Backup & Restore Elasticsearch

<https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html>

1. Tổng quan:

Hoàn thiện chương trình được viết bằng Python3:

* Backup, restore với vòng lặp.
* Backup hằng ngày, rotate 10 ngày 1 lần.

Giải quyết được bài toán Restore Elasticsearch bằng các bước:

* Take snapshot all indices in Elasticsearch (bước này được thực hiện mỗi ngày, đảm bảo luôn sẵn sàng trước khi xảy ra sự cố).
* Xóa tất cả indices cũ.
* Restore tất cả indices từ snapshot mới nhất (ngoại trừ indices có tên là “searchguard” – Search Guard không cho phép restore indice này) và đặt tên mới với cú pháp: <snapshot\_name>\_restored\_<indice\_name>.
* Rename các indice mới ở trên thành tên của các indice cũ.
* Xóa các indices có tên bắt đầu bằng “<snapshot\_name>\_restored”

Chi tiết quá trình thực hiện được viết trong file:

1. Backup

Mỗi lần take snapshot thì snapshot đó sẽ thuộc một repository. Repository nằm trong "path.repo" được cấu hình trong elasticsearch.yml.

**Repositories**

Tạo thư mục chứa dữ liệu backup

mkdir /elastic\_backup

Cấp quyền thư mục backup

chown -R elasticsearch. /elastic\_backup

Sửa file cấu hình elasticsearch.yml:

path.repo: ["/elastic\_backup"]

Khai báo repository:

*Nếu dùng Console trên Kibana:*

|  |
| --- |
| PUT /\_snapshot/my\_backup  {  "type": "fs",  "settings": {  "location": "/elastic\_backup",  "compress": true  }  } |

*Nếu dùng command line:*

curl -XPUT "https://localhost:9200/\_snapshot/my\_backup" -H 'Content-Type: application/json' -d'

{

"type": "fs",

"settings": {

"location": "/elastic\_backup",

"compress": true

}

}'

Nếu đã cài searchguard thì lệnh curl thêm 2 options:

curl -u admin:admin -XGET "…" --insecure

**Snapshot**

Cách take snapshot:

*Dùng Kibana Console:*

|  |
| --- |
| PUT /\_snapshot/my\_backup/snapshot\_1?wait\_for\_completion=true |

*Dùng command line:*

curl -XPUT "https://localhost:9200/\_snapshot/my\_backup/snapshot\_1?wait\_for\_completion=true"

* Dùng PUT
* my\_backup là tên của repository.
* snapshot\_1 là tên của snapshot, không được trùng nhau.
* wait\_for\_completion=true: màn hình output sẽ đợi cho đến khi take snapshot xong.

Mặc định snapshot sẽ chứa tất cả indices, nếu chỉ muốn backup một số indices thì dùng:

*Console:*

|  |
| --- |
| PUT /\_snapshot/my\_backup/snapshot\_3?wait\_for\_completion=true  {  "indices": "index\_1, filebeat-2018.06.04",  "ignore\_unavailable": true,  "include\_global\_state": false  } |

*Command line:*

curl -XPUT "https://localhost:9200/\_snapshot/my\_backup/snapshot\_3?wait\_for\_completion=true" -H 'Content-Type: application/json' -d'

{

"indices": "index\_1, filebeat-2018.06.04",

"ignore\_unavailable": true,

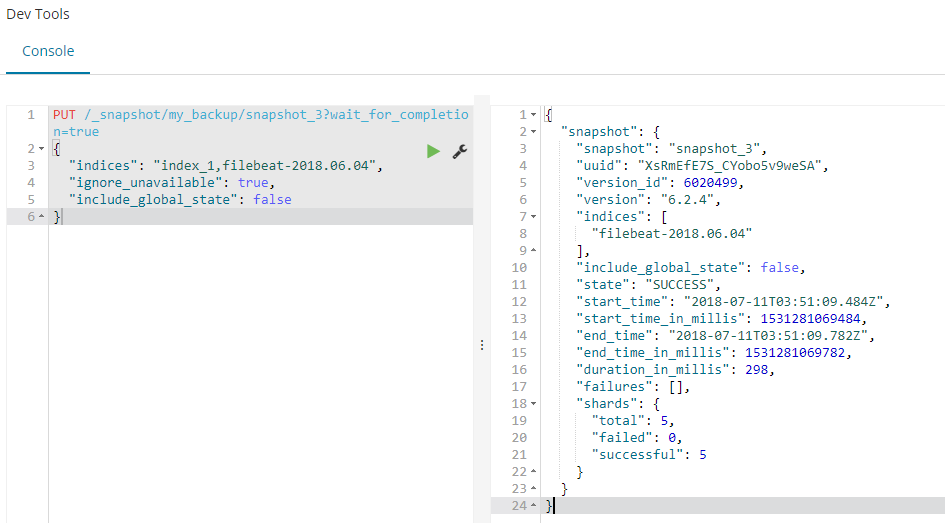
"include\_global\_state": false

}'

Giải thích:

* indices: list các indices, phân cách với nhau bởi dấu phẩy.
* "ignore\_unavailable": true : Nếu không tìm thấy 1 indice nào đó trong list thì sẽ bỏ qua và thực hiện snapshot indice tiếp theo. VD: Không tìm thấy “index\_1” thì bỏ qua, nếu không có option này thì trường hợp này take snapshot fail.

Kết quả trả về:

**

Kiểm tra trạng thái của Snapshots:

Liệt kê các repositories:

|  |
| --- |
| GET \_snapshot |

curl -XGET "https://localhost:9200/\_snapshot"

curl -u admin:admin -XGET "https://localhost:9200/\_snapshot?pretty" --insecure

Xem các tiến trình snapshot đang chạy:

|  |
| --- |
| GET /\_snapshot/my\_backup/\_current |

curl -XGET "https://localhost:9200/\_snapshot/my\_backup/\_current"

Kiểm tra 1 snapshot:

|  |
| --- |
| GET /\_snapshot/my\_backup/snapshot\_1 |

curl -XGET "https://localhost:9200/\_snapshot/my\_backup/snapshot\_1"

Kiểm tra tất cả snapshot

|  |
| --- |
| GET /\_snapshot/my\_backup/\_all |

curl -XGET "https://localhost:9200/\_snapshot/my\_backup/\_all"

Ví dụ trạng thái snapshot:

**

Trạng thái thường gặp:

|  |  |
| --- | --- |
| IN\_PROGRESS | The snapshot is currently running. |
| SUCCESS | The snapshot finished and all shards were stored successfully. |
| FAILED | The snapshot finished with an error and failed to store any data. |

1. Restore

Để user của search\_guard có quyền restore, ta phải cấu hình:

vim /etc/elasticsearch/elasticsearch.yml

Đảm bảo dòng sau có giá trị là true:

|  |
| --- |
| searchguard.enable\_snapshot\_restore\_privilege: true |

Restore mặc định:

|  |
| --- |
| POST /\_snapshot/my\_backup/snapshot\_1/\_restore |

curl -XPOST "https://localhost:9200/\_snapshot/my\_backup/snapshot\_1/\_restore"

Restore với tùy biến (đổi tên indices sau khi restore):

|  |
| --- |
| POST /\_snapshot/my\_backup/snapshot\_1/\_restore  {  "indices": "filebeat-\*",  "ignore\_unavailable": true,  "include\_global\_state": false,  "rename\_pattern": "filebeat-(.+)",  "rename\_replacement": "restored\_index\_$1"  } |

curl -XPOST "https://localhost:9200/\_snapshot/my\_backup/snapshot\_1/\_restore" -H 'Content-Type: application/json' -d'

{

"indices": "filebeat-\*",

"ignore\_unavailable": true,

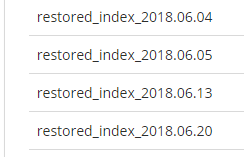
"include\_global\_state": false,

"rename\_pattern": "filebeat-(.+)",

"rename\_replacement": "restored\_index\_$1"

}'

Kết quả:

**

Xóa một tiến trình Snapshot / Restore đã / đang chạy:

|  |
| --- |
| DELETE /\_snapshot/my\_backup/snapshot\_1 |

curl -XDELETE "https://localhost:9200/\_snapshot/my\_backup/snapshot\_1"

1. Quy trình thực hiện
   1. Chạy hằng ngày với cronjob

* Cài đặt Python3 trên máy chạy code
* Cài đặt gói pip3 (nếu chưa cài):

sudo yum install python36-setuptools

yum -y install python36-devel

sudo easy\_install-3.6 pip

* Cài thêm một số thư viện của Python3 :

pip3 install PyYAML

pip3 install requests

* Tạo thư mục chứa dữ liệu backup
* Cấp quyền thư mục backup
* Sửa file cấu hình elasticsearch.yml
* Khởi động lại Elasticsearch
* Khai báo repository
* Đưa các file code và file cấu hình vào chung 1 thư mục, các file này gồm có: *elasticsearch\_backup.py, MyLogger.py, credentials.yaml, elasticsearch\_backup\_config.yaml*
* Sửa file elasticsearch\_backup.py :

*Sửa đường dẫn đến file backup log tại dòng 13 (đường dẫn tuyệt đối)*

*Sửa dòng 14: thư mục chứa file config (đường dẫn tuyệt đối)*

* Sửa cấu hình ở các file \*.yaml
* Đưa vào crontab:

crontab -e

0 0 \* \* \* /usr/bin/python36 /etc/elasticsearch/scripts/elasticsearch\_backup.py > /dev/null 2>&1

* 1. Chạy thủ công
* Thực hiện các bước giống như mục 4.1
* Cách sử dụng file thủ công để:
  + Liệt kê tất cả snapshot trong repo:

python3.6 elasticsearch\_backup.py --list-all-snapshots

* + Xem 1 snapshot chứa những gì:

python3.6 elasticsearch\_backup.py --check-one-snapshot <snapshot\_name>

*Ví dụ:*

python3.6 elasticsearch\_backup.py --check-one-snapshot snapshot-2018-08-24

* + Xóa một snapshot:

python3.6 elasticsearch\_backup.py --delete-one-snapshot snapshot-2018-08-24

* + Take snapshot (nên đặt tên theo định dạng này):

python3.6 elasticsearch\_backup.py --take-snapshot *snapshot-2018-08-24*

* 1. Restore khi gặp sự cố
* Các bước cấu hình ban đầu giống mục 4.1
* Sửa file elasticsearch\_restore.py :
  + Sửa đường dẫn tại dòng 11 và 12
  + Sửa tên snapshot mà mình chọn để restore, tại dòng 225
  + Thực thi:

python36 /etc/elasticsearch/scripts/elasticsearch\_restore.py

1. Phụ lục
   1. Chương trình Elasticsearch backup:

File *elasticsearch\_backup.py* :

|  |
| --- |
| import json import os import requests import sys import yaml  from MyLogger import MyLogger  from \_datetime import datetime from requests.auth import HTTPBasicAuth   backup\_log\_path = '/etc/elasticsearch/scripts/backup\_elasticsearch.log' config\_dir = '/etc/elasticsearch/scripts/' my\_log = MyLogger(backup\_log\_path)   def load\_config(config\_file\_path):  with open(config\_file\_path, 'r') as stream:  data = yaml.load(stream)  return data   def take\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard):  querystring = {  'pretty': ''  }  headers = {  'content-type': "application/json"  }  payload = {  #"indices": "filebeat-\*",  "ignore\_unavailable": True,  "include\_global\_state": False  }   try:  my\_log.info('Taking snapshot: {} ...'.format(snapshot\_name))  print('Taking snapshot: {} ...'.format(snapshot\_name))  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_snapshot/{}/{}?wait\_for\_completion=true' \  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }   response = requests.request("PUT",  url,  headers=headers,  data=json.dumps(payload),  params=querystring,  verify=False,  auth=HTTPBasicAuth(user, password))   else:  url = 'http://{}/\_snapshot/{}/{}?wait\_for\_completion=true'\  .format(hostport, repo\_name, snapshot\_name)   response = requests.request("PUT",  url,  headers=headers,  data=json.dumps(payload),  params=querystring,  verify=False)  if response.status\_code == 200:  my\_log.info('Take snapshot done! Snapshot name: {}'.format(snapshot\_name))  print('Take snapshot done! Snapshot name: {}'.format(snapshot\_name))  else:  my\_log.error('Take snapshot failed! '  'The snapshot has not been created yet. Detail:\n{}'  .format(response.text))  print('Take snapshot failed! '  'The snapshot has not been created yet. Detail:\n{}'  .format(response.text))   except Exception:  my\_log.error('Take snapshot failed! Something went wrong.')   def check\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard):  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_snapshot/{}/{}' \  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  response = requests.request("GET", url, params=querystring, verify=False,  auth=HTTPBasicAuth(user, password))   else:  url = 'http://{}/\_snapshot/{}/{}'\  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  response = requests.request("GET", url, params=querystring, verify=False)  print(response.text)   def get\_all\_snapshot\_names(hostport, repo\_name, use\_searchguard):  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_snapshot/{}/\_all'\  .format(hostport, repo\_name)  querystring = {  'pretty': ''  }  response = requests.request("GET", url, params=querystring, verify=False,  auth=HTTPBasicAuth(user, password))  else:  url = 'http://{}/\_snapshot/{}/\_all' \  .format(hostport, repo\_name)  querystring = {  'pretty': ''  }  response = requests.request("GET", url, params=querystring, verify=False)   data = json.loads(response.text)   result = []   for snapshot in data['snapshots']:  result.append(snapshot['snapshot'])   return result   def delete\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard):  querystring = {  'pretty': ''  }   try:  my\_log.info('Deleting snapshot: {} ...'.format(snapshot\_name))  print('Deleting snapshot: {} ...'.format(snapshot\_name))  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_snapshot/{}/{}' \  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  response = requests.request("DELETE", url, params=querystring, verify=False,  auth=HTTPBasicAuth(user, password))  else:  url = 'http://{}/\_snapshot/{}/{}'\  .format(hostport, repo\_name, snapshot\_name)  response = requests.request("DELETE", url, params=querystring, verify=False)   if response.status\_code == 200:  my\_log.info('Snapshot was be deleted: {}'.format(snapshot\_name))  print('Snapshot was be deleted: {}'.format(snapshot\_name))  else:  my\_log.error('Delete snapshot failed! Detail:\n{}'  .format(response.text))  print('Delete snapshot failed! Detail:\n{}'  .format(response.text))   except Exception:  my\_log.error('Delete snapshot failed ! Something went wrong.')   def rotate\_snapshot(hostport, repo\_name, limit, use\_searchguard):  snapshots = get\_all\_snapshot\_names(hostport, repo\_name, use\_searchguard)   if len(snapshots) - limit > 0:  my\_log.info('Rotating snapshot...')  for snapshot in snapshots[:len(snapshots) - limit]:  delete\_snapshot(hostport, repo\_name, snapshot, use\_searchguard)   def main():  # Ignore warnings:  import warnings  warnings.simplefilter("ignore")   args = sys.argv  config\_file\_path = '{}/elasticsearch\_backup\_config.yaml'.format(config\_dir)  config = load\_config(config\_file\_path)   hostport = config['HOSTPORT']  repo\_name = config['REPO']  limit = config['LIMIT'] # So ngay rotate snapshot  use\_searchguard = config['USE\_SEARCHGUARD']   os.environ['NO\_PROXY'] = hostport  if len(args) == 1:  snapshot\_name = 'snapshot-' + str(datetime.date(datetime.now()))  take\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard)  rotate\_snapshot(hostport, repo\_name, limit, use\_searchguard)   else:  if '--list-all-snapshots' in args:  print(get\_all\_snapshot\_names(hostport, repo\_name, use\_searchguard))  elif '--check-one-snapshot' in args:  if len(args) < 3:  print('Missing snapshot name !')  sys.exit(1)  else:  check\_snapshot(hostport, repo\_name, args[2], use\_searchguard)  elif '--delete-one-snapshot' in args:  if len(args) < 3:  print('Missing snapshot name !')  sys.exit(1)  else:  delete\_snapshot(hostport, repo\_name, args[2], use\_searchguard)  elif '--take-snapshot' in args:  if len(args) < 3:  print('Missing snapshot name !')  sys.exit(1)  else:  take\_snapshot(hostport, repo\_name, args[2], use\_searchguard)  else:  print('Invalid argument !')   if \_\_name\_\_ == '\_\_main\_\_':  main() |

*Nếu chỉ muốn take snapshot 1 số indices nhất định thì tại dòng 32, xóa comment:*

*"indices": "filebeat-\*",*

*Chỉ take snapshot các indices bắt đầu bằng “filebeat”.*

File MyLogger.py :

|  |
| --- |
| import logging   class MyLogger:  logger = logging.getLogger(\_\_name\_\_)  logger.setLevel(logging.DEBUG)   def \_\_init\_\_(self, filepath):  self.filepath = filepath   # create a file handler  handler = logging.FileHandler(self.filepath)  handler.setLevel(logging.DEBUG)   # create a logging format  formatter = logging.Formatter('%(asctime)s '  '- %(levelname)s - %(message)s')  handler.setFormatter(formatter)   # add the handlers to the logger  self.logger.addHandler(handler)   def error(self, message):  self.logger.error(message)   def warning(self, message):  self.logger.warning(message)   def info(self, message):  self.logger.info(message) |

File *credentials.yaml* chứa username và password quyền admin của searchguard:

|  |
| --- |
| **user:** admin **password:** admin |

File *elasticsearch\_backup\_config.yaml* :

|  |
| --- |
| **HOSTPORT:** 172.30.27.23:9200 **LIMIT:** 10 *# So luong snapshots toi da (so ngay)* **REPO:** repo3 **USE\_SEARCHGUARD:** 1 |

Giải thích:

* HOSTPORT: hostport kết nối và elasticsearch.
* LIMIT: Giới hạn số lượng snapshot trong repo (số ngày).
* REPO: tên repository.
* USE\_SEARCHGUARD: 1 nếu đang dùng searchguard, 0 nếu không dùng.
  1. Chương trình Elasticsearch restore:

File *elasticsearch\_restore.py*

|  |
| --- |
| import json import os import re import requests import yaml  from MyLogger import MyLogger from requests.auth import HTTPBasicAuth   restore\_path = '/etc/elasticsearch/scripts/restore\_elasticsearch.log' config\_dir = '/etc/elasticsearch/scripts' my\_log = MyLogger(restore\_path)   def load\_config(config\_file\_path):  with open(config\_file\_path, 'r') as stream:  data = yaml.load(stream)  return data   def check\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard):  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_snapshot/{}/{}' \  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  response = requests.request("GET", url, params=querystring, verify=False,  auth=HTTPBasicAuth(user, password))   else:  url = 'http://{}/\_snapshot/{}/{}'\  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  response = requests.request("GET", url, params=querystring, verify=False)  return response.text   def delete\_snapshot(hostport, repo\_name, snapshot\_name):  url = 'https://admin:admin@{}/\_snapshot/{}/{}'\  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  response = requests.request("DELETE", url, params=querystring, verify=False)  print(response.text)   def restore\_indice(hostport, repo\_name, snapshot\_name, indice\_name, use\_searchguard):  try:  my\_log.info('Restoring indice: {} ...'.format(indice\_name))  if snapshot\_name:  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_snapshot/{}/{}/\_restore?wait\_for\_completion=true' \  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  headers = {  'content-type': "application/json"  }  payload = {  "indices": indice\_name,  "ignore\_unavailable": False,  "include\_global\_state": False,  # "rename\_pattern": "(.+)",  # "rename\_replacement": snapshot\_name + "\_restored\_$1"}  }  response = requests.request("POST",  url,  headers=headers,  data=json.dumps(payload),  params=querystring,  verify=False,  auth=HTTPBasicAuth(user, password))   else:  url = 'http://{}/\_snapshot/{}/{}/\_restore?wait\_for\_completion=true' \  .format(hostport, repo\_name, snapshot\_name)  querystring = {  'pretty': ''  }  headers = {  'content-type': "application/json"  }  payload = {  "indices": indice\_name,  "ignore\_unavailable": False,  "include\_global\_state": False  }  response = requests.request("POST",  url,  headers=headers,  data=json.dumps(payload),  params=querystring,  verify=False)   if response.status\_code == 200:  my\_log.info('Restore indice done! Indice name: {}'.format(indice\_name))  print('Restore indice done! Indice name: {}'.format(indice\_name))  else:  my\_log.error('Restore indice failed! Detail:\n{}'  .format(response.text))  print('Restore indice failed! Detail:\n{}'.format(response.text))   except Exception:  my\_log.error('Restore indice failed! Something went wrong.')   def get\_all\_indice\_names\_in\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard):  data = check\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard)  data = json.loads(data)  return data['snapshots'][0]['indices']   def get\_all\_indice\_names\_on\_cluster(hostport, use\_searchguard):  if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/\_cat/indices' \  .format(hostport)  response = requests.request("GET",  url,  verify=False,  auth=HTTPBasicAuth(user, password))   else:  url = 'http://{}/\_cat/indices' \  .format(hostport)  response = requests.request("GET",  url,  verify=False)  data = response.text  result = []  for line in data.splitlines():  result.append(line.split()[2])  return result   def delete\_indice(hostport, indice\_name, use\_searchguard):  try:  my\_log.info('Deleting old indice: {} ...'.format(indice\_name))  print('Deleting old indice: {} ...'.format(indice\_name))   if use\_searchguard:  data = load\_config('{}/credentials.yaml'.format(config\_dir))  user = data['user']  password = data['password']  url = 'https://{}/{}' \  .format(hostport, indice\_name)   response = requests.request("DELETE",  url,  verify=False,  auth=HTTPBasicAuth(user, password))   else:  url = 'http://{}/{}' \  .format(hostport, indice\_name)  response = requests.request("DELETE",  url,  verify=False)   if response.status\_code == 200:  my\_log.info('Old indice was be deleted: {}'.format(indice\_name))  print('Old indice was be deleted: {}'.format(indice\_name))   else:  my\_log.error('Delete indice failed! Detail:\n{}'  .format(response.text))  print('Delete indice failed! Detail:\n{}'.format(response.text))   return response.text   except Exception:  my\_log.error('Delete indice failed! Something went wrong.')  def rename\_indice(hostport, indice\_name):  url = 'https://admin:admin@{}/\_reindex' \  .format(hostport)  headers = {  'content-type': "application/json"  }  new\_name = re.findall('restored\_(.+)', indice\_name)[0]  print('Renaming ' + new\_name + '...')  payload = {  "source": {  "index": indice\_name  },  "dest": {  "index": new\_name  }  }  response = requests.request("POST",  url,  headers=headers,  data=json.dumps(payload),  verify=False)  return response.text   def main():  # Ignore warnings:  import warnings  warnings.simplefilter("ignore")   config\_file\_path = '{}/elasticsearch\_backup\_config.yaml'.format(config\_dir)  config = load\_config(config\_file\_path)   hostport = config['HOSTPORT']  repo\_name = config['REPO']  use\_searchguard = config['USE\_SEARCHGUARD']  os.environ['NO\_PROXY'] = hostport   snapshot\_name = 'snapshot-2018-08-27'   indices = get\_all\_indice\_names\_in\_snapshot(hostport, repo\_name, snapshot\_name, use\_searchguard)  my\_log.info('-' \* 60)  my\_log.info('Start restoring snapshot: {} ...'.format(snapshot\_name))  my\_log.info('-' \* 60)  print('Start restoring snapshot: {} ...'.format(snapshot\_name))   for indice in indices:  delete\_indice(hostport, indice, use\_searchguard)  restore\_indice(hostport, repo\_name, snapshot\_name, indice, use\_searchguard)   if \_\_name\_\_ == '\_\_main\_\_':  main() |