**通过Lambda以及EventBridge自动为Opensearch拍快照**

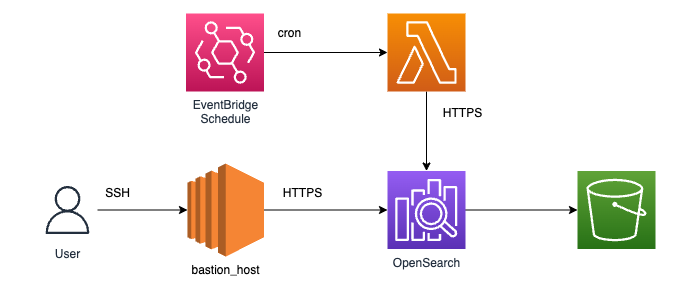
**背景**

[OpenSearch](https://opensearch.org/) 是一种分布式，由社区驱动并取得 Apache 2.0 许可的 100% 开源搜索和分析系统，被广泛应用于实时应用程序监控、日志分析和网站搜索等场景。Amazon OpenSearch Service是OpenSearchAWS云上的托管版本（以下简称”托管版“）。”托管版“提供两种类型的快照：其一，自动快照(Automated snapshots)，用于实现其集群节点出现故障后的恢复；其二，手工快照(Manual snapshots)，用于保证数据的安全，例如在索引被误删的情况下，系统管理员能够根据快照恢复系统。

由于目前“托管版”尚未支持类似开源版Snapshot Management的功能，手工快照的配置以及调用存在一定的复杂性，以及缺少自动化执行的支持，因此客户需要一种自动化执行“手工快照”的解决方案来解决上述运维痛点。

本文阐述一种Lambda+EventBridge的解决方案以及方案的实现细节。

**解决方案架构**



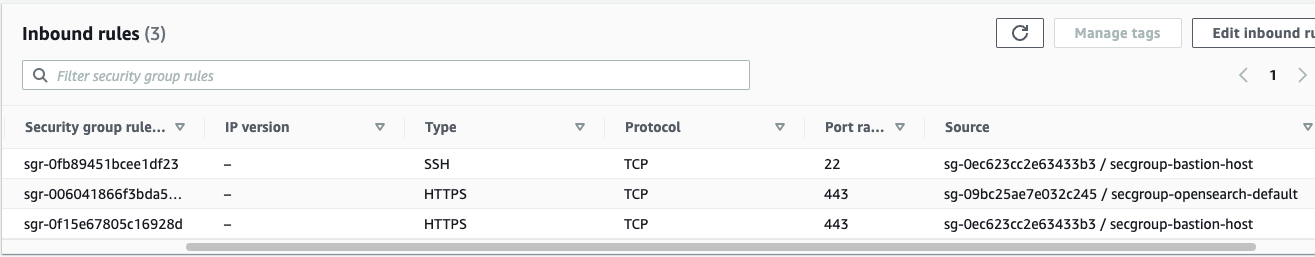
**1. Setup Bastion Host**

1. Create a new security group secgroup-bastion-host.
   * Add an inbound rule to allow TCP connection at port 22 from anywhere, or from your own IP CIDR. You may use <https://checkip.amazonaws.com/> to find your current IP address.
   * Leave outbound rules as it is. By default, it allows all outbound traffic to anywhere.
2. Launch an EC2 instance as a bastion-host.
   * Application and OS Images: Amazon Linux 2 AMI
   * Instance Type: Choose a small one, e.g. t2.micro
   * Key pair: Select an existing key pair
   * Network settings: Choose VPC, subnet, and the newly created security group secgroup-bastion-host.
   * Leave other configurations at their default values.
3. Launch the instance and take note of it's public IP address.
4. Test connection to bastion host using terminal or any other SSH client, e.g. Termius.
   * Using Terminal on Mac

 ssh -i <PEM\_KEY\_PATH> ec2-user@<BASTION\_IP>

**2. Setup OpenSearch in VPC**

1. Create a new security group secgroup-opensearch-default.
   * Add an inbound rule to allow SSH connection (TCP at port 22) from the security group of bastion host.
   * Add an inbound rule to allow HTTPS connection (TCP at port 443) from the security group of bastion host.
   * Add an inbound rule to allow HTTPS connection within the same security group, i.e. secgroup-opensearch-default.
   * Leave outbound rule which allows connections to anywhere.



1. Create a new OpenSeach Cluster.
   * For testing purpose, we choose minimum resourse to save cost.
     + Development type = **Development and testing**.
     + **Auto-Tune** = Disable
     + Availbility Zones = **1-AZ**
     + Instance Type = **t3.small.search**
     + Number of nodes = 1
   * Set Network to **VCP access**
     + Choose the **same** VPC, subnet as the bastion host.
     + Use the newly created security group secgroup-opensearch-default
   * Check **Enable fine-grained access control**
     + Select Create master user
   * Set **Access policy** to Only use fine-grained access control.
   * Leave other settings at default.
2. While domain is creating, double check on following settings.
   * Cluster configuration: VPC, Subnet, Security Group are configured correctly.
   * Security configuration: Fine-grained access control is enabled; Access policy is similar to folllowing.

 {  
   "Version": "2012-10-17",  
   "Statement": [  
    {  
       "Effect": "Allow",  
       "Principal": {  
         "AWS": "\*"  
      },  
       "Action": "es:\*",  
       "Resource": "arn:aws:es:ap-southeast-1:460453255610:domain/privatedomain/\*"  
    }  
  ]  
 }

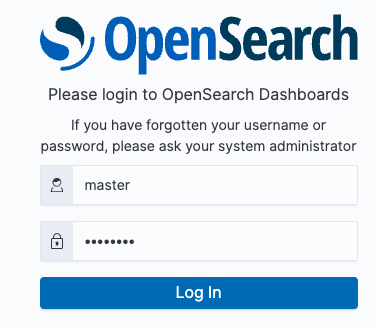
1. After domain is Active, take note of its Domain endpoint (VPC).
   * For domains within VPC, their endpoints have the format of vpc-\*-\*.\*.es.amazonaws.com.

**3. Test Port Forwarding**

1. Run following command to forward local port 9200 to port 443 of domain endpoint through the bastion host.
   * DOMAIN\_ENDPOINT: the domain endpoint of the opensearch cluster **without** https://
   * The command will not return any message.

 ssh -i <PEM\_KEY\_PATH> ec2-user@<BASTION\_IP> -N -L 9200:DOMAIN\_ENDPOINT:443

1. On web browser, go to OpenSearch Dashboards URL, which is the https://DOMAIN\_ENDPOINT/\_dashboards.
   * Login with username and password which we set during cluster creation.



**Using SSH Profile [Optional]**

Alternatively, you can create a SSH profile so that you don't need to remember the port forwarding command.

1. Edit SSH config file.

 vim ~/.ssh/config

1. At following profile estunnel above existing profiles.

 ########################  
 # ElasticSearch Tunnel  
 Host estunnel  
 HostName <BASTION\_HOST\_PUBLIC\_IP>  
 User ec2-user  
 IdentitiesOnly yes  
 IdentityFile <PATH\_TO\_PEM\_KEY>  
 LocalForward 9200 <DOMAIN\_ENDPOINT>:443

1. Run the SSH command with host ID, i.e. estunnel.

 ssh -N estunnel

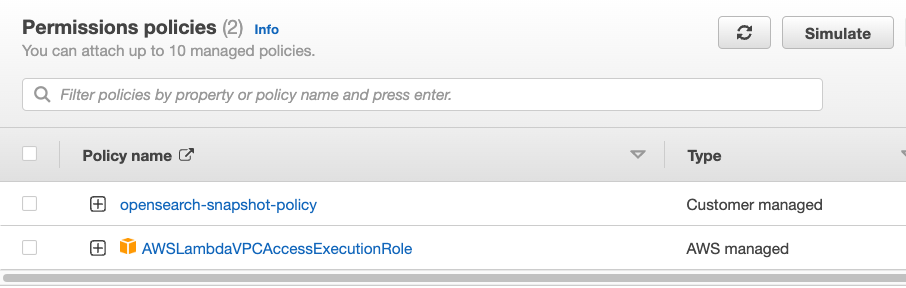
**4. Deploy Lambda Function to Create OpenSearch Snapshot**

1. 创建S3桶，以下用BUCKET\_NAME来表示。
2. 想好一个IAM角色的名字，以下用 IAM\_ROLE\_NAME来表示。
3. 创建一个IAM策略，例如 opensearch-snapshot-policy 。这个策略赋予Lambda函数连接OpenSearch服务的权限。
   * 替代REGION，ACCOUNT\_ID，BUCKET\_NAME，IAM\_ROLE\_NAME。

 {  
     "Version": "2012-10-17",  
     "Statement": [  
        {  
             "Sid": "VisualEditor0",  
             "Effect": "Allow",  
             "Action": [  
                 "iam:PassRole"  
            ],  
             "Resource": [  
                 "arn:aws:iam::ACCOUNT\_ID:role/IAM\_ROLE\_NAME"  
            ]  
        },  
        {  
             "Sid": "VisualEditor1",  
             "Effect": "Allow",  
             "Action": [  
                 "s3:ListBucket"  
            ],  
             "Resource": [  
                 "arn:aws:s3:::BUCKET\_NAME"  
            ]  
        },  
        {  
             "Sid": "VisualEditor2",  
             "Effect": "Allow",  
             "Action": [  
                 "s3:PutObject",  
                 "s3:GetObject",  
                 "s3:DeleteObject"  
            ],  
             "Resource": "arn:aws:s3:::BUCKET\_NAME/\*"  
        },  
        {  
             "Sid": "VisualEditor3",  
             "Effect": "Allow",  
             "Action": [  
                 "es:ESHttpPut",  
                 "es:ESHttpGet",  
                 "es:ESHttpPost",  
              "es:ESHttpDelete"  
            ],  
             "Resource": [  
                 "arn:aws:es:REGION:ACCOUNT\_ID:domain/\*"  
            ]  
        }  
    ]  
 }

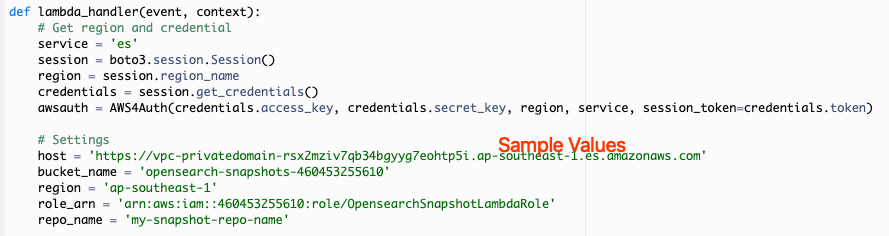
1. Create role OpensearchSnapshotLambdaRole with following trust relationship.
   * Add above policy opensearch-snapshot-policy and the AWS managed policyAWSLambdaVPCAccessExecutionRole to the role.

 {  
     "Version": "2012-10-17",  
     "Statement": [  
        {  
             "Effect": "Allow",  
             "Principal": {  
                 "Service": [  
                     "lambda.amazonaws.com",  
                     "ssm.amazonaws.com",  
                     "es.amazonaws.com"  
                ]  
            },  
             "Action": "sts:AssumeRole"  
        }  
    ]  
 }

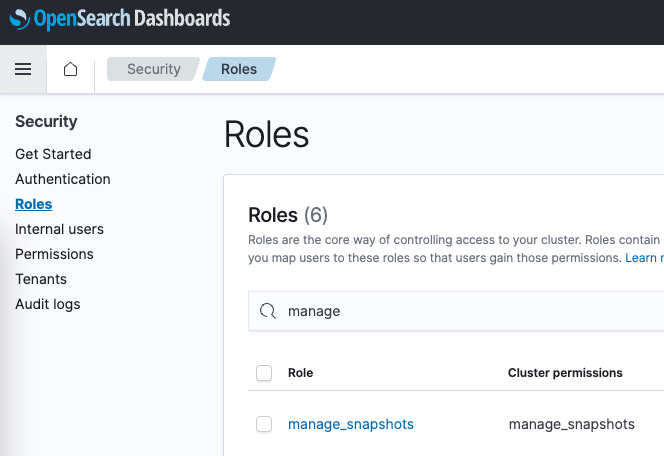


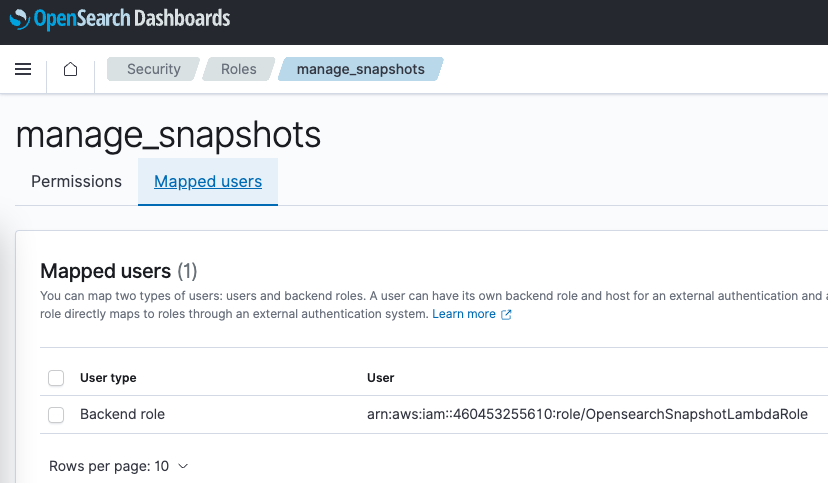
1. 创建一个Lambda函数，例如 BackupOpensearchSnapshots。选择以下配置。
   * 运行时库： Python 3.8
   * 执行角色： 刚创建的IAM角色，例如 OpensearchSnapshotLambdaRole
   * Choose same VPC, Subnet and Security Group (i.e. secgroup-opensearch-default) as OpenSearch domain.
2. Use following Python code.
   * Update <DOMAIN\_ENDPOINT\_WITH\_HTTPS>, <BUCKET\_NAME>, <AWS\_REGION>, <ARN\_OF\_IAM\_ROLE\_OpensearchSnapshotLambdaRole> and <REPOSITORY\_NAME> accordingly.

 import json  
 from typing import Dict, Tuple  
 ​  
 import boto3  
 import requests  
 from requests\_aws4auth import AWS4Auth  
 from datetime import datetime  
 ​  
 ​  
 def lambda\_handler(event, context):  
     # Get region and credential  
     service = 'es'  
     session = boto3.session.Session()  
     region = session.region\_name  
     credentials = session.get\_credentials()  
     awsauth = AWS4Auth(credentials.access\_key, credentials.secret\_key, region, service, session\_token=credentials.token)  
 ​  
     # Settings  
     host = '<DOMAIN\_ENDPOINT\_WITH\_HTTPS>'  
     bucket\_name = '<BUCKET\_NAME>'  
     region = '<AWS\_REGION>'  
     role\_arn = '<ARN\_OF\_IAM\_ROLE\_OpensearchSnapshotLambdaRole>'  
     repo\_name = '<REPOSITORY\_NAME>'  
 ​  
     # # Register repository  
     # register\_repository(host, awsauth, repo\_name, bucket\_name, region, role\_arn)  
     #  
     # # List all repositories  
     # list\_all\_repositories(host, awsauth)  
 ​  
     # Create a snapshot  
     snapshot\_name = take\_snapshot(host, awsauth, repo\_name)  
     print(snapshot\_name)  
 ​  
     # # Get snapshot in-progress  
     # get\_snapshot\_status(host, awsauth)  
     #  
     # # List all snapshots in all repository  
     # snapshots = list\_snapshots\_in\_repo(host, repo\_name, awsauth)  
     #  
     # # Delete a snapshot by name  
     # if len(snapshots) > 0:  
     #     delete\_one\_snapshot(host, awsauth, repo\_name, snapshot\_name=snapshots[0].get('snapshot'))  
     #  
     # # List all snapshots in all repository after deletion  
     # list\_snapshots\_in\_repo(host, repo\_name, awsauth)  
 ​  
     return {  
         'statusCode': 200  
    }  
 ​  
 ​  
 def get\_snapshot\_status(host: str, awsauth: AWS4Auth, repo\_name: str = None, snapshot\_name: str = None):  
     """  
    Retrieves a detailed description of the current state for each shard participating in the snapshot.  
    """  
     if repo\_name and snapshot\_name:  
         path = f'/\_snapshot/{repo\_name}/{snapshot\_name}/\_status'  
     elif repo\_name:  
         path = f'/\_snapshot/{repo\_name}/\_status'  
     else:  
         path = f'/\_snapshot/\_status'  
 ​  
     url = host + path  
     r = requests.get(url, auth=awsauth)  
     print(r.text)  
 ​  
 ​  
 def list\_snapshots\_in\_repo(host: str, repo\_name: str, awsauth: AWS4Auth) -> Dict:  
     """  
    List all snapshots in a repository  
    """  
     path = f'/\_snapshot/{repo\_name}/\_all'  
     url = host + path  
 ​  
     r = requests.get(url, auth=awsauth)  
     snapshots = r.json().get("snapshots", [])  
     print(f'Snapshot count = {len(snapshots)}')  
     print(r.text)  
 ​  
     return snapshots  
 ​  
 ​  
 def list\_all\_repositories(host: str, awsauth: AWS4Auth):  
     """  
    List all repositories  
    """  
     path = '/\_snapshot/\_all'  
     url = host + path  
 ​  
     r = requests.get(url, auth=awsauth)  
     print(r.text)  
 ​  
 ​  
 def register\_repository(host: str, awsauth: AWS4Auth, repo\_name: str, bucket\_name: str, region: str, role\_arn: str):  
     """  
    Register a snapshot repository  
    """  
     path = f'/\_snapshot/{repo\_name}'  
     url = host + path  
 ​  
     payload = {  
         "type": "s3",  
         "settings": {  
             "bucket": bucket\_name,  
             "region": region,  
             "role\_arn": role\_arn  
        }  
    }  
 ​  
     headers = {"Content-Type": "application/json"}  
     r = requests.put(url, auth=awsauth, json=payload, headers=headers)  
     print(r.text)  
 ​  
 ​  
 def take\_snapshot(host: str, awsauth: AWS4Auth, repo\_name: str, snapshot\_name: str = None) -> str:  
     """  
    Take a snapshot in a repo. If snapshot\_name is omitted, it will use current datetime string as name.  
    Return snapshot name.  
    """  
     if snapshot\_name is None:  
         # Use current datetime as snapshot name  
         now = datetime.now()  
         snapshot\_name = now.strftime("%Y%m%d-%H%M%S")  
     path = f'/\_snapshot/{repo\_name}/{snapshot\_name}'  
     url = host + path  
 ​  
     r = requests.put(url, auth=awsauth)  
     print(r.text)  
 ​  
     return snapshot\_name  
 ​  
 ​  
 def delete\_one\_snapshot(host: str, awsauth: AWS4Auth, repo\_name: str, snapshot\_name: str):  
     """  
    Deletes a snapshot.  
    """  
     path = f'/\_snapshot/{repo\_name}/{snapshot\_name}'  
     url = host + path  
 ​  
     r = requests.delete(url, auth=awsauth)  
     print(r.text)  
 ​  
 ​  
 def delete\_one\_repository(host: str, awsauth: AWS4Auth, repo\_name: str):  
     """  
    Deletes a snapshot.  
    """  
     path = f'/\_snapshot/{repo\_name}'  
     url = host + path  
 ​  
     r = requests.delete(url, auth=awsauth)  
     print(r.text)  
 ​



1. We need to add Lambda's IAM role to OpensearchSnapshotLambdaRole OpenSearch domain backend before lambda is able to work with the domain.
   * In the domain dashboard, go to Security > Roles.
   * Search for manage\_snapshots.
   * Add IAM ARN as a Backend role in Mapped Users.





**5. Test Lambda Function**

We will update and run the lambda function to register snapshot repository, and then to take snapshot.

1. Update the lambda\_handler() function of Lambda, which is to register a snapshot repository.

 def lambda\_handler(event, context):  
     # Get region and credential  
     service = 'es'  
     session = boto3.session.Session()  
     region = session.region\_name  
     credentials = session.get\_credentials()  
     awsauth = AWS4Auth(credentials.access\_key, credentials.secret\_key, region, service, session\_token=credentials.token)  
 ​  
     # Settings  
     host = '<DOMAIN\_ENDPOINT\_WITH\_HTTPS>'  
     bucket\_name = '<BUCKET\_NAME>'  
     region = '<AWS\_REGION>'  
     role\_arn = '<ARN\_OF\_IAM\_ROLE\_OpensearchSnapshotLambdaRole>'  
     repo\_name = '<REPOSITORY\_NAME>'  
 ​  
     # Register repository  
     register\_repository(host, awsauth, repo\_name, bucket\_name, region, role\_arn)  
 ​  
     # # Create a snapshot  
     # snapshot\_name = take\_snapshot(host, awsauth, repo\_name)  
     # print(snapshot\_name)  
       
     # List all snapshots in all repository after deletion  
     list\_snapshots\_in\_repo(host, repo\_name, awsauth)  
       
     return { 'statusCode': 200 }

1. Run the lambda to register a snapshot repository.
   * Make sure there is no error/warning in the execution output.
2. Update the lambda\_handler() function to take snapshot.

 def lambda\_handler(event, context):  
     # Get region and credential  
     service = 'es'  
     session = boto3.session.Session()  
     region = session.region\_name  
     credentials = session.get\_credentials()  
     awsauth = AWS4Auth(credentials.access\_key, credentials.secret\_key, region, service, session\_token=credentials.token)  
 ​  
     # Settings  
     host = '<DOMAIN\_ENDPOINT\_WITH\_HTTPS>'  
     bucket\_name = '<BUCKET\_NAME>'  
     region = '<AWS\_REGION>'  
     role\_arn = '<ARN\_OF\_IAM\_ROLE\_OpensearchSnapshotLambdaRole>'  
     repo\_name = '<REPOSITORY\_NAME>'  
 ​  
     # # Register repository  
     # register\_repository(host, awsauth, repo\_name, bucket\_name, region, role\_arn)  
 ​  
     # Create a snapshot  
     snapshot\_name = take\_snapshot(host, awsauth, repo\_name)  
     print(snapshot\_name)  
 ​  
     # List all snapshots in all repository  
     snapshots = list\_snapshots\_in\_repo(host, repo\_name, awsauth)  
      
     return { 'statusCode': 200 }

1. Run the lambda function and make sure there is no error/warning message in the output.
2. Check the S3 bucket for the generated files.

**6. Add EventBridge Trigger**

We can use EventBridge to setup a daily schedule to run the Lambda function.

1. 在Lambda函数中里建立一个触发规则，Lambda 函数 > Triggers > Add Trigger。
2. Choose **EventBridge (CloudWatch Events)**
   * Create a new rule
   * Choose Rule type = Schedule expression
   * Set the expression to a cron expression, e.g. cron(0 16 \* \* ? \*)

**结语**

AWS Lambda 是一种无服务器的计算服务，适用于本方案中调用其他AWS服务API进行运维管理工作，与此同时EventBridge提供定时事件触发，它可以与[AWS Lambda](https://aws.amazon.com/cn/lambda/)无缝直接集成。在本案例中，EventBridge定时生成事件并触发Lambda函数执行是一种简单、高效的解决方案，它很好地解决了客户的运维痛点。