# 1、pbr简介

pbr是一个python软件包，通过执行命令pip install pbr来安装，通过pip show pbr命令的结果可以看到pbr这个软件包是由openstack开发维护的

[root@kolla ~]# pip show pbr

Name: pbr

Version: 3.1.1

Summary: Python Build Reasonableness

Home-page: http://docs.openstack.org/developer/pbr/

Author: OpenStack

Author-email: openstack-dev@lists.openstack.org

License: UNKNOWN

Location: /usr/lib/python2.7/site-packages

Requires:

openstack官方对pbr的详细介绍的主页：<https://docs.openstack.org/pbr/latest/>

pbr的github地址：<https://github.com/openstack-dev/pbr>

我记得以前碰到的一个问题：在使用pip install .的方式安装从git上克隆下来的OpenstackClient时，一直卡在Processing .....中，解决方法就是安装pbr，可见pbr是安装OpenstackClient所必须的一个依赖

pbr是Python Build Reasonableness的缩写【意为python下合理编译】，pbr是一个setuptools的扩展工具、辅助工具，用来配合setuptools完成安装/打包工作，它会通过一个setup hook来读取和过滤setup.cfg配置文件中的数据，将解析出的数据作为调用时的参数提供给setup.py，pbr具体可以做的事情有：

**Version**: 基于git revisions和tags来管理版本号

**AUTHORS**: 从git log生成AUTHORS

**ChangeLog**: 从git log生成ChangeLog

**Manifest**: 从git files和一些标准文件中生成有意义的manifest

**Sphinx Autodoc**: 为整个模块生成autodoc stub files

**Requirements**: 将软件包的依赖信息存入requirements文件

**long\_description**: 使用README文件作为long\_description

**Smart find\_packages**: 在root package下智能寻找软件包

setup.py通用内容：

setuptools.setup(setup\_requires=['pbr>=1.8'],pbr=True)

openstack项目客户端的源码包中的setup.py中都有这么一句，用来指定使用pbr来辅助setuptool来完成安装或打包的工作，这里的setup\_requires参数意思是setup函数在执行之前需要依赖的包的列表，那么OpenStack社区为什么要开发pbr呢？因为setuptools库使用起来还是有点麻烦，参数太多，而且直接通过指定setup函数的参数的方法实在太不方便了。pbr就是为了方便而生的

# 2、pbr对版本的管理

pbr通过两种方式管理软件包的版本： postversioning和preversioning，可以通过在setup.cfg配置文件的metadata区域中设置version字段来启用preversioning管理方式，如果没有这样设置，则默认的管理方式是postversioning，两种管理方式下软件包的版本号都是从git的history中推理计算得到的，pbr使用的版本号标准是Linux/Python Compatible Semantic Versioning 3.0.0，该标准中版本号的结构是：MAJOR.MINOR.PATCH，三部分的含义如下

MAJOR version when you make incompatible API changes,

MINOR version when you add functionality in a backwards-compatible manner,

PATCH version when you make backwards-compatible bug fixes.

开发版本号的形式：MAJOR.MINOR.PATCH.devN【N=从最近的git tag开始的提交数量】

**pbr推导git源码包中版本号的规则：**

摘自<http://www.infoq.com/cn/articles/OpenStack-demo-packagemanagement>【有问题】

1. 如果有在setup.cfg中设置version字段的值，即使用的是preversioning管理方式，则

1.1、如果存在一个与该version对应的tag，则这个version就是最终版本号

1.2、如果不存在一个与该version对应的tag，如果version大于2中推导出的版本号，则

使用该版本号作为最终版本号，否则抛出异常

1. 如果未在setup.cfg中设置version字段的值，即使用的是postversioning管理方式，则

pbr会先找到最近的一个tag，并通过为其MINOR值加1来得到一个临时版本号，然后从这个最近的tag开始遍历所有的git commit历史，并搜索每个commit message中的

Sem-Ver字段，根据其值的不同做出不同的行为：

2.1、若Sem-Ver值为api-break，则增加临时版本号中的MAJOR值

2.2、若Sem-Ver值为feature或deprecation，则增加临时版本号中MINOR值

2.3、若Sem-Ver值为bugfix或不存在Sem-Ver字段，则增加临时版本号中PATCH值

2.4、若Sem-Ver值不在上述之列，则会发出警告信息

经过以上步骤得到的便是最终版本号

注：最终版本号指的就是软件包的版本号，且最终版本号对应的tag必须是有签名的

**pbr推导git源码包中版本号的规则：**

译自<https://docs.openstack.org/pbr/latest/user/features.html>

1、检查最近的commit的标签

1.1、如果当前被check out出来的commit已经被打了标签，那么就把该标签作为版本号

1.2、如果当前被check out出来的commit没有被打标签，那么会选择最近的一个标签，然后按照下面的规则来增加它的版本号，首先从最近的那个tag开始遍历所有的commit，在每一个commit中寻找Sem-Ver字段

1.2.1、若Sem-Ver值为api-break，则版本号中的MAJOR值加1

1.2.2、若Sem-Ver值为feature或deprecation，则版本号中MINOR值加1

1.2.3、若Sem-Ver值为bugfix或不存在Sem-Ver字段，则版本号中PATCH值加1

1.2.4、若Sem-Ver值不在上述之列，则会发出警告信息

2、检查当前使用的版本号管理方式

如果当前使用的是postversioning管理方式，则用上面得到的结果作为版本号

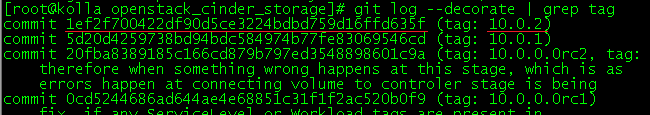
如果当前使用的是preversioning管理方式，则如果在setup.cfg配置文件中设置的version字段的值大于通过上述方法得到的版本号，那么就使用setup.cfg中的版本号作为最终版本号，否则报错

**对源码包openstack\_cinder\_storage中版本号的解释：**

通过命令git log --pretty=oneline -n 1可知，当前被checkout出来的commit未打标签



通过命令git log --decorate | grep tag可以看到最近的一个标签是10.0.2，所以临时采用此标签作为版本号，其对应的commit id为1ef2f700422df90d5ce3224bdbd759d16ffd635f



通过命令git log -n 44 | grep "Sem-Ver" 寻找最近的tag后的commit中的Sem-Ver字段

结果为空，根据规则，版本号中的PATCH的值要加1，所以得到的版本号是10.0.3

通过命令git log --pretty=oneline -n 44可以看到在最近的一个标签之后有43个commit



所以最后得到的开发版本号就是10.0.3.dev43



**试错**

下载的python-openstackclient源码包的版本是3.8.2，我故意在setup.cfg配置文件的metadata区域中设置version字段的值为3.8.1，然后pip install .来安装，结果报错：

ERROR:root:Error parsing

Traceback (most recent call last):

File "/usr/lib/python2.7/site-packages/pbr/core.py", line 111, in pbr

attrs = util.cfg\_to\_args(path, dist.script\_args)

File "/usr/lib/python2.7/site-packages/pbr/util.py", line 249, in cfg\_to\_args

pbr.hooks.setup\_hook(config)

File "/usr/lib/python2.7/site-packages/pbr/hooks/\_\_init\_\_.py", line 25, in setup\_hook

metadata\_config.run()

File "/usr/lib/python2.7/site-packages/pbr/hooks/base.py", line 27, in run

self.hook()

File "/usr/lib/python2.7/site-packages/pbr/hooks/metadata.py", line 26, in hook

self.config['name'], self.config.get('version', None))

File "/usr/lib/python2.7/site-packages/pbr/packaging.py", line 740, in get\_version

version = \_get\_version\_from\_git(pre\_version)

File "/usr/lib/python2.7/site-packages/pbr/packaging.py", line 680, in \_get\_version\_from\_git

result = \_get\_version\_from\_git\_target(git\_dir, target\_version)

File "/usr/lib/python2.7/site-packages/pbr/packaging.py", line 642, in \_get\_version\_from\_git\_target

dict(new=new\_version, target=target\_version))

ValueError: git history requires a target version of pbr.version.SemanticVersion(3.8.2), but target version is pbr.version.SemanticVersion(3.8.1)

error in setup command: Error parsing /tmp/pip-Ut19iq-build/setup.cfg: ValueError: git history requires a target version of pbr.version.SemanticVersion(3.8.2), but target version is pbr.version.SemanticVersion(3.8.1)

可见，在安装python-openstackclient时，pbr会先通过setup\_hook钩子来获取setup.cfg中的配置信息，发现其中有version字段，于是采用preversioning版本管理方式