我们公司现在是用Django框架（Python语言）开发的Web项目，我们现在的定时任务实现很简单

将定时任务编写成command命令脚本，然后k8s定时跑脚本执行定时任务

我现在接手调研定时任务框架并将其引入到工程项目中

我调研了以下解决方案：

Python中定时任务的解决方案，总体来说有四种，分别是： crontab、 scheduler、 Celery、 APScheduler

实现方案	优点	缺点	备注
crontab		不适合多台服务器的配置	
scheduler		scheduler太过于简单 ▪ 没有提供任务执行的配置参数：Executor、Timezone... ▪ 没有提供Event监听事件 ▪ 调用方式过于简单 ▪ 不能动态添加任务或持久化任务	
Celery	1. 支持分布式部署	1. 不能动态添加定时任务到系统中 2. 使用起来比较繁锁	
APScheduler	1. 提供基于cron、date、interval等触发方式的时间调度 2. 灵活，可动态增删定时任务并持久化 3. 支持多种存储后端 4. 可以配置Executor、JobStores、Lister等参数	不支持分布式部署（需要自己解决）	https://github.com/agronholm/apscheduler

方案选型

其中 crontab不适合多台服务器的配置、 scheduler太过于简单、 Celery依赖的软件比较多，比如broker依赖rabbitmq或者redis，比较耗资源。最好的解决方案就是 APScheduler。因此本次以APScheduler作为设计方案。

APscheduler不支持分布式锁，于是我添加了一个分布式

```python
class DistributedLock:
    def __init__(self, lock_key, ttl):
        self.client = redis_client.client
        self.lock_key = lock_key
        self.ttl = self.__fix_ttl(ttl)
        self.value = self.__generate_value_by_uuid()
        self.running = None

    def lock(self):
        set_nx_status = self.client.set(self.lock_key, self.value, ex=self.ttl, nx=True)
        if set_nx_status == redis_const.SetNXStatus.Fail:
            _LOGGER.info("[Cron Task] Set nx fail,key:{}".format(self.lock_key))
            return set_nx_status
        expire_time = dt.local_now() + dt.timedelta(self.ttl)
        _LOGGER.info(
            "[Cron Task] Set nx success, key:{},ttl:{},expire time:{},value:{}".format(
                self.lock_key, self.ttl, expire_time, self.value
            )
        )
        self.running = True
        refresh_lock_thread = threading.Thread(target=self.__refresh_lock)
        refresh_lock_thread.start()
        return set_nx_status

    def unlock(self):
        self.running = False

    def __generate_value_by_uuid(self):
        value = str(uuid.uuid4()).replace("-", "")
        return value

    def __fix_ttl(self, ttl):
        # 判断ttl是否小于2，如果小于，需要修改为2，为了防止__refresh_lock时候锁已经被delete
        if ttl < 2:
            ttl = 2  # 不能设置为2
        return ttl
```

现在APscheduler能支持动态添加任务，也支持分布式部署了，后期只需要开发一个前端界面就能实现通过前端

但是leader不想重复造轮子，想找一个开源的定时任务框架，支持分布式部署，支持通过web界面管理定时任务，支持动态新增定时任务等功能

我现在找了

- APscheduler（支持Python）：CronTask概要设计文档
- E-Job（支持Java）：https://shardingsphere.apache.org/elasticjob/current/cn/overview/
- AirFlow（支持Python、有管理界面、使用起来很不便）：https://airflow.apache.org/docs/apache-airflow/stable/ui.html
- xxl-job（仅支持Python脚本，当定时任务中有依赖其他模块时候运行不了、有管理界

面）：https://www.xuxueli.com/xxl-job/#%E3%80%8A%E5%88%86%E5%B8%83%E5%BC%8F%E4%BB%BB%E5%8A%A1%E8%B0%83%E5%BA%A6%E5%B9%B3%E5%8F%B0XXL-JOB%E3%80%8B，找到了一个Python依赖：https://fcfangcc.github.io/pyxxl/

- Celery（支持Python、有管理界面）：https://docs.celeryq.dev/en/stable/django/first-steps-with-django.html#using-celery-with-django
- DolpinScheduler：https://dolphinscheduler.apache.org/zh-cn/

有没有推荐的开源定时任务框架