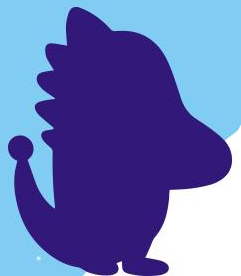
DATAXDAYS

Du on-premise au cloud : Glue vs EMR

Par Scauglog et Franck Cussac
https://github.com/scauglog/prez



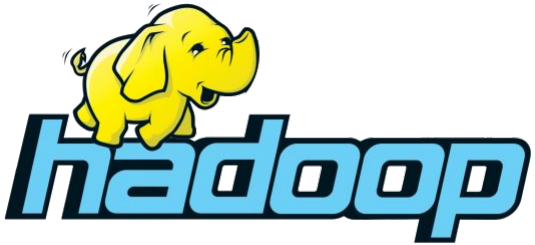
publicis sapient BY ENGINEERS at PUBLICIS SAPIENT

Projet legacy : Hadoop on premise

c’est la galère

On-Premise

- Une dizaine de batch Spark Scala
- Ordonnancé avec airflow
- Lecture écriture sur HDFS
- Hive metastore
- Zeppelin



Pourquoi on veut changer

- Cluster en prod Majoritairement inutilisé (5H - 10H)
- Complexité de gestion du cluster Hadoop
  - Espace de stockage
  - Sécurité (kerberos)
  - Ressources insuffisante en dev

Migration vers cloud

On a choisi AWS...

Migration vers le cloud

- HDFS -> S3
- Airflow -> Airflow
- Zeppelin -> Zeppelin ou SageMaker
- Hive Metastore -> Glue DataCatalog

Glue

stick with it

Qu’est-ce que Glue ?

- Serverless
- AWS ETL
- Spark Managé
- Crawler
- Data Catalog

HOW TO GLUE : Launch Script

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import org.apache.spark.SparkContext
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import scala.collection.JavaConverters._
import xke.local.HelloWorld

object runner {
 def main(sysArgs: Array[String]) {
   val spark: SparkContext = new SparkContext()
   val glueContext: GlueContext = new GlueContext(spark)

   val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME", "input", "output").toArray)

   Job.init(args("JOB_NAME"), glueContext, args.asJava)

   HelloWorld.main(Seq("--input-file", args("input"), "--output-file", args("output")).toArray)

   Job.commit()
 }
}
```

HOW TO GLUE : Launch Script

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import org.apache.spark.SparkContext
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import scala.collection.JavaConverters._
import com.sapient.HelloWorld

object runner {
 def main(sysArgs: Array[String]) {
   val spark: SparkContext = new SparkContext()
   val glueContext: GlueContext = new GlueContext(spark)

   val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME", "input", "output" .toArray)

   Job.init(args("JOB_NAME"), glueContext, args.asJava)

   HelloWorld.main(Seq("--input-file", args("input"), "--output-file", args("output")).toArray)

   Job.commit()
 }
}
```

HOW TO GLUE : Glue Job

```
import boto3

client = boto3.client('glue')

response = client.create_job(
   Name='hello_glue',
   Role='arn:aws:iam::111111111111:role/Glue',
   Command={
       'Name': 'glueetl',
       'ScriptLocation': 's3://artifactory/sapient/scalaRunScript.scala'
   },
   DefaultArguments={
               '--class': 'runner',
               '--job-language': 'scala',
               '--extra-jars': 's3://artifactory/sapient/hello-world-10167-jar-with-dependencies.jar'
   },
   Timeout=20,
   MaxCapacity=2.0,
   GlueVersion='1.0'
)

response = client.start_job_run(
   JobName=job_name,
   Arguments = {
               '--input':   's3://hello/input.txt',
               '--output':  's3://hello/output'
               }
)
```

HOW TO GLUE : Glue Job

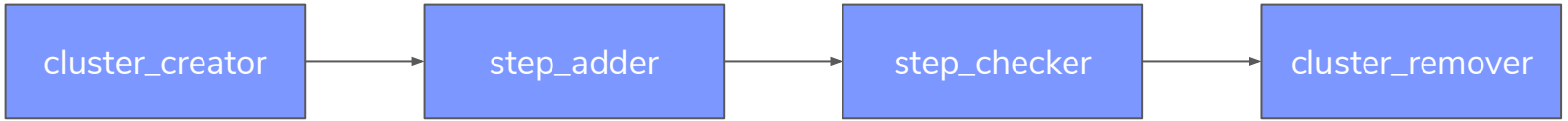
```python
import boto3

client = boto3.client('glue')

response = client.create_job(
   Name='hello_glue',
   Role='arn:aws:iam::111111111111:role/Glue',
   Command={
       'Name': 'glueetl',
       'ScriptLocation': 's3://artifactory/sapient/scalaRunScript.scala'
   },
   DefaultArguments={
               '--class': 'runner',
               '--job-language': 'scala',
               '--extra-jars': 's3://artifactory/sapient/hello-world-10167-jar-with-dependencies.jar'
   },
   Timeout=20,
   MaxCapacity=2.0,
   GlueVersion='1.0'
)

response = client.start_job_run(
   JobName=job_name,
   Arguments = {
               '--input':  's3://hello/input.txt',
               '--output': 's3://hello/output'
               }
)
```

EMR

Qu’est-ce que EMR ?

- Elastic Map Reduce
- Service managé
- Hadoop (HDFS, YARN,...)

Airflow DAG with EMR



HOW TO EMR : EMR definition

```
JOB_FLOW_OVERRIDES = {
    'Name': 'HelloWorld',
    'ReleaseLabel': 'emr-5.29.0',
    'Instances': {
        'InstanceGroups': [
            {
                'Name': 'Master node',
                'Market': 'SPOT',
                'InstanceRole': 'MASTER',
                'InstanceType': 'm5.2xlarge',
                'InstanceCount': 1
            },
            {
                "Name": "Slave nodes",
                "Market": "SPOT",
                "InstanceRole": "TASK",
                'InstanceType': "m5.2xlarge",
                'InstanceCount': 5
            }
        ],
        'KeepJobFlowAliveWhenNoSteps': False
    },
    "Applications": [
         {"Name": "Spark"},
         {"Name": "Zeppelin"},
         {"Name": "Hadoop"}
    ],
    'JobFlowRole': 'EMR_EC2_DefaultRole',
    'ServiceRole': 'EMR_DefaultRole',
}
```

HOW TO EMR : EMR Job

```
SPARK_STEPS = [{
    "Name": "Hello_world",
    "ActionOnFailure": "CANCEL_AND_WAIT",
    "HadoopJarStep": {
    "Jar": "command-runner.jar",
    "Args": [
            "spark-submit",
            "--deploy-mode",
            "cluster",
            "--master",
            "yarn",
            "--conf",
            "spark.sql.session.timeZone=Europe/Paris",
            "--conf",
            "spark.driver.maxResultSize=4g",
            "--executor-memory",
            "11g",
            "--driver-memory",
            "12g",
            "--class",
            "com.sapient.HelloWorld",
            "s3://artifactory/sapient/hello-world-10167-jar-with-dependencies.jar",
            "--input-file", "s3://hello/input.txt",
            "--output-file", "s3://hello/output"
    ]
}]
```

HOW TO EMR : EMR Workflow init

```
cluster_creator = EmrCreateJobFlowOperator(
    task_id='create_job_flow',
    job_flow_overrides=JOB_FLOW_OVERRIDES,
    aws_conn_id='aws_default',
    emr_conn_id='emr_default'
)
```

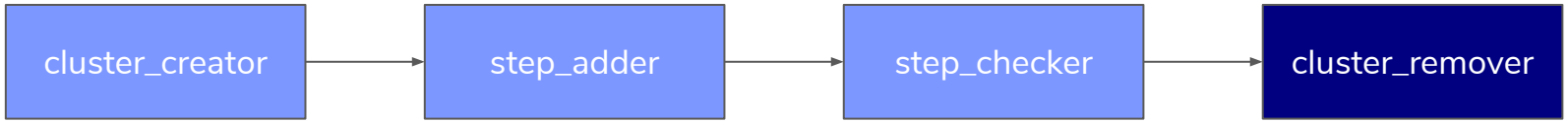
cluster_creator

HOW TO EMR : EMR Workflow tasks

```
step_adder = EmrAddStepsOperator(
        task_id='add_steps',
        job_flow_id="{{ task_instance.xcom_pull(task_ids='create_job_flow', key='return_value') }}",
        aws_conn_id='aws_default',
        steps=SPARK_STEPS
    )

    step_checker = EmrStepSensor(
        task_id='watch_step',
        job_flow_id="{{ task_instance.xcom_pull('create_job_flow', key='return_value') }}",
        step_id="{{ task_instance.xcom_pull(task_ids='add_steps', key='return_value')[0] }}",
        aws_conn_id='aws_default'
    )
```

HOW TO EMR : EMR Workflow end

```
cluster_remover = EmrTerminateJobFlowOperator(
        task_id='remove_cluster',
        job_flow_id="{{ task_instance.xcom_pull(task_ids='create_job_flow', key='return_value') }}",
        aws_conn_id='aws_default'
    )

cluster_creator >> step_adder >> step_checker >> cluster_remover
```



Conclusion

Avantages et inconvénients

Glue

Les Inconvénients

- Script de lancement
- Pas d'intégration avec airflow
- Allocation de ressource peu flexible
  - 4 cores 16Go
  - 8 cores 32Go
- Version de spark Limité (2.2.1, 2.4.3)
- Temps d'initialisation peut être assez long

Start-up time	Execution time
9 mins	1 min
20 secs	50 secs
37 secs	1 min
10 mins	1 min
9 mins	1 min
10 mins	1 min
9 mins	1 min

EMR

Les Inconvénients

- Temps d'initialisation long (10 minutes)
- Configuration de l'EMR à maintenir
- Délaissé par AWS au profit d'autres solutions (EKS)
- Pricing : Service managé + master + workers
- Gestion des transients plus compliqué

Glue

Les Avantages

- Intégré à l'écosystème AWS
- Serverless
- Tooling associé Crawler, Data Catalog, endpoint Zeppelin, Workflow
- En constante Amélioration
- Facturé à la seconde minimum 10 minutes (0,44 USD/h/DPU)
- Timeout

EMR

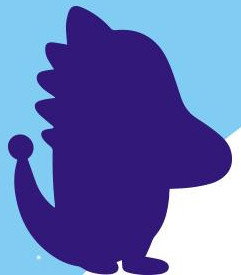
Les Avantages

- Intégré à l'écosystème AWS
- Peut enchainer plusieurs étapes sans cold start
- intégré à airflow
- Utilisation similaire au on-premise

DATAXDAYS

De hadoop à kubernetes

Par Scauglog et Franck Cussac



publicis sapient BY ENGINEERS at PUBLICIS SAPIENT

PROGRAMME
DES SLOTS SUIVANTS

DATAXDAYS

Programme du mardi 16 juin

publicis sapient

13h - 13h45

AI & applied ML

Pierre-Antoine Ganaye :

Toute la vérité sur l'utilisation du deep learning en imagerie médicale

Niveau 2 : ML-Li

17h30 - 18h15

Data Architecture

Ruben Berenguel :

Internals of Speeding up PySpark with Arrow

Niveau 1 : Prototys

18h30 - 18h50

Data Architecture

Franck Cussac & Scauglog :

Du on-premise au cloud : Glue vs EMR

Niveau 1 : Prototys

19h - 19h20

AI & applied ML

Mesut Durukal :

Future of Software Testing: Machine Learning Assistance

Niveau 1 : DS-Li

DATAXDAYS

Programme du mardi 23 juin

publicis sapient

ENGINEERING • PRODUCT • AI & DATA • STRATEGY • EXPERIENCE

13h - 13h45

Real time & streaming data

Robin Moffatt :

Apache Kafka and ksqlDB in Action : Let's Build a Streaming Data Pipeline!

Niveau 1 : Streamèche

17h30 - 18h15

DataScience en prod.

Ryan Dawson :

DevOps for Machine Learning: why is it different?

Niveau 2 : SciProdaffe

18h30 - 19h15

DataScience en prod.

Sarah hakim & Julien Cheillan :

Ré-entraînement automatique de modèles TensorFlow avec AWS

Niveau 2 : SciProdaffe

DATAXDAYS

Programme du mardi 30 juin

publicis sapient

13h - 13h45

DataScience en prod.

Guillaume Blaquiere :

Quelles plateforme de prediction severless sur Google Cloud Platform et avec Tensorflow

Niveau 2 :
SciProdaffe

17h30 - 18h15

AI & applied ML

Niall Turbitt :

Koalas: Parallelising pandas with Apache Spark

Niveau 2 :
ML-Li

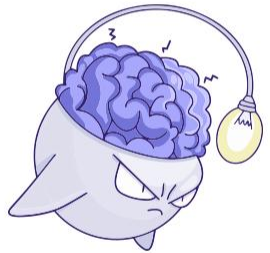
18h30 - 19h15

AI & applied ML

Olga Petrova :

Semi-supervised deep learning with GANs

Niveau 3 :
AI-Li



MERCI

publicis sapient BY ENGINEERS at PUBLICIS SAPIENT

DATAXDAYS