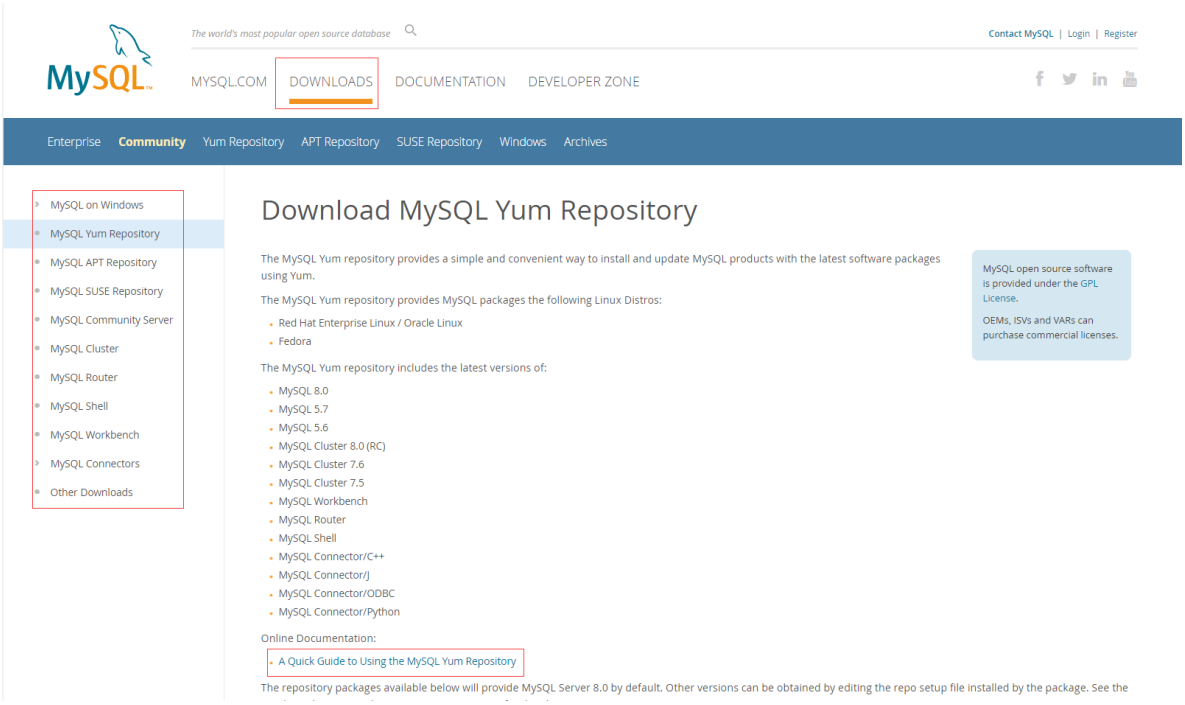
# 一、当前系统发行版

```
[root@i ~]# uname -r
3.10.0-514.26.2.el7.x86_64
```

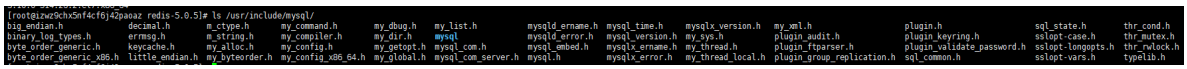
# 二、安装好MySQL

[官方地址]




# 三、安装MySQL的开发包

`yum install mysql-devel -y`




# 四、简单的加法运算

```
[root@aa mysql]# cat simple.cpp
#include <mysql.h>

extern "C" long long simple_add(UDF_INIT *initid, UDF_ARGS *args, char *is_null,
char *error){
    int a = *((long long *)args->args[0]);
    int b = *((long long *)args->args[1]);
    return a+b;
}

 extern "C" my_bool simple_add_init(UDF_INIT *initid, UDF_ARGS *args, char
*message){
    return 0;
}
```

# 五、编译命令

```
g++ -shared -fPIC -I /usr/include/mysql -o simple_add.so simple.cpp
```

| -shared | 表示编译和链接时使用的是全局共享的类库 |
| --- | --- |
| -fPIC | 编译输出位置无关的目标代码，适用于动态库 |
| -I | 指明包含头文件mysql.h所在位置 |

# 六、拷贝生成的动态库到MySQL的插件目录下

```
cp simple_add.so /usr/lib64/mysql/plugin/

# 插件目录查看方式
mysql> show variables like "%plugin%";
+---------------------------------+--------------------------+
| Variable_name                   | Value                    |
+---------------------------------+--------------------------+
| default_authentication_plugin | mysql_native_password    |
| plugin_dir                      | /usr/lib64/mysql/plugin/ |
+---------------------------------+--------------------------+
```

# 七、创建用户自定义函数

```
# 语法
CREATE [AGGREGATE] FUNCTION function_name
    RETURNS {STRING|INTEGER|REAL|DECIMAL}
    SONAME shared_library_name
# 创建
create function testtest returns string soname simple_add.so;
#使用
select simple_add(3,7);
#删除
drop function simple_add;
```

# 八、在MySQL中直接操作redis

## 1、安装依赖

`yum install boost boost-devel`

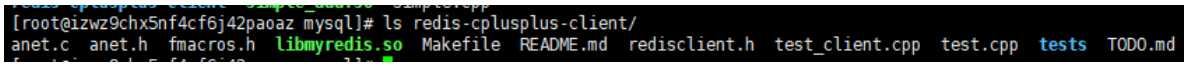
## 2、下载redis cpp client源码

下载地址

`git clone https://github.com/mrpi/redis-cplusplus-client`

> 使用时需要把redisclient.h、anet.h、fmacros.h、anet.c 这4个文件考到目录下，开始编写关于Redis的UDF。我们定义了redis_hset作为主函数，连接Redis并调用hset插入哈希表，redis_hset_init作为初始化，检查参数个数和类型。

## 3、编写C++操作类




```
[root@izwz9chx5nf4cf6j42paoaz mysql]# cat redis-cplusplus-client/test.cpp
#include <stdio.h>
#include <mysql.h>
#include "redisclient.h"
using namespace boost;
using namespace std;

static redis::client *m_client = NULL;

extern "C" char *redis_hset(UDF_INIT *initid, UDF_ARGS *args, char *result,
unsigned long *length, char *is_null, char *error) {
    try {
    if(NULL == m_client) {
        const char* c_host = getenv("REDIS_HOST");
        string host = "127.0.0.1";
        if(c_host) {
            host = c_host;
        }
        m_client = new redis::client(host);
    }
    if(!(args->args && args->args[0] && args->args[1] && args->args[2])) {
        *is_null = 1;
        return result;
    }

    if(m_client->hset(args->args[0], args->args[1], args->args[2])) {
        return result;
    } else {
        *error =1;
        return result;
    }
    } catch(const redis::redis_error& e){
    return result;
    }

}

extern "C" my_bool redis_hset_init(UDF_INIT *initid, UDF_ARGS *args, char
*message) {
    if (3 != args->arg_count) {
        strncpy(message, "Please input 3 args for: hset('key', 'field',
'value');", MYSQL_ERRMSG_SIZE);
        return -1;
    }
    if (args->arg_type[0] != STRING_RESULT  || args->arg_type[1] !=
STRING_RESULT  ||  args->arg_type[2] != STRING_RESULT) {
        strncpy(message, "Args type error: hset('key', 'field', 'value');",
MYSQL_ERRMSG_SIZE);
        return -1;
    }
    args->arg_type[0] = STRING_RESULT;
    args->arg_type[1] = STRING_RESULT;
    args->arg_type[2] = STRING_RESULT;
```

```
    initid->ptr = NULL;
    return 0;
}
```

## 4、编译命令

```
g++ -shared -fPIC -I /usr/include/mysql -lboost_serialization -lboost_system -lboost_thread -o libmyredis.so anet.c test.cpp
```

## 5、将生成的动态库拷贝到mysql的插件目录下

```
cp libmyredis.so /usr/lib64/mysql/plugin/
```

## 6、新建函数并测试

```
CREATE FUNCTION redis_hset RETURNS STRING SONAME 'libmyredis.so';
SELECT redis_hset('test', 'idhsd', 'Oddfdfdfdfk;df9388334');
drop function redis_hset;

127.0.0.1:6379> HGETALL test
1) "id"
2) "09388334"
3) "ids"
4) "Odfdfdf9388334"
5) "idhs"
6) "Odfdfk;df9388334"
```