AWS Glue

여기서는 Hail을 VCF to Parquet 목적으로 사용하는 법을 설명합니다.

스크린샷이 첨부된 버전은 여기서 확인할 수 있습니다.

?? ??

1. hail-all-spark.jar 파일을 다운로드 받습니다.
2. Amazon S3 서비스로 접속해서 앞에서 다운로드 받은 `hail-all-spark.jar` 파일을 본인에 알맞은 버킷에 업로드합니다.
3. 업로드한 `hail-all-spark.jar` 파일을 선택하고 `Copy S3 URL` 을 눌러 주소를 복사합니다.

이 복사한 주소는 다음 섹션에서 다룰 AWS Glue의 노트북 작업 코드에 필요합니다.

AWS IAM

IAM 서비스로 진입하여 정의된 Role 을 수정합니다. `GenomicsAnalysis-Genomics-JobRole-*` 으로 검색하여 나오는 Role에 대해서 2가지 Policy를 추가할 것입니다.

GetRole, PassRole

1. Create inline policy 를 클릭합니다.
2. 다음과 같이 Policy를 JSON을 선택해서 작성합니다. 이때 반드시 `{account-id}` 는 본인의 AWS 계정아이디와 `{GenomicsAnalysis-Genomics-JobRole-*}` 은 해당되는 것으로 변경해서 작성합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Action": [
```

```
□□□□"iam:GetRole",
□□□□"iam:PassRole"
□□□],
□□□"Resource": [
□□□□"arn:aws:iam::**{account-id}**:role/**{GenomicsAnalysis-Genomics-JobRole-*}**"
□□□]
□□}
□]
}
```

3. 작성한 커스텀 정책의 이름을 입력하고 `Create policy` 를 클릭합니다.
4. 아래와 같이 방금 만든 정책이 해당 Role에 추가된 것을 확인할 수 있습니다. (여기서는 `MyGluePolicy`)

S3 Read

1. 이번에는 사전 정의된 정책을 첨부하여 추가해보겠습니다. `Add permissions` > `Attach policies`
2. `AmazonS3ReadOnlyAccess` 정책을 검색하여 선택합니다.
3. 최종적으로 아래와 같이 2개의 Policy가 2개더 추가된 것을 확인할 수 있습니다. (여기서는 `MyGluePolicy` , `AmazonS3ReadOnlyAccess`)

AWS Glue

1. 콘솔을 통해 AWS Glue 서비스에 접속합니다.
2. ETL jobs > Notebook 을 클릭해서 새로운 노트북을 생성합니다.

이때 IAM role 은 사전에 생성되어 있는 `GenomicsAnalysis-Genomics-JobRole-*` 를 선택합니다.

3. Glue notebook 창으로 돌아와 아래 코드를 모두 붙여 넣습니다. hail-all-spark.jar의 S3 URI 새로운 셀을 추가하려면 원하는 위치에서 `+` 를 눌러 추가할 수 있습니다.

```
%idle_timeout 2880
%glue_version 4.0
%worker_type G.1X
%number_of_workers 5
%additional_python_modules hail
%extra_jars "**{본인의 버킷에 업로드한 hail-all-spark.jar의 S3 URI}**"
```

```
%%configure
{
   "--conf": "spark.serializer=org.apache.spark.serializer.KryoSerializer --conf
spark.kryo.registrator=is.hail.kryo.HailKryoRegistrator"
}
```

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

import hail as hl

sc = SparkContext.getOrCreate()
hl.init(sc=sc)
```

```
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

job.init("JobNameEx")

vds = hl.import_vcf("s3://**{본인의 버킷명}**/genomics-tertiary-analysis-and-data-lakes-using-aws-glue-and-amazon-athena/latest/variants/vcf/variants.vcf.gz", force_bgz=True, reference_genome='GRCh38')

vds.make_table().to_spark().write.mode('overwrite').parquet("s3://**{본인의 버킷명}**/genomics-tertiary-analysis-and-data-lakes-using-aws-glue-and-amazon-athena/latest/variants/vcf_to_parquet")

job.commit()
```

4. 이제 S3 콘솔로 접속하여 결과 Parquet이 잘 만들어졌는지 확인합니다.

Optional ??

- 해당 데이터를 S3의 Query with S3 Select 기능을 사용해 쿼리해봅니다.
- 해당 데이터를 AWS Glue 크롤러를 만들어 카탈로깅해봅니다. 그리고 Athena 에서 쿼리해 볼 수 있습니다.

Revision #3
Created 24 April 2024 12:52:18 by Hyunmin Kim
Updated 26 April 2024 09:07:35 by Hyunmin Kim