fabric을 활용한 배포 자동화

지금까지 배포 자동화 스크립트를 만들어야겠다는 생각을 하면서도 바쁘다는 핑계 때문에 미루고 살아왔다. 그렇게 살다가 수업에서도 활용하고 실무 프로젝트에서도 활용해야겠다는 생각으로 만들기 시작했다. 현재 환경이 지속적으로 한 가지 업무에 집중할 수 있는 환경이 아니라 시간이 나는 짬짬이 학습하면서 만들다보니 시간이 많이 걸렸다. 사실 최종 완성된 버전은 아니지만 추후 필요한 시점에 현재까지 학습한 내용을 기반으로 구축할 수 있겠다는 생각이 들어 지금 시점에 간단하게 정리해 본다.

내가 배포 자동화 스크립트를 만들기 위해 사용한 오픈 소스는 fabric이다. 다양한 오픈 소스들이 있었는데 가볍게 시작하기에는 딱 좋았다. 이전 회사에서 capistrano를 사용한 경험도 있어 capistrano를 기반으로 할까도 생각해 봤는데 학생들이 ruby에 대한 경험이 없어 python 기반인 fabric으로 시작했다.

시작은 fabric에 대한 맛보기를 하기 위해 정말 작은 과정을 fabric으로 자동화했다.

배포 자동화 1단계 - one project, one environment 초기 버전

1단계 배포 과정은 정말 단순하다.

- 배포를 위한 환경 설정
- 프로젝트 빌드
- tomcat 서버 stop
- war 파일 복사
- tomcat 서버 start

```
from fabric.api import env,execute
from fabric.operations import local

env.catalina_home="/Users/javajigi/my-workspace/apache-tomcat-7.0.34/"

def hostname():
    local('uname -a')

def build():
    local('mvn clean package')

def start():
    local('%s/bin/startup.sh' % env.catalina_home)  # tomcat instance
stop

def stop():
    local('%s/bin/shutdown.sh' % env.catalina_home) # tomcat instance
stop

def copy():
    local('cp ./target/slipp-user-1.0.0.war %s/webapps/ROOT.war' %
env.catalina_home) # file copy

def deploy():
    execute(build)
    execute(stop)
    execute(copy)
    execute(start)
```

이 작업에 100% 몰입할 수 있는 환경이 아니였기 때문에 1단계 작업은 최대한 단순화해서 최초 작업을 마무리하는 집중했다.

배포 자동화 2단계 - one project, one environment

2단계 작업은 프로젝트 하나를 배포하기 위한 전체 과정을 설계한 후에 이 과정을 해결하기 위해 하나씩 해결하는 방법으로 진행했다. 2단계에서 최초 설계한 배포 과정은 다음과 같다.

- 배포 서버에서 버전 관리 시스템으로부터 소스 코드를 checkout(or pull)

- 현재 배포 버전을 위한 디렉토리를 생성한다.(current_release)
- 빌드 작업을 진행해 war 파일을 생성한다.
- 생성한 war 파일을 current_release에 복사한다.
- current_release 디렉토리를 n 대의 실서버에 복사한다.(scp 활용)
- n 대의 실서버에 복사한 war 파일의 압축을 해제한다.
- WAS를 stop한다.
- 현재 실 서비스 중인 symbolic link를 current_release 디렉토리 변경한다.
- WAS를 start한다.
- 정상적으로 배포되었는지 테스트한다.
- WAS에 에러가 발생하는 경우 rollback 한다.

위 과정 중 대부분의 작업은 마무리하고 마지막 과정에 해당하는 테스트와 rollback 작업은 하지 않았다. 이 작업에 작성한 스크립트를 통해 해결할 수 있으리라 생각한다.
작업은 위와 같이 작업 목록을 작성한 후 시간 나는 틈틈이 하나씩 해결하는 방식으로 진행했다. 프로그래밍을 할 때 TDD로 개발하는 과정으로 진행했다. 연속적으로 시간을 투자하지 못하는 상황에서는 이와 같은 접근 방식을 취해야 제대로 된 경험을 할 수 있겠다는 생각이 들었다.
2단계 배포 자동화 스크립트를 만드는 과정과 스크립트는 2st fabric examples 문서에서 참고할 수 있다.

배포 자동화 3단계 - multi project, multi environment

2단계에서 프로젝트 하나를 배포하는 과정에 대해 제대로 된 스크립트를 만들어 봤다. 다음은 프로젝트 하나가 아닌 여러 개의 프로젝트와 여러 환경으로의 배포 자동화를 만들기 위한 스크립트를 만드는 것에 집중했다.

3단계에서는 배포에 필요한 설정 파일을 관리하는 것에 집중했다. 특히 여러 개의 프로젝트를 관리하기 위한 디렉토리 구조와 개발, 스테이징, 실서버 환경별로 다른 설정 파일을 관리하는 방법을 찾았다. 이 부분은 python에 대한 경험이 있으면 어렵지 않게 해결할 수 있다.

3단계 배포 자동화 스크립트를 만드는 과정과 스크립트는 3rd fabric examples 문서에서 참고할 수 있다.

배포 과정 설계

지금까지 3단계의 과정을 통해 배포 스크립트에 대한 사용 방법과 해결책을 얻었다면 다음 과정은 배포 과정을 설계해야 한다. 각 배포 환경별로 공통되는 부분, 달라지는 부분을 파악해야 한다. 또한 각 프로젝트별로 공통되는 부분과 달라지는 부분을 파악한다. 이 부분에 대한 파악이 끝나면 다음을 이를 기반으로 디렉토리 구조를 설계하고 설정 파일을 통해 관리할 수 있도록 스크립트를 만들면 된다.

배포 스크립트를 만드는 과정도 프로그래밍을 하는 과정과 똑같다. 한번 만들어 놓는다고 끝나는 것이 아니다. 프로젝트가 추가되고, 환경이 바뀌어 스크립트가 변경될 때마다 중복이 있는지를 파악하고, 지속적으로 리팩토링하는 과정을 거쳐야 한다. 이런 과정을 거치려면 배포 스크립트 또한 버전 관리 시스템을 통해 철저하게 관리해야 한다.

이번 과정을 통해 느낀 점

내가 학교에 온 후 느끼는 가장 큰 부분은 기존 회사보다 너무 다양한 업무들을 접하게 된다는 것이다. 따라서 각 업무들 간에 context switching 비용이 높다. 이는 한 가지 업무에 집중해서 투자할 시간이 많지 않다는 것을 의미한다. 한 가지 일에 집중하지 못하다보니 깊이 있는 학습을 하는데도 한계가 있음을 느낀다. 이 부분을 어떻게 극복할까 고민하다 이 작업과 같이 단계를 나누어 접근하는 방식을 취했다. 한 번에 많은 부분을 학습할 수는 없지만 목표를 세우고, 목표에 대한 세부사항을 먼저 설계한다. 그 후 시간 나는 틈틈이 하나씩 해결하고, 학습하는 방식으로 접근했다. 이와 같이 접근한 결과 작업을 마무리하기까지 다소 긴 시간이 필요했지만 최초 내가 목표로한 단계까지 작업할 수 있었다. 이 같은 접근 방식으로 학교 선생님들 뿐만 아니라 다양한 업무로 인해 context switching 비용이 많은 지식 노동자들이 갖추어야 하는 습관이다.

내가 작성한 이 스크립트가 더 많은 이들에게 활용되어 지금보다 훨씬 더 안정된 버전으로 거듭났으면 하는 바람이다.