**python概述**

python简介

python起源

吉多\*范罗苏姆于1989年底始创了python

1991年初，python发布了第一个公开发行版

为了更好的完成荷兰的CWI（国家数学和计算机科学研究院）的一个研究项目而创建

python版本

python2.x

目前所有系统默认安装的版本

python3.x

2009年2月13日发布

在语法和功能上有较大调整

python的发展趋势

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，可以在上面开发软件，实现代码的重用

可扩展：通过将其分离为多各稳健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可以运行在人和带有ANSIC编译器的平台上

易学：python关键字少、结构简单、语法清晰

易读：没有其他语言通畅用来访问变量、定义代码块喝进行模式匹配的命令式符号

内存管理器：内存管理是由python解释器负责的

**环境准备**

安装与配置

获取python源码

官方站点

http://www.python.org

选择正确的系统

选择正确的版本

安装python3

安装依赖包

yum install -y gcc gcc-c++ zlib-devel openssl-devel readline-devel

libffi-devel sqlite-devel tcl-devel tk-devel

安装python3

tar -xf python-3.6.4.tar.gz

cd python-3.6.4

./configure --prefix=/usr/local

make && make install

设置pycharm

pycharm是由jetbrains打造的一款python ide

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/>

分为收费的专业版和免费的社区版

**使用git**

本地操作

git简介

git是一个开源的分布式版本控制系统，用于敏捷高效地出力任何或大或小的项目

git是linux 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、svn等不不同，它采用了分布式版本库的方式，不必服务器端软件支持

安装及配置

yum -y install git

git config --global user.name "Mr.Zhang"

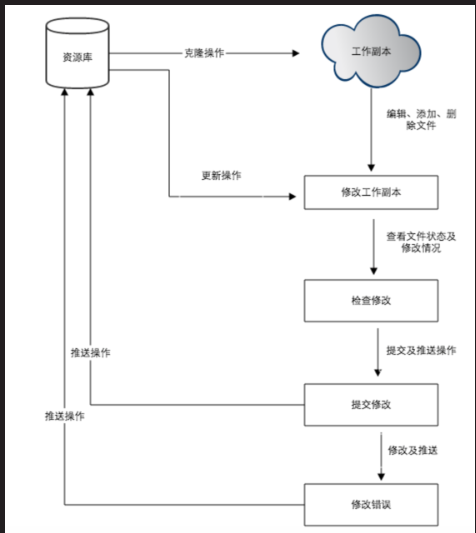
git config --global user.email ["zhangzg@tedu.cn"](mailto:\"zhangzg@tedu.cn\")

git config --global core.editor vim

git config --list

cat ~/.gitconfig

git工作流程

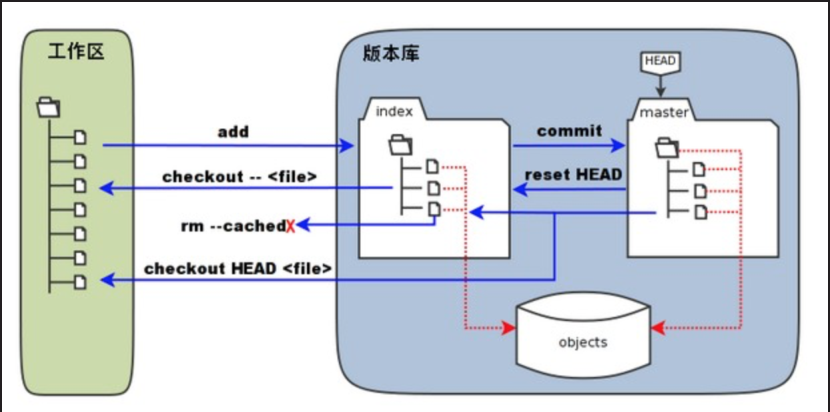


工作区、暂存区和版本库

工作区：就是电脑里能看到的目录

暂存区：英文叫sdtage，或index。一般存放在”.git目录”下的index文件（.git/index）中，所以把暂存区也叫做索引

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

git是用git init命令来初始化一个git仓库，git的很多命令都需要在git的仓库中运行，所以git init是使用git的第一个命令

mkdir devops

cd devops

git init或者git init devops

添加文件到暂存区

添加指定文件

echo 'print("hello world!")' > hello.py

git add hello.py

git status

添加所有文件

cp hello.py welcome.py

git add .

git status -s

确认至仓库

提交之前务必先设置用户信息

git commit -m “add”

git status

添加追踪文件并提交到版本库

echo 'print("done.")' >> hello.py

git commit -am "向hello.py添加新行"

删除跟踪的文件

要从git中移除某个文件，就必须要从已跟踪文件清单中移除，然后提交

git ls-files //查看版本库中文件

git rm welcome.py

git commit -m '删除welcome.py'

使用远程服务器

搭建本地gitlab服务器

导入中文版gitlab镜像

将物理主机ssh端口改为2022后，启动容器

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

初始化gitlab服务器

密码需大于8位



用户名默认为root



添加gitlab项目

创建群组group

使用群组管理项目和人员是非常好的方式

创建项目project

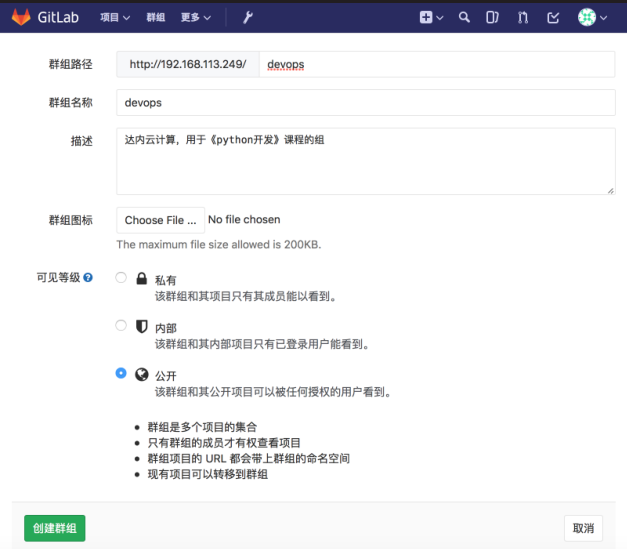
存放代码的地方，里面还包含问题列表、危机温度一级其它一些gitlab功能

创建成员member

添加团队成员或其他人员到gitlab

创建群组





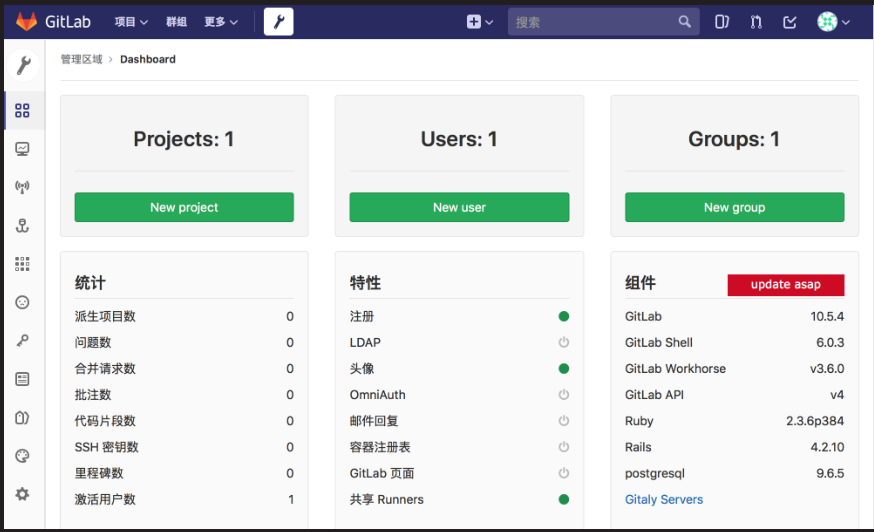
创建项目







创建用户



创建用户后，再次编辑可设置密码



root用户将新用户加入组中，并设置新用户为“主程序员”



新用户初次登录，需设置自己的密码



用户管理

将本地生成的公钥上传至服务器



将本地仓库推送至服务器

git remote rename origin old-origin

git remote add origin

git push -u origin --all

添加新文件

echo '# this is a test' > hi.py

git add hi.py

git commit hi.py -m '新的测试'

git push origin master

下载代码到本地

git clone [git@192.168.113.249:devops/core\_py.git](mailto:git@192.168.113.249:devops/core_py.git)

ls -a core\_py/

更新代码到本地

git pull

**python起步**

python运行方式

交互式解释器

进入交互式解释器

python3

退出交互式解释器

exit或ctrl+d

文件形式

明确指定解释器

python3 hello.py

赋予python文件可执行权限

chmod +x hello.py

./hello.py

python语法结构

语句块缩进

python代码块通过缩进对齐表达代码缩进而不是是用大括号

缩进表达一个语句属于哪个代码块

缩进风格

1或2：可能不够，很难确定代码语句属于哪个块

8至10：可能太多，如果代码内嵌的层次太多，就会使得代码很难阅读

4个空格：非常流行， 范·罗萨姆支持的风格

缩进相同的一组语句构成一个代码块，称为代码组

首行以关键字开始，以冒号：结束，该行至后的以行或多行代码构成代码组

如果代码组只有一行，可以将其直接写在冒号后面，但是这样的写法可读性差，不推荐

注释及续行