


```
提取码: ejfb
```

解压到指定目录/opt 下，生成/opt/gbase8s-odbc-driver 目录

```
[root@localhost test]# ll
总用量 35188
-rw-r--r--. 1 root root 36029237 3月 11 20:52 GBase8s_3.0.0_1-Linux64-ODBC-Driver.tar.gz
[root@localhost test]# tar -zxf GBase8s_3.0.0_1-Linux64-ODBC-Driver.tar.gz -C /opt/
[root@localhost test]# cd /opt/
[root@localhost opt]# ll
总用量 4
drwxr-xr-x. 20 gbasedbt gbasedbt 4096 3月 10 15:14 gbase
drwxrwxr-x. 9 1001 1003 88 12月 13 2020 gbase8s-odbc-driver
drwxr-xr-x. 2 root root 6 10月 31 2018 rh
```

创建必须的环境变量，并使环境生效

```
# 注意：不要设置 ODBCINI 环境变量，如果有，需要 py 里 os.unsetenv("ODBCINI")
export GBASEBTDIR=/opt/gbase8s-odbc-driver
export CSDK_HOME=/opt/gbase8s-odbc-driver
export PATH=$GBASEBTDIR/bin:$PATH
export LD_LIBRARY_PATH=$GBASEBTDIR/lib:$GBASEBTDIR/lib/cli:$GBASEBTDIR/lib/esql:$LD_LIBRARY_PATH
```

创建 sqlhosts 配置文件

```
[root@localhost test]# vi /opt/gbase8s-odbc-driver/etc/sqlhosts
[root@localhost test]# more /opt/gbase8s-odbc-driver/etc/sqlhosts
gbase01 onsoctcp a02.gbasedbt.com 9088
```

3, 安装 sqlalchemy-gbasedbt

3.1, 在线安装 sqlalchemy-gbasedbt

确认 python3、python3-devel 和 gcc 均已经安装，CSDK 也已经安装以及环境变量已经配置的情况下，可直连网络的情况下，可使用 pip3 install sqlalchemy-gbasedbt 直接安装

```
[root@localhost test]# pip3 install sqlalchemy-gbasedbt
Collecting sqlalchemy-gbasedbt
  Using cached sqlalchemy_gbasedbt-0.2.4-py3-none-any.whl (10 kB)
Collecting DbtPy
  Using cached DbtPy-3.0.5.4.tar.gz (162 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: sqlalchemy in /usr/local/lib64/python3.6/site-packages (from sqlalchemy-gbasedbt) (1.4.46)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib64/python3.6/site-packages (from
```

```
sqlalchemy->sqlalchemy-gbasetbt) (2.0.2)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.6/site-packages (from
sqlalchemy->sqlalchemy-gbasetbt) (4.8.3)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.6/site-packages (from
importlib-metadata->sqlalchemy->sqlalchemy-gbasetbt) (4.1.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/site-packages (from importlib-
metadata->sqlalchemy->sqlalchemy-gbasetbt) (3.6.0)
Using legacy 'setup.py install' for DbtPy, since package 'wheel' is not installed.
Installing collected packages: DbtPy, sqlalchemy-gbasetbt
  Running setup.py install for DbtPy ... done
Successfully installed DbtPy-3.0.5.4 sqlalchemy-gbasetbt-0.2.4
```

将同时安装依赖包：sqlalchemy、greenlet、importlib-metadata、typing-extensions、zipp 和 DbtPy，安装后的 pip3 列表如下：

```
[root@localhost test]# pip3 list
Package            Version
-----
DbtPy              3.0.5.4
greenlet           2.0.2
importlib-metadata 4.8.3
pip                21.3.1
setuptools         39.2.0
SQLAlchemy         1.4.46
sqlalchemy-gbasetbt 0.2.4
typing_extensions 4.1.1
zipp               3.6.0
```

4， 编写测试 Demo， 执行测试

测试 demo 文件

```
#!/usr/bin/env python3
# Filename: testSqlalchemy_gbasetbt

from sqlalchemy import MetaData, Table, Column, String, create_engine
from sqlalchemy.dialects import registry
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base

registry.register("gbasetbt", "sqlalchemy_gbasetbt.dbtdb", "GBasetbtDialect")
```

```
import os
os.unsetenv(' ODBCINI' )

# 创建对象的基类:
Base = declarative_base()

# 定义 User 对象:
class User(Base):
    # 表的名字:
    __tablename__ = 'user'

    # 表的结构:
    id = Column(String(20), primary_key=True)
    name = Column(String(20))

# 初始化数据库连接:
# ConStr = ' gbasedbt://<username>:<password>@<host name>:<port number>/<databasename>;SERVER=<server name>'
# 注意: hostname:port 基本无意义, 直接 SERVER 匹配 GBASEDBTSQLHOSTS 配置文件中的 GBASEDBTSERVER
ConStr = ' gbasedbt://gbasedbt:GBase123@/testdb;SERVER=gbase01;DLOC=zh_CN.utf8;CLOC=zh_CN.utf8;DELIMIDENT=y'
engine = create_engine(ConStr)

# 删除对象
Base.metadata.drop_all(engine)

# 创建对象
Base.metadata.create_all(engine)

# 创建 DBSession 类型:
DBSession = sessionmaker(bind=engine)

# 创建 session 对象:
session = DBSession()

# 创建新 User 对象:
new_user = User(id='2', name='测试用户 a20512')
# 添加到 session:
session.add(new_user)
# 提交即保存到数据库:
session.commit()
# 关闭 session:
session.close()

# 创建 Session:
session = DBSession()
```

```

# 创建 Query 查询, filter 是 where 条件, 最后调用 one() 返回唯一行, 如果调用 all() 则返回所有行:
user = session.query(User).filter(User.id=='2').one_or_none()
# 打印类型和对象的 name 属性:
print('Print user')
if user :
    print(' type:', type(user))
    print(' name:', user.name)

# 执行原生 SQL 语句
sqlstr = "select tabid,tabname from systables where tabid < :id"
params = {"id" : 10}
result_proxy = session.execute(sqlstr, params)

result = result_proxy.fetchall()

print('\nPrint systbles')
for item in result:
    print(item)

# 关闭 Session:
session.close()

```

测试结果:

```

[root@localhost test]# ./testSqlalchemy_gbasedbt.py
Print user
type: <class '__main__.User'>
name: 测试用户 a20512

Print systbles
(1, 'systables')
(2, 'syscolumns')
(3, 'sysindices')
(4, 'systabauth')
(5, 'syscolauth')
(6, 'sysviews')
(7, 'sysusers')
(8, 'sysdepend')
(9, 'syssynonyms')

```