Proyecto Reportes

Silva Serna José del Refugio

5 de junio de 2018

Índice

1. Resumen

Se describe como se llevó a cabo un proyecto con ayuda de celery, apoyados por la estructura de django y con redis el cual se corrio en otra maquina de la misma red, con la finalidad de poder enviar y resivir información de un programa a redis estos en maquinas disitintas.

2. Palabras clave

Celery, Redis, Django.

3. Marco teórico

Celery: Es una aplicación que nos permite crear tareas de trabajo asíncronas gestionadas por un gestor de colas que está basada en el envío de mensajes de manera distribuida. Se focaliza en operaciones en tiempo real pero también soporta la calendarización de tareas, es decir, puede lanzar tareas que se tengan que ejecutar en un momento determinado o de manera periódica. Las unidades de ejecución, llamadas tareas, se ejecutan de manera concurrente en uno o más nodos de trabajo. Estas tareas pueden ejecutarse de manera asíncrona bien de manera síncrona (esperando hasta que la tarea está lista).

Redis: es un motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos durable o persistente. Está escrito en ANSI C por Salvatore Sanfilippo quien fue patrocinado por VMware

Django: Django es un framework web de alto nivel que permite el desarrollo de sitios web rapido, seguro y mantenible. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda.

4. Introducción

Se desarrolló un proyecto con la ayuda de celery, django y redis, y con atom como editor, el proyecto maneja reportes los cuales cada 5 y 10 segundos se irán generando, en este documento se verá el cómo se realizó la conexión del cliente con el servidor mostrando la configuración y los pasos a seguir para que redis se pueda correr en otra máquina conectada a la misma red como servidor.

5. Planteamiento del problema

En la actualidad se generan y se imprimen una gran cantidad de reportes con al cual se puede transmitir mucha informacion en el dia a dia, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios. Por lo cual es importante que se tenga un buen manejo de expedicion para que la gente que lo necesita lo tenga al alcance de su mano y haga un buen uso con la información que estos contienen.

6. Desarrollo

Utilizando un ambiente virtual se creó el proyecto llamado Reporte y una aplicación llamada reportes, en el cual cada 5 segundos realizara una tarea, la cual la tarea es en una lista de reportes con los campos report id, created at, user, colored status, async task status los cuales se declaran en el modelo.

```python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.db import models

# Create your models here.
class ReporteItem(models.Model):
    report_id = models.CharField(max_length=120, null=True, blank = True)
    created_at= models.CharField(max_length=120, null=True, blank = True)
    user = models.CharField(max_length=120, null=True, blank = True)
    colored_status = models.CharField(max_length=120, null=True, blank = True)
    async_task_status =models.CharField(max_length=120, null=True, blank = True)
    mensaje = models.CharField(max_length=120, null=True, blank = True)

    def __str__(self):
        return str(self.mensaje)
```

. Model

En el modelo se crea como ya se explico los valores en una clase llamada ReporteItem

```python
from __future__ import absolute_import, unicode_literals
import random
from celery.decorators import task
from .models import ReporteItem
from datetime import datetime
@task (name="mandar_reporte")
def sent(report_id,created_at,user,colored_status,async_task_status):

    list_mails={"sent":[
    {'report_id':'Solicitado por Oficinas', 'created_at':datetime.now().strftime('%a, %d %B %Y %H:%M:%S %Z'),'user':"Administrador",
    {'report_id':'Solicitado por Almacen', 'created_at':datetime.now().strftime('%a, %d %B %Y %H:%M:%S %Z'),'user':"Administrador",
    ]}
    i=random.randint(0,1)
    report_id=list_mails['sent'][i]['report_id']
    created_at=list_mails['sent'][i]['created_at']
    user=list_mails['sent'][i]['user']
    colored_status=list_mails['sent'][i]['colored_status']
    async_task_status=list_mails['sent'][i]['async_task_status']

    sent_report_id = report_id
    sent_created_at= created_at
    sent_user= user
    sent_colored_status = colored_status
    sent_async_task_status = async_task_status
    mensaje = report_id,created_at,user,colored_status,async_task_status
    new_obj = ReporteItem.objects.create(report_id=sent_report_id,created_at=sent_created_at,user=sent_user, colored_status=sent_col
    return mensaje
```

. Task

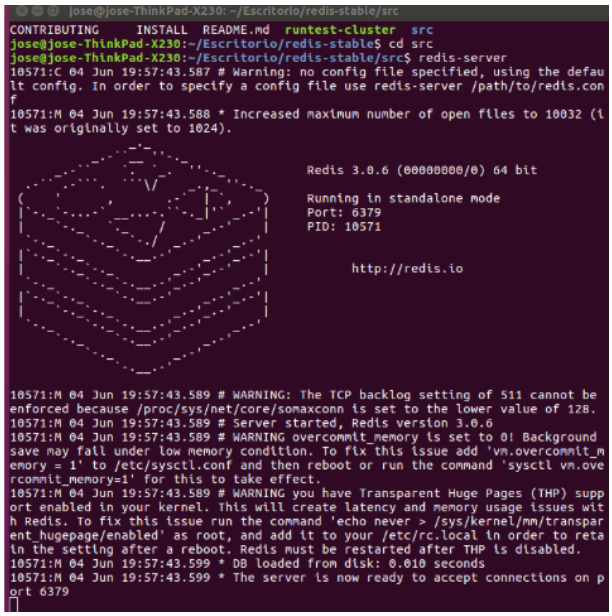
Como tarea en el task se crea una lista y con un número random que tenga el mismo tamaño que la lista, para guardar los valores de la lista en variables con ayuda del random que nos estará dando los elementos de la lista aleatoriamente para y con esto poder retornar las como un mensaje.

```python
from __future__ import absolute_import, unicode_literals
import os
from celery import Celery
from celery.schedules import crontab

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "Reportes.settings")
app = Celery('proj')
app.config_from_object('django.conf:settings', namespace = 'CELERY')
app.autodiscover_tasks()


app.conf.beat_schedule ={
    'add-every-5-seconds':{
        'task':'mandar_reporte',
        'schedule':5.0,
        'args':(16,16,16,16,16),
        },
    'add-every-10-seconds':{
        'task':'mandar_reporte',
        'schedule':10.0,
        'args':(16,16,16,16,16),
        },
    }
@app.task(bind=True)
def debug_task(self):
    print('Request:{0!r}'.format(self.request))
```

. Celery

Para poder correr la tarea que se creó en task en un determinados tiempos tenemos que llamarlas en celery gracias a que importamos el módulo de celery podemos usar app.conf.beat schedule que este nos permite asignar el tiem-

po a las tareas con parámetros.



. Redis-server

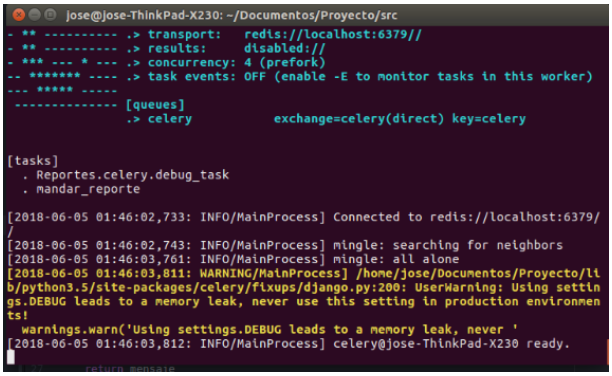
Ahora para para poder realizar la conexión e ingresar a redis como servidor mediante la terminar entrando a la src en la carpeta redis e ingresar el comando redis-server con el cual se correrá redis como servidor.

Después el usuario que se conectara en una máquina externa teniendo en cuenta que se tiene que conectar a la misma red que en la que se encuentra el servidor, el usuario que ingresa como cliente necesariamente tiene que modificar el archivo local en la parte donde se tiene la dirección de redis, localhost se cambia a la ip del servidor.

```
124  STATIC_URL = '/static/'
125  CELERY_BROKER_URL='redis://172.16.1.130:6379'
126  CELERY_RESULT_CONTENT='redis://172.16.1.130:6379'
127  CELERY_ACCEPT_CONTENT=['application/json']
128  CELERY_RESULT_SERIALIZER='json'
129  CELERY_TIMEZONE=TIME_ZONE
130
```

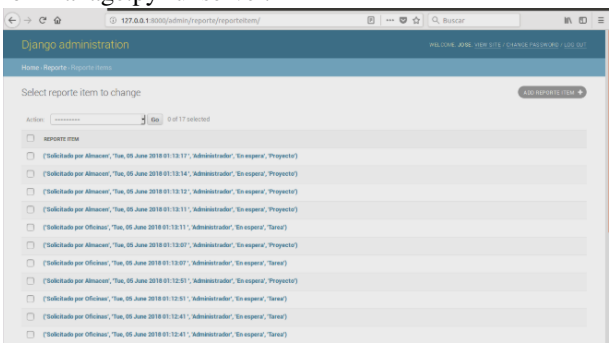
. Local

Después el cliente necesita correr el worker para que este le mande permisos al servidor, y después el servidor ingresa como cliente y da los permisos al cliente con el comando CONFIG SET protected-mode no, lo que hará que no esté protegido y el cliente pueda ingresar, teniendo en cuenta que solo 1 cliente se debe de conectar ya que tendrán problemas por conectarse al mismo tiempo.



. Worker

Y a continuación el cliente ejecuta el beat para que se empiecen a realizar las tareas y estas se puedan ver reflejadas en el servidor y también en django con el comando python manage.py runserver.



. Django administration

7. Implementación

Este proyecto se de mucha utilidad ya que se puede implementar en la solicitud de reportes y en la generacion de estos, con esto se podria facilitar estos procesos.

8. Conclusión

Como se pudo ver, la herramienta redis y nos funcionan muy bien en este tipo de proyectos en los cuales se maneja celery, ya que el motor de base de datos que al conectarlo en red puede servirnos para cualquier proyecto a futuro y con este ver las llaves que le envia el cliente al servidor.

9. Referencias

https://es.wikipedia.org/wiki/Redis
http://blog.enriqueoriol.com/2014/06
/integrando-celery-con-django-programar.html
https://developer.mozilla.org/es/docs/Learn/Server-side
/Django/Introducción—