

**[root@node3 devops]# ssh-agent**

**将屏幕输出内容执行后，再执行**

**[root@node3 devops]# ssh-add**

**5、在gitlab上访问项目，可以看到本地代码已推送到服务器**

**如果出现@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@远程主机身份改变，**

**则把~/.ssh/known\_hosts删除再试**

**pycharm配置**

**file -> settings -> project -> project interpreter -> 右窗格点击右上角**

**的齿轮选择add local -> 在弹出的窗口，选择下面的Existing enviroment，把**

**Make available to all projects勾选，再点击右侧的...跳出窗口输入**

**/usr/local/bin/python3。然后确定返回。**

**运行python代码**

**1、在pycharm的文件中右击选择“运行”**

**2、也可以在终端中：python3 xxxx.py**

**python中字符串必须有引号，单双引号表示相同的含义。如果不加引号，python会把它当**

**成一个名称，如果该名称没有定义将会报错。**

**username = input('用户名：')**

**print('欢迎你：', username)**

**print('欢迎你：' + username)**

**# print('Hello World!')**

**# if 3 > 0:**

**# print('yes')**

**# print('ok')**

**# print('abc')**

**# a = 3**

**# b = 4**

**# x = 3; y = 4**

**# print('Hello World')**

**# # print语句各项之间默认用空格分隔**

**# print('Hello', 'World')**

**# # print语句各项之间也可以自定义分隔符**

**# print('Hello', 'world', sep='\*\*\*')**

**# # 字符串可以使用+进行拼接**

**# print('Hello' + 'world')**

**# # print语句默认最后会加入一个回车\n，可以用end来自定义结束**

**# print('Hello World', end='AAA')**

**a = input('number: ')**

**print(a)**

**# b = a + 10 # input读入的数据都是字符型，字符不能和数字相加**

**print(int(a) + 10) # 将a转成整数后和10相加，再打印出出结果**

**print(a + str(10)) # 将10转成字符串后和a拼接，再打印出出结果**

**合法的变量名，有三个要求：**

**1、首字符必须是字母或下划线**

**2、其他字符可以是字母、数字或下划线**

**3、区分大小写**

**python中变量使用之前必须赋值**

**n = 10**

**n = n + 1**

**赋值语句自右向左运算。首先取出n的值10，再计算n+1的值得到11，最后把11赋值给n**

**可以简写成n += 1**

**5 / 2 值为2.5**

**5 // 2 值为2**

**5 % 2 求余，也被称作模运算，结果为1**

**2 \*\* 3 2的3次方，乘方、幂运算**

**10 < 20 < 30 python支持连续比较**

**10 < 20 > 15 相当于10 < 20 and 20 > 15**

**https://www.jianshu.com/ -> 搜索“python百例”**

**做“1－5”和“120-打造vim为python IDE”**