Chapter - 2 : Parallel Processing with Celery

By default `celery` is concurrency enabled and it created multiple workers simultaneously so that multiple tasks can be processed at the same time

we can see the same using `top` and/or `htop` commands

Let's see the concurrency in action by adding one more function to send the notification for whatsapp also

task_one.py

```python
# senf SMS notifications to the account holder
@app.task
def task_send_sms(account_no, message):
    time.sleep(5)  # sleep for 5 seconds
    print("Sending SMS to account holder of {0} with message \n\n {1}".format(account_no, message))

@app.task
def task_send_whatsapp(account_no, message):
    time.sleep(5)  # sleep for 5 seconds
    print("Sending WhatsApp to account holder of {0} with message \n\n {1}".format(account_no, message))
```

entry_point.py

```python
# Deposit money into the bank account and send the sms & whatsapp
def deposit_and_send_sms(account_no, amount, message):
    bank_deposit_money(account_no, amount)
    # send SMS and WhatsApp
    task_send_sms.delay(account_no, message)
    task_send_whatsapp.delay(account_no, message)
```

SMS and WhatsApp

Example : 1

we can also setup the concurrency using the following command

```
celery -A task_one worker --loglevel=INFO --concurrency=1
```

Generally concurrency should correspond to the number of cores (Default behaviour), however, there is no "fit all formula" and we need to apply the best of our judgement based on the problem we're solving

We can verify the concurrency behaviour by applying some sleep on the messages being sent and check it with concurrency value of 1 ,2 or even more

```
celery -A task_one worker --loglevel=INFO --concurrency=1
```

```
app = Celery('cel_main', backend='rpc://', broker='pyamqp://')

# send SMS notifications to the account holder
@app.task
def task_send_sms(account_no, message):
    time.sleep(1)
    print("Sending SMS to account holder of {0} with message \n\n {1}".
          format(account_no, message))

@app.task
def task_send_whatsapp(account_no, message):
    time.sleep(5)
    print("Sending WhatsApp to account holder of {0} with message \n\n {1}".
    format(account_no, message))
```

With Single Worker

Example : 2

Celery workers are generally limited by the number of file descriptors, if the limit is exhausted, you'll get an error

```
OSError: [Errno 24] Too many open files
```

You need to play with your `ulimit` values to get around this behaviour

Auto Scaling & De-Scaling

It may be sometimes required to let celery auto scale instead of keeping lots of worker processes in the memory all the time, we don't need to eat the resources when they're not needed.

Here is the command to enable auto scaling while starting the celery

```
celery -A task_one worker --autoscale=1,10
```

The auto scale parameters are `MIN` and `MAX` number of workers

Example : 2 - With AutoScale Parameter