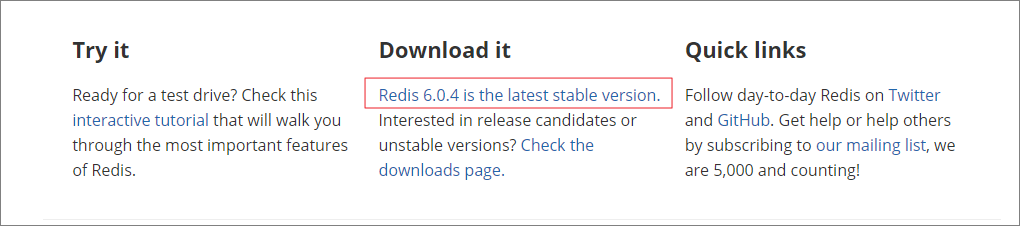
**Install Redis & Celery**

**下载**

官网下载：

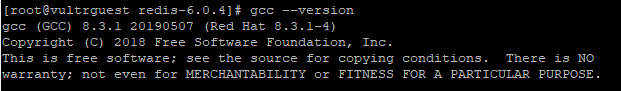
<https://redis.io/>



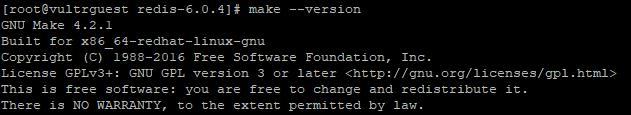
wget <http://download.redis.io/releases/redis-6.0.4.tar.gz>

**解压、编译、安装**

yum install gcc



yum install make



把redis-\*\*\*.tar.gz解压到/usr/local目录；

cd redis-6.0.4

make MALLOC=libc

cd src && make install

**设置redis后台启动、开机自启动**

1、在/etc目录下新建redis目录

mkdir /etc/redis

2、将 /usr/local/redis-6.0.4/redis.conf 文件复制一份到/etc/redis目录下，并命名为6379.conf

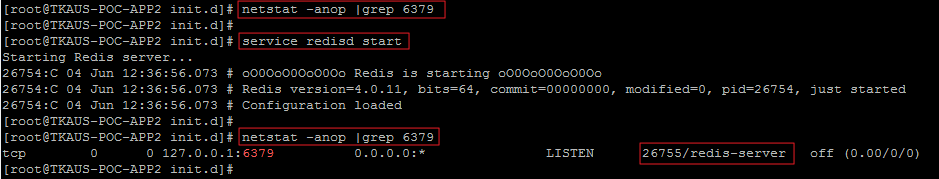
检查 redis.conf 其中的daemonize由no改为yes；

cp /usr/local/redis-6.0.4/redis.conf /etc/redis/6379.conf

3、将redis的启动脚本复制一份放到/etc/init.d目录下

cp /usr/local/redis-6.0.4/utils/redis\_init\_script /etc/init.d/redisd

4、后台启动redis



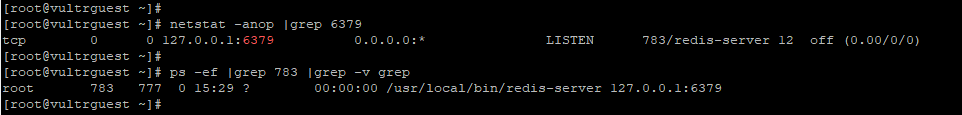
5、设置redis开机自启动

先切换到/etc/init.d目录下

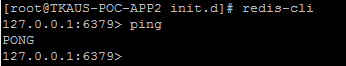
然后执行自启命令

chkconfig redisd on

6、reboot 并检视监听 6379 端口的进程，发现 redis-server 果然自动启动了。



7、联通测试



**安装Flask**

pip install flask

**安装Celery**

pip install celery

**代码框架**

Parallel.py

运行这个程序会提供一个监听本地5000端口的服务，接收到客户端发来的异步请求，会把这个请求转出一个任务放入celery任务队列中，并立即返回{"msg": 'ok', "code": '0'}消息给客户端。等到计算任务完成之后，通过客户端提供的回调接口把计算结果返回给客户端。

|  |
| --- |
| #!/usr/local/python3/bin/python3  import ...  app = Flask(\_\_name\_\_)  celery = Celery(app.name, broker='redis://127.0.0.1:6379', backend='redis://127.0.0.1:6379//0')  @app.route('/')  def index():...  @app.route('/asynsum', methods=['GET', 'POST'])  def asynsum\_dispatch():...  @celery.task  def add(json\_d):...  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |

主要函数说明：

asynsum\_dispatch()

这个函数处理客户端发送的异步请求。收到请求后提取参数，转成一个任务后放入任务队列，并立即返回一个消息给客户端。

add(json\_d)

这个函数是celery的task函数，通过celery命令行启动多个worker并发的处理任务队列中的任务，并通过客户端提供的回调接口把计算结果通知给客户端。

Callback.py

这个文件启动一个Flask进程监听server端通过回调返回的计算结果，并发taskID和计算结果打印在屏幕上。

|  |
| --- |
| #!/usr/local/python3/bin/python3  import …  app = Flask(\_\_name\_\_)  @app.route('/callback', methods=['POST'])  def callback():…  if \_\_name\_\_ == '\_\_main\_\_':  app.run(port='5002', debug=True) |

主要函数说明：

callback()

这个函数在收到回调请求的时候被调用，从POST请求中提取出taskID、计算结果，打上时间戳，然后打印在屏幕上。

send\_request.py

通过进程池并发的发送计算请求给server端。

|  |
| --- |
| #!/usr/local/python3/bin/python3  import …  def send\_request(task\_id, x, y):…  if \_\_name\_\_ == '\_\_main\_\_':  p = Pool(5)  for i in range(10):  task\_id = f'{10000000 + i}'  x = i + 1  y = x \* 3  p.apply\_async(send\_request, args=(task\_id, x, y,))  p.close()  p.join() |

主要函数说明：

send\_request(task\_id, x, y)

这个函数把要计算的数据以及获取计算结果的回调地址放在POST请求中发送给server。

**运行并观察结果**

启动parallel.py监听5000端口，接收客户端的计算请求：

nohup ./paralle.py &

启动callback.py监听5002端口，接收server端口发送的计算结果，并打印在屏幕上：

./callback.py

通过以下命令启动celery的worker进程：

/usr/local/python3/bin/celery -A parallel.celery worker --concurrency=5 -l info

启动send\_request.py发送计算请求给server端：

./send\_request.py

可以观察到屏幕上打印出来的计算结果：

