# گزارش کار چالش فنی

## مقدمه

در این چالش خواسته شده بود که سرویسی نوشته شود که یک فایل کانفیگ حاوی نام اندکس، حداکثر تعداد داکیومنت مجاز در ایندکس و زمان دلخواه جهت چک کردن ایندکس را بخواند و اگر تعداد داکیومنت ها بیشتر تعداد مجاز بود، آن ها را به پورت *514* سرویس *syslog* ارسال کند و همچنین آن ها را از *Elasticsearch* پاک کند و *log* اجرای هر مرحله از اجرای برنامه در یک فایل ذخیره شود.

و همچنین یک *API* برای بازنویسی فایل کانفیگ، یک *Linux systemd service* و یک فایل نصبی *deb*. برای آن در نظر گرفته شود.

## پیاده سازی

این پروژه حاوی فایل های *main.py, apk.py, conf_api.py, config.json, apk.log* می باشد که در ادامه هر کدام را بررسی میکنیم.

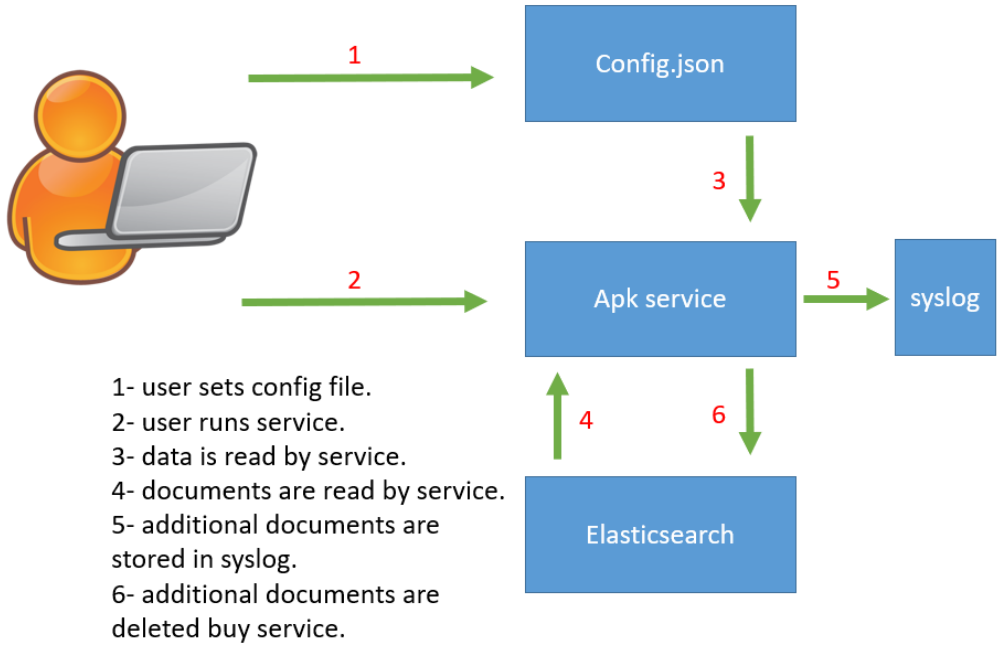
فایل *config.json* یک فایل *JSON* برای ذخیره سازی کانفیگ سرویس می باشد. نمونه کانفیگ ذخیره شده در این فایل به صورت زیر است.

```
{
 "product": {
   "max_value": "30000",
   "check_time": "* * * * *"
 }
}
```

با اجرای فایل *main.py* ابتدا یک شئ از کلاس *APK* موجود در فایل *apk.py* ساخته و سپس متد *run* متعلق به آن کلاس فراخوانی میشود. در متد *run*، فایل کانفیگ خوانده میشود و به ازای هر *index* موجود در آن فایل متد *make_thread* فراحوانی میشود.

در متد *make_thread* برای *index* پاس داده شده به آن یک نخ درست شده و متد *check_index* فراخوانی می شود. در متد *check_index* یک کانکشن با سرور *elasticsearch* برقرار میکنیم. در یک حلقه بینهایت ابتدا زمان بعدی بررسی کردن *index* مورد نظر را بدست می اوریم. این کار با توجه به پارامتر *check_time* هر *index* موجود در فایل کانفیگ و با استفاده از متد *get_next* از ماژول *croniter* انجام میشود.

بعد از به دست اوردن زمان، با استفاده متد *until* از ماژول *pause* پروسه تا زمان مورد نظر متوقف میشود. بعد از رسیدن زمان مورد نظر ابتدا تمام داکیومنت های ایندکس را میخوانیم و تعداد داکیومنت ایندکس با حداکثر تعداد آن (*max_value*) مقایسه میشود. اگر بیشتر از تعداد مجاز بود، داکیومنت های قدیمی تر را بدست می اوریم (با توجه به تاریخ ذخیره شدن داکیومنت در ایندکس) و با استفاده از متد *syslog* آن ها را با پروتکل UDP به پورت 514 سرویس syslog ارسال میکنیم. سپس *id* داکیومنت های بدست امده را در یک لیست ذخیره میکنیم تا با استفاده از این *id*ها و متد *delete_by_query* کلاس *Elasticsearch* موجود در ماژول *elasticsearch* داکیومنت ها را از ایندکس حذف کنیم.




همچنین با اجرای فایل *main.py* یک شئ از کلاس *ConfigApi* موجود در فایل *cong_api.py* ساخته و *run* میشود. با اجرای این کد میتوانیم از طریق سرویس *flask_restful* فایل کانفیگ را بازنویسی کنیم. کافی است دستور زیر را در ترمینال وارد شود.

```
curl -XPOST -H "Content-Type:application/json" 127.0.0.1:3000/confapi -d '{data}'
```

با اجرای هر مرحله از این سرویس، لاگ های آن مانند *id* داکیومنت و نام ایندکسی که داکیومنت از ان حذف شده، تعداد داکیومنت حذف شده از ایندکس، تغییرات فایل کانفیگ و یوزرنیم کاربری که برنامه را اجرا کرده است، با استفاده از ماژول *logging* در فایل *apk.log* ذخیره میشود.

برای ساخت فایل *deb*.، در یک دایرکتوری، دایرکتوری های *DEBIAN* و *usr/share/apk* را ایجاد کرده و فایل های پروژه را در دایرکتوری *usr/share/apk* که ساخته ایم کپی میکنیم. همچنین فایل های *control* با محتویات

```
Package: APK
Version: 1.0
Section: base
Priority: optional
Architecture: all
Maintainer: mjavadrezaei77@gmail.com
Description: APK technical test debi file
```

که اطلاعاتی مربوط به پکیج هست و فایل *postinst* با محتویات

```
/cp /usr/share/apk/apk.service  /etc/systemd/system
/cp /usr/share/apk/apk  /etc/init.d
```

```
systemctl daemon-reload
systemctl start apk
```

که دستوراتی است که بعد از کپی شدن فایل ها در سیستم مقصد اجرا میشود، را در دایرکتوری *DEBIAN* ایجاد میکنیم.

با دستور `dpkg-deb --build` فایل *deb*. ساخته میشود..

## روش استفاده

برای اجرای این برنامه به صورت دستی کافی است دستور زیر را در ترمینال وارد کنیم.

```
python main.py -C [config file address] -H [Remote syslog server ip]
```

برای اجرای سروریس این برنامه از طریق *systemd*، ابتدا یک فایل *apk.service* را در مسیر */etc/systemd/system/* ایجاد میکنیم. متحوای این فایل در زیر امده است.

```
[Unit]
Description=APK Service
After=multi-user.target
Conflicts=getty@tty1.service

[Service]
ExecStart=/usr/bin/python3 <absolute path of main.py>

[Install]
WantedBy=multi-user.target
```

بعد از ذخیره این فایل با اجرای دستورات زیر در ترمینال سرویس را اجرا میشود.

```
systemctl daemon-reload
systemctl enable apk.service
systemctl start apk.service
```

با نصب فایل deb.، فایل های برنامه در مسیر */usr/share/apk* سیستم مقصد ذخیره میشود و دستورات فایل *postinst* اجرا میشود و سرویس *APK* به حالت اجرا در می‌اید. لازم به ذکر است که این پکیج با پایتون 3.8.5 تست شده است