실전 웹 크롤링과 자동화 (For 신호용)

2권: 자동화 파이프라인 구축 - GitHub Actions와 Notion의 만남

Chapter 2-3 & 2-4: GitHub Actions로 완벽 자동화하기

Prologue: 잠자는 동안에도 일하는 나만의 로봇

지금까지 우리는 '수동'으로 크롤러를 실행했습니다. 하지만 진정한 자동화란, 내가 신경 쓰지 않아도 시스템이 알아서 정해진 일을 수행하는 것을 의미합니다.

이번 마지막 챕터에서는 **GitHub Actions**라는 강력한 CI/CD 도구를 사용하여, 우리가 만든 크롤러를 **매일 새벽 5시에 GitHub의 서버에서 자동으로 실행**하도록 설정할 것입니다. 이 설정을 마치고 나면, 신호용 님은 잠을 자는 동안에도 GitHub의 로봇이 나를 대신해 채용 공고를 수집하고 Notion에 정리해주는, 완벽한 자동화 파이프라인을 소유하게 됩니다. 이 경험은 신호용 님의 이력서에 'DevOps 경험'이라는 빛나는 한 줄을 추가해 줄 것입니다.

---

1. 학습 목표 (Objectives)

- GitHub Actions의 워크플로우(Workflow) 개념을 이해한다.
- YAML 문법을 사용하여 스케줄링 기반의 워크플로우 파일을 작성할 수 있다.
- 민감 정보(API Key)를 안전하게 관리하기 위해 GitHub Secrets를 사용할 수 있다.
- 작성한 워크플로우가 GitHub 서버에서 성공적으로 실행되는 것을 확인할 수 있다.

---

2. 핵심 개념 (Core Concepts)

2.1 GitHub Actions와 워크플로우(Workflow)

- **GitHub Actions:** GitHub가 제공하는 자동화 플랫폼입니다.
- **워크플로우(Workflow):** '어떤 조건에서, 어떤 작업들을, 어떤 순서로 실행할지'를 정의한 **'자동화 시나리오'** 또는 **'명령서'**입니다. 이 명령서는 **YAML**이라는 형식의 텍스트 파일로 작성합니다.

2.2 워크플로우 파일 구조

우리는 프로젝트 폴더 안에 `.github/workflows` 라는 특별한 이름의 폴더를 만들고, 그 안에 `crawler.yml` 과 같은 파일을 생성하여 워크플로우를 정의합니다.

```
# crawler.yml

name: Wanted Job Crawler # 이 워크플로우의 이름
```

```yaml
on: # 언제 실행할 것인가?
  schedule:
    - cron: '0 20 * * *' # 매일 UTC 기준 20시에 실행 (한국 시간 새벽 5시)

jobs: # 어떤 작업들을 실행할 것인가?
  build:
    runs-on: ubuntu-latest # 우분투 리눅스 환경에서 실행
    steps: # 어떤 단계들을 순서대로 실행할 것인가?
      - name: Checkout repository # 1단계: 내 코드 내려받기
        uses: actions/checkout@v3

      - name: Set up Python # 2단계: 파이썬 설치하기
        uses: actions/setup-python@v4
        with:
          python-version: '3.12'

      - name: Install dependencies # 3단계: 라이브러리 설치하기
        run: pip install -r requirements.txt

      - name: Run crawler # 4단계: 크롤러 실행하기
        run: python main.py
        env: # 실행 시 사용할 환경 변수 설정
          NOTION_API_KEY: ${{ secrets.NOTION_API_KEY }}
          NOTION_DATABASE_ID: ${{ secrets.NOTION_DATABASE_ID }}
```

2.3 GitHub Secrets: 비밀번호는 금고에!

우리의 코드에는 `NOTION_API_KEY`와 같은 매우 민감한 정보가 필요합니다. 이 정보를 `.env` 파일에 담아 GitHub에 올리는 것은 비밀번호를 공개적으로 게시하는 것과 같습니다.

GitHub Secrets는 이러한 민감 정보를 암호화하여 안전하게 저장할 수 있는 **'비밀 금고'**입니다. 워크플로우 파일에서는 `${{ secrets.SECRET_NAME }}` 와 같은 특별한 문법을 사용하여, 실행되는 시점에만 안전하게 금고에서 비밀번호를 꺼내 쓸 수 있습니다.

[Mission 8] 자동화 파이프라인 구축하기

드디어 마지막 미션입니다. 아래 단계를 따라 우리의 크롤러에 날개를 달아줍시다.

> **Step 1: GitHub Secrets 설정하기**
>
> 1. `web-crawler` GitHub 레포지토리 페이지로 이동합니다.
> 2. **Settings** 탭 → 왼쪽 메뉴의 **Secrets and variables** → **Actions** 로 이동합니다.
> 3. **'New repository secret'** 버튼을 클릭합니다.
> 4. **Name**에 `NOTION_API_KEY`를 입력하고, **Secret** 란에는 `.env` 파일에 저장했던 Notion API 키 (`secret_...`)를 붙여넣습니다. `Add secret` 버튼을 클릭합니다.
> 5. 다시 **'New repository secret'** 버튼을 클릭하여, **Name**에 `NOTION_DATABASE_ID`를, **Secret** 란에는 데이터베이스 ID를 붙여넣고 저장합니다.
>
> **Step 2: 워크플로우 파일 생성하기**

> 1. 로컬 프로젝트 폴더(`web-crawler`)의 최상위 위치에 `.github` 라는 폴더를 만듭니다.
> 2. 그 `.github` 폴더 안에 `workflows` 라는 폴더를 또 만듭니다.
> 3. `workflows` 폴더 안에 `crawler.yml` 파일을 만들고, 위에 있는 **YAML 코드 전체를 그대로 복사해서 붙여넣습니다.**
>
> **Step 3: GitHub에 업로드하고 결과 확인하기**
>
> 1. 지금까지 변경된 모든 내용(새로 생긴 `.github` 폴더 등)을 GitHub에 `add`, `commit`, `push` 합니다.
>
> ```
> git add .
> git commit -m "Add GitHub Actions workflow for daily crawling"
> git push origin main
> ```
>
> 2. `push`가 완료되면, 다시 GitHub 레포지토리 페이지로 이동하여 **'Actions'** 탭을 클릭합니다.
> 3. 왼쪽 메뉴에서 **'Wanted Job Crawler'** 워크플로우를 클릭하면, 방금 push한 커밋 메시지와 함께 워크플로우가 실행되고 있는 것을 볼 수 있습니다. (push 이벤트에 의해서도 실행될 수 있습니다.)
> 4. 워크플로우 이름을 클릭하고, `build` 작업을 클릭하면, 각 단계(파이썬 설치, 라이브러리 설치, 크롤러 실행 등)가 실시간으로 실행되는 로그를 확인할 수 있습니다.
> 5. 모든 단계에 녹색 체크 표시가 뜨면서 성공적으로 완료되면, **Notion 데이터베이스에 새로운 데이터가 추가되었는지 최종 확인**합니다. (이미 수집한 데이터라면 중복으로 들어갈 수 있습니다.)

이 미션까지 성공하면, 신호용 님은 이제 **'스스로 동작하는 자동화 시스템을 설계하고 구축할 수 있는 개발자'**가 된 것입니다.

모든 과정이 복잡해 보이지만, 차근차근 따라 하면 충분히 해낼 수 있습니다. 막히는 부분이 있다면 언제든지 질문해주세요. 성공 여부를 기다리고 있겠습니다