rescale

REST-API를 이용한 Rescale 작업 자동화

목차

안내 (반드시 숙지해 주십시오)

- 해당 내용은 **Rescale REST-API**를 사용하시기 편하시도록 작성된 가이드라인 코드입니다.
- 해당 코드는 **Rescale**의 공식 입장을 대변하지 않으며, **Rescale Web platform**을 사용했을때의 어떤 서비스 수준도 담보하지 않습니다.
- **Rescale**은 기본 계약 상으로 해당 코드에 대해서 어떠한 기술지원도 제공할 의무가 없습니다. **(Global support** 기술 지원 포함)
- 본 코드의 사용의 모든 책임은 최종 사용자가 지게 되며, **Rescale**은 이에 대해 어떠한 책임이나 의무 사항도 지지 않습니다.
- **Community base**로 해당 코드에 대한 개선 사항을 제안할 수 있지만, 그것이 해당 내용 개선을 확약하거나, 개발을 하겠다고 **commit**하는 것은 아닙니다.
- 본 코드의 목적은 **Rescale REST-API** 자동화 가이드라인의 한 예시로 사용될 수 있는 것이며, 이것이 스크립트가 유지 보수되어야 하거나 개선을 의미하는 것은 아닙니다.

1.들어가기에 앞서

- **사용에 앞서 본 문서를 주의 깊게 읽어 주시길 부탁드립니다.**
- Rescale에서의 작업 제출은 첫번째로는 Web 플랫폼을 이용한 작업 제출, 두번째로는 CLI 를 이용한 자동화, 마지막으로 REST-API를 이용한 자동화를 들 수 있습니다.
- GUI 환경을 제공하는 측면에서 Web 플랫폼을 사용하는 이점이 있겠지만, 숙련된 사용자라면 스크립트 기반의 워크플로우 자동화에 대한 요구가 있을 것으로 생각됩니다. 이에 CLI와 REST-API를 사용하는 것을 고려할 수 있습니다.
- 단순하게 작업을 제출하고 결과 파일을 다운로드 하는 것은 Rescale CLI의 기능인 rescale-cli submit -i <input-script-file> -E 를 사용하면 해결할 수 있습니다. 하지만, Rescale CLI에서는 작업 input 파일을 Rescale files에서의 지정과 workflow에 대한 것을 상세히 대응하기 어렵기 때문에, 사용자의 workflow에 맞게 자동화하기 위해서는 REST-API를 통한 자동화가 필요합니다.
- Rescale REST-API 에 대한 정보는 Rescale REST-API문서에서 얻을 수 있습니다.
  - https://engineering.rescale.com/api-docs/
- 본 자동화에 관련된 내용은 Rescale Job 을 자동화 하는 것에 초점을 맞추고 있으며, job scheduler나 PLM과의 통합 내용은 다루고 있지 않습니다.

2.Workflow 자동화를 위한 가정들

- Rescale REST-API를 이용하기 위해서 On-premise에서의 작업과 Rescale Cloud에서의 작업의 차이를 생각하면, 가장 큰 차이는 기존 on-premise에서 사용하시는 상시 가동되고 있는 병렬 파일 시스템이 클라우드에는 부재하기 때문에 (비용적인 측면이 가장 큰 문제입니다), Rescale 각 작업에서 인스턴스의 볼륨에 생성되는 작업 결과물이 Rescale files (AWS S3)로 sync되어야 한다는 점일 겁니다.
- 그래서 강조드리고 싶은 내용은 $HOME/work 폴더 이하의 결과 파일들만 작업 종료 시에 Rescale files로 sync 되며, 그 후 인스턴스 볼륨의 모든 데이터는 삭제된다는 점입니다.
- On-premise에서 사용하시는 것처럼 트리 구조에서 복잡한 상호관계로 얽혀있는 입력 파일들은 Rescale로 곧바로 적용하기 어렵습니다. 그래서 다음과 같이 상정했습니다.
    - 가정1. 모든 입력 파일은 특정 working directory 아래에 있는 개별 파일들로 한정된다. (기본적으로는 특별하게 지정하지 않으면, main.py가 실행되는 경로입니다)
    - 가정2. Rescale 로의 파일 전송의 용이함을 위해 해당 working directory의 파일들을 전부 압축하여 **rescale.tar.gz**으로 만든다.
    - 가정3. 본 REST-API 스크립트는 모든 입력자료와 결과파일을 **rescale.tar.gz**의 형태로 다룬다. (용량과 갯수를 최소로 해야 전송에 포함되는 작업 시간 단축 및 자동화 구성이 용이해집니다)

3.rescale_rest_api.py 설명

- **1. platform_my_token**
- 해당 함수는 Rescale REST-API사용을 위한 platform과 Rescale API key를 가져오기 위한 함수입니다.
  기본적으로 KR 플랫폼을 사용하시면 rescale_platform에 해당하는 값은 https://kr.rescale.com 입니다.
  그리고 my_token에 해당 하는 값은 “Token XXXXXXXXXXXXXXXXXXXXX” 와 같은 형태로 주어집니다.
- Rescale 사용을 위해서 각 user의 HOME directory아래에 다음과 같이 설정했을 것을 가정한 함수입니다.
  CLI 사용을 위해서라면, 이 부분을 진행하셨을 거라고 생각합니다.
  - $HOME/.config/rescale/apiconfig
  - apiconfig 내용 예
    [default]
    apibaseurl = https://kr.rescale.com
    apikey = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- 만일 다음과 같은 설정이 불필요하다고 생각되시면, **main.py에서 직접 rescale_platform과 my_token 값을 선언하고 platform_my_token 함수 실행문을 주석 처리 하십시오.**

3.rescale_rest_api.py 설명

- **2. create_tar_gz**
- 해당 함수는 Rescale files로 업로드하기 위해 특정 input_path (주어지지 않으면 현재 main.py가 실행되는 경로가 됩니다) 의 개별 파일들을 하나의 rescale.tar.gz으로 압축하는 명령어입니다.
- 앞서 말씀드린대로, rescale.tar.gz으로 압축하는 건 자동화를 간편하게 하기 위함 뿐만 아니라, 압축을 통해 Rescale 로 업로드 하는 시간을 단축하는 효과도 있습니다
- 정상적으로 만들어지면 다음과 같이 출력됩니다.
  Successfully created rescale.tar.gz in <경로>

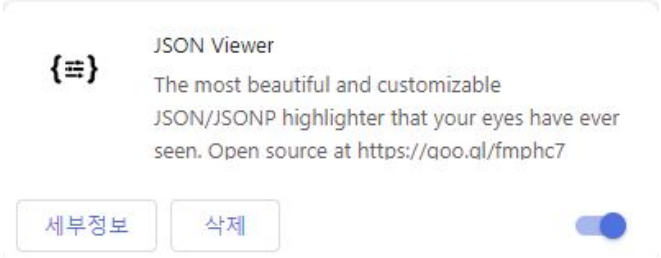
3.rescale_rest_api.py 설명

- **3. upload_local_files**
- rescale_platform, my_token을 이용해서 REST-API를 이용해 inputfile을 손쉽게 업로드 할 수 있습니다. 핵심은 앞서 압축했던 rescale.tar.gz파일을 Rescale files로 업로드하고 file id를 호출하는 것입니다. 이를 inputfiles_list라는 list 변수로 받습니다. 코드상의 'decompress' 가 True로 되어 있는 부분은 업로드 시 압축을 풀어서 Rescale 작업의 인스턴스 볼륨에 $HOME/work나 $HOME/work/shared 하에 업로드 하게 될 것입니다. **주의할 점은 심볼릭 링크는 decompress가 정상적으로 반영되지 않습니다.** 입력 파일에 symbolic 링크 파일이 만일 있다면, 대체하시거나, decompress 를 False로 변경하고 작업 command 라인에서 직접 압축을 풀어야 합니다.
- 정상적으로 업로드가 이뤄지면 다음과 같이 산출됩니다.
  - rescale.tar.gz uploaded
  All files uploaded successfully!
  inputfiles_list = [{'id': 'dxGonb', 'decompress': True}]

3.rescale_rest_api.py 설명

- **4. job_setup**
- Rescale 작업 제출에 필요한 JSON을 만들고 이를 이용해서 작업을 만든 후 뒤의 job_submit으로 작업을 제출합니다. 핵심적으로 지정해야 할 부분을 함수로 만들었습니다. 다음과 같습니다.
- **작업 이름, 프로젝트 id, envVars에서 선언되는 라이센스 정보, 작업 커맨드, SW tile에 해당하는 코드 네임, 코드 버전, coretype의 code, 코어 갯수, slot, walltime, inputfile**
- 매우 복잡하게 느껴지며, 바로 알 수 있는 것도 아닙니다. 특히 문제가 되는 부분은 ***SW tile에 해당하는 코드 네임, 코드 버전, coretype의 code, envVars에서 선언되는 라이센스 정보*** 은 실제 작업의 JSON를 직접 확인해야 알 수 있습니다. 그리고 작업 커맨드가 정상적으로 실행되는지도 판단해야 합니다.
- 가장 좋은 가이드라인은 처음에 Web platform을 이용해서 간단하게 작업을 구성하고 실행해 봅니다. Command가 정상적으로 작동하면 이제 해당 작업을 자동화하면 됩니다. 앞서 말씀드린 SW tile에 해당하는 코드 네임, 코드 버전, coretype의 code 정보는 정상적으로 구동된 작업의 JSON 정보를 보면 알 수 있습니다.
- https://kr.rescale.com/jobs/<job_id>/status 로 정상적인 작업의 status를 볼 수 있습니다. JSON 정보를 보려면 다음과 같이 입력합니다. **https://kr.rescale.com/api/v2/jobs/<job_id>**
- 정상적으로 수행되면 다음과 같이 작업 ID가 산출됩니다.
  Job_ID: LFZphb

3.rescale_rest_api.py 설명



- **4. job_setup**
- https://kr.rescale.com/api/v2/jobs/<job_id> 예시
- 편의를 위해서 web browser에 **JSON viewer 같은 확장 프로그램을 설치**하시면 도움이 됩니다.

```
38 ▾  "jobanalyses": [
39 ▾    {

51 ▾      "analysis": {
52          "code": "abaqus",
53          "name": "Abaqus",
54          "version": "2022-2241",

65          "slots": 1,
66          "coresPerSlot": 1,
67          "coreType": "starlite_max",
```

SW tile에 해당하는 코드 네임

코드 버전

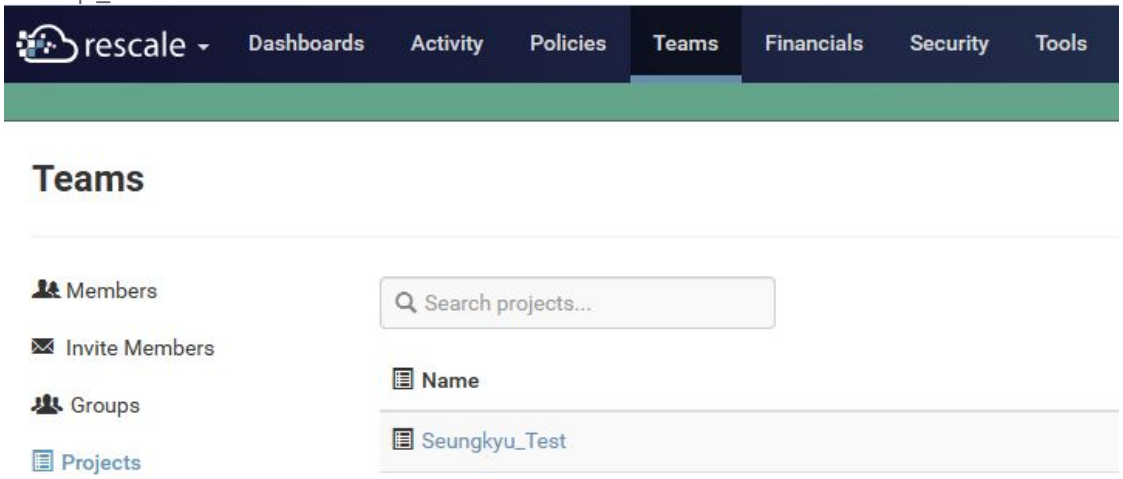
DOE가 아닌 이상 배치에서는 무조건 1입니다

허용하는 core갯수가 어떤 식인지 확인이 필요합니다. 웹 플랫폼에서 바로 확인 가능합니다.

coretype의 code

3.rescale_rest_api.py 설명

- **4. job_setup**
- 어떻게 project id를 알 수 있나요?
  → 관리자 페이지에서 Teams→ project →해당 project를 클릭하면 뜨는 url에 마지막 문자열이 해당 프로젝트 id 입니다.



'08-658737582/projects/xDdRk/

Seungkyu_Test는 xDdRk 가 project ID 입니다.

3.rescale_rest_api.py 설명

- **5.job_submit**
- 기존에 job_setup으로 작업을 저장하고, 해당 작업을 제출 하는 함수입니다.
- 기본적으로 job_setup이 올바르게 되었다면, 문제없이 수행됩니다. 편의를 위해서 2개의 파일이 생성됩니다.
  1. job_name.job 파일 (Rescale 작업 ID를 담고 있습니다)
  2. job_name.desktop (Rescale 작업 결과를 볼 수 있는 page로 이동하는 link가 표시됩니다)
- 작업이 올바르게 제출되면 다음과 같이 산출됩니다.
  Job LFZphb : submitted

3.rescale_rest_api.py 설명

- **6. job_monitor**
- 작업을 submit 하고 결과 파일을 추후에 확인한다고 하면 문제가 복잡하지 않겠지만 (스크립트가 여기서 끝나게 될것입니다), 기본적으로 자동화 할 때 작업이 정상적으로 돌아가는지 실시간으로 확인하고, 또한 작업이 끝나면 결과 파일을 다운로드 하시고 싶어합니다. 이를 위해서 python 스크립트를 Rescale 작업이 실행되는 동안 계속 켜 놓아야 하며, 이를 위해서 **screen 과 같은 가상 터미널 프로그램을 이용하시는 것을 매우 권장해 드립니다**.
- 본 함수는 5초마다 작업 status를 확인하고 업데이트 되면 변경사항을 표출합니다. 그리고 실행되는 동안 process_output.log의 결과를 15초 마다 20 라인씩 표출합니다. 이 작업은 Rescale 제출한 작업이 끝날때까지 계속 됩니다.
- 다음과 같이 표출될 것으로 보입니다.
  Job LFZphb : Pending
  Job LFZphb : Queued
  Job LFZphb : Validated
  Job LFZphb : Executing
  이 이후로는process_output.log가 표출됨

3.rescale_rest_api.py 설명

- **7. job_download**
- 앞서 job_monitor에서 작업이 종료된 것을 확인하면, 해당 job_name과 job_id를 이용해서 작업 파일을 내려받게 됩니다. 역시 내려받는 파일도 rescale.tar.gz이며, 이 파일은 job_name의 폴더가 python main.py 실행경로에 생겨나게 됩니다.
- 제대로 실행되면 다음과 같이 출력됩니다.
  Total 1 files will be downloaded
  1 user/user_xMTMk/output/job_qTfDW/run1/rescale.tar.gz downloaded
  Total 1 files, 93.345 MB downloaded

3.rescale_rest_api.py 설명

- **8. file_previous_job**
- Dependency가 있는 작업을 제출하기 위해 처음에 3. upload_local_files 에서처럼 파일을 local에서 업로드하는 것이 아닌, Rescale 작업에 수반된 Rescale files에 있는 파일을 다른 작업의 input 파일로 바로 업로드하게 만듭니다. 결국 중요한 정보는 해당되는 rescale.tar.gz의 files id를 list로 받아오는 것입니다. 7. job_download 과 3. upload_local_files 을 섞어서 만들어진 코드입니다.
- 성공적으로 파일 id 정보를 가져오면 다음과 같이 출력됩니다.
  inputfiles_list = [{'id': 'JKNBbb', 'decompress': True}]

4.사용자가 main.py에서 변경해야 할 요소

- 정해야 할 요소는 다음과 같습니다.
- **batch_name**: 원하는 dependency 가 있는 작업 만큼 만들 수 있습니다. batch 작업의 이름을 정해 주십시오.
- **command**: 각 batch 작업에 해당하는 command 라인이 필요합니다. 맨 뒤에 \n을 적용해서 다음 행으로 캐리지 리턴이 될 수 있도록 해 주십시오. 앞서 말씀드린 대로 command는 기본적으로 web platform을 이용해서 간단한 작업이 수행되는지 확인 후, 본격적으로 자동화 할 command 라인에 넣어 주시면 됩니다.
- code_name, version_code, license_info, coretype_code, core_per_slot, walltime의 경우 관리자가 정할 수도 있습니다. 확인이 어려운 code_name, version_code, coretype_code, license_info는 작업을 하나 만들고 https://kr.rescale.com/api/v2/jobs/<job_id> 로 확인한다고 말씀드렸습니다. Walltime의 경우 사용자가 조정을 할 수도 있다고 봅니다.
- **Dependence가 있는 작업의 경우 예제로 드리는 main.py의 3.1~6.1에서 나타나듯이 inputfiles_list, job_id, job_name, command, 에 주의하면서 작성합니다. Dependency가 있는 작업이 늘어나면, 그만큼 반복해서 작성합니다.**

```
    # iterate as you want from 3.1 to 6.1 for dependent job
    # 3.1 Get the ID of input file in Rescale files (AWS S3) from Previous Rescale Job
    inputfiles_list2 = rescale.file_previous_job(rescale_platform, my_token, job_id1)
    # 4.1 Rescale Job Configuration
    job_id2 = rescale.job_setup(rescale_platform, my_token, job_name2, command2, code_name, version_code, license_info, coretype_code, core_per_slot,
slot, walltime, projectid, inputfiles_list2)
    # 5.1 Rescale Job Submit
    rescale.job_submit(rescale_platform, my_token, job_name2, job_id2)
    # 6.1 Rescale Job Moinitor
    rescale.job_monitor(rescale_platform, my_token, job_id2)
```

5.실행 방법

- 1. Python 가상 환경을 만들어 줍니다.
  python3 -m venv 가상환경이름
- 2. 가상환경이름 폴더로 들어가서 가상환경을 활성화 해 줍니다.
  cd 가상환경이름/bin
  source activate
  → 관련해서 프롬프트 맨 앞에 (가상환경이름) 이 붙으면 성공입니다.
- 3. requests와 requests_toolbelt를 설치합니다.
  pip install requests
  pip install requests_toolbelt
- 4. main.py를 수정합니다.
- 5. screen 과 같은 가상터미널 환경을 실행합니다.
  screen -S 세션명
- 5. main.py 를 실행합니다.
  python main.py
- 6. Screen 세션을 보고 있을 필요가 없을 경우 detach
  Ctrl + a + d (컨트롤 키 누른 상태에서 순서대로 a 그리고 d를 누른다)

5. 실행 방법

- **screen 실행 명령어**
  - 세션 실행
    screen -S 세션명
  - 세션 확인
    screen -list
  - 세션 detach
    Ctrl + a + d (컨트롤 키 누른 상태에서 순서대로 a 그리고 d를 누른다)
  - 세션 재진입
    screen -r 세션명
  - 세션 삭제
    screen -X -S 세션id quit

High Performance Computing Built for the Cloud

- Digital Engineering
- Workload Optimization
- Intelligent Automation
- Security & Compliance

Become a Rescale Certified User Expert. Learn more at rescale.com/training