

# RFC on AWS方案

## 背景

海外广告系统建设目标：



RFC需要在AWS上提供特征生产能力。

## 目标

AWS有丰富的组件，可以像搭积木一样构建我们的产品，在产品体验、先进性上都有更大的想象空间，可以预期的最高目标：

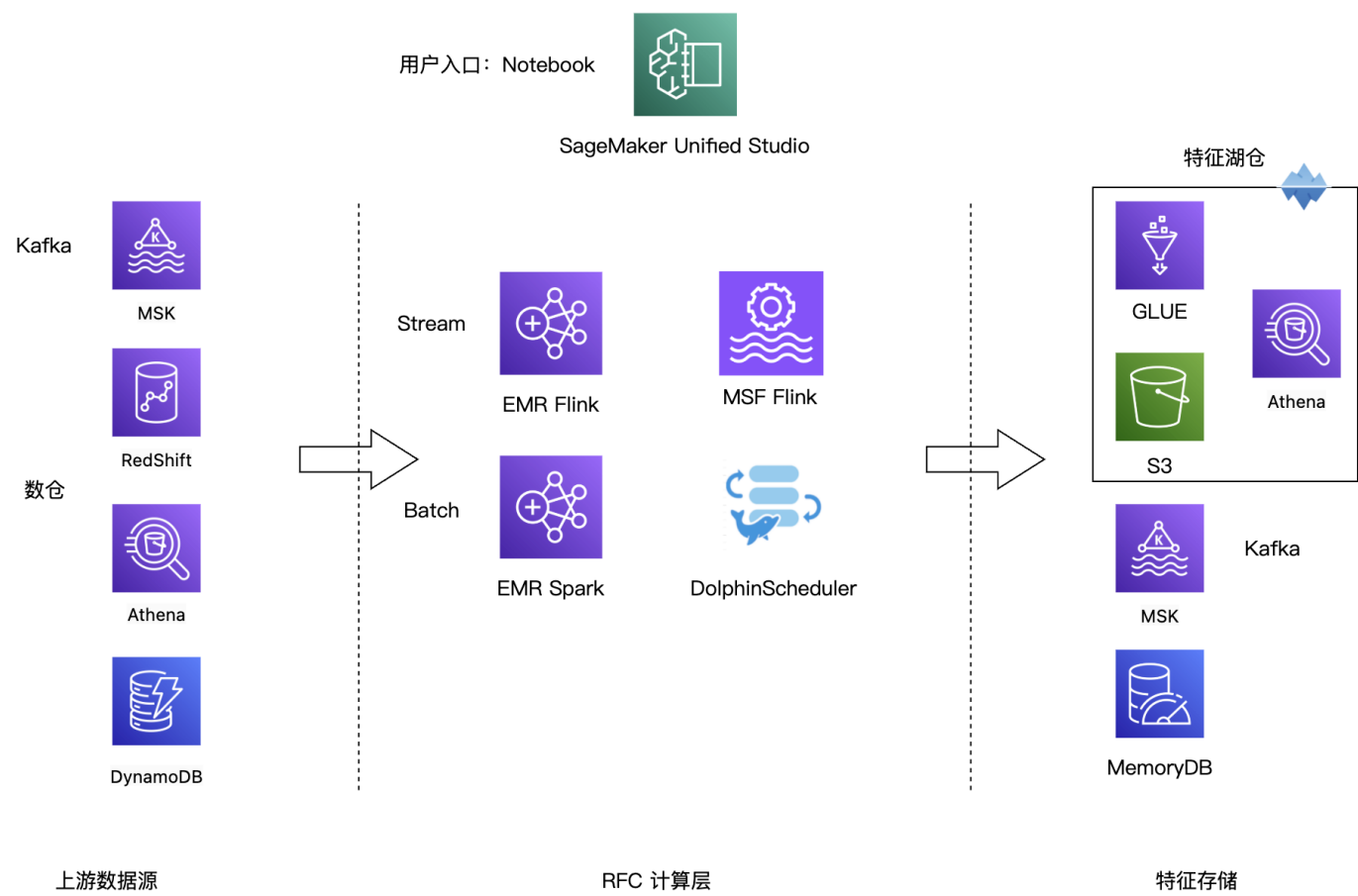
- 在AWS上构建完整的RFC产品，包括：
  - Notebook交互式开发
  - 例行生产调度
  - 特征元数据管理
  - 任务与资源管控
- 基于AWS组件，应用各种框架的前沿版本，强化RFC核心功能，优化性能，提升产品体验：
  - 在Spark 3.5版本上应用Arrow Optimized Python UDF
  - 更新的Flink版本带来更丰富的内置SQL函数
  - Flink + Zeppelin，实现真正的流式交互式开发
  - .....
- 探索最佳实践，提高运维效率，并优化成本

海外广告系统最近的里程碑要求在**2025年8月跑通首条广告**，模型方确定6月份开始联调测试，预计合作的第三方数据会在6月入场，依此倒排，**RFC需要在2025年5月份就具备生产能力**，因此先明确一期目标，即在2025年5月份要在AWS上跑通核心功能：

- Notebook交互式开发
- 例行生产，包括批处理任务的周期调度

RFC长期以来在内部环境上迭代，内部平台如太极、Oceanus、日志汇.....在AWS上没有现成的完全对标产品，搭建完全形态的RFC产品需要较大的工作量，一期目标以数据生产跑通、跑稳为主。

## 总体架构



## 子系统选型与设计

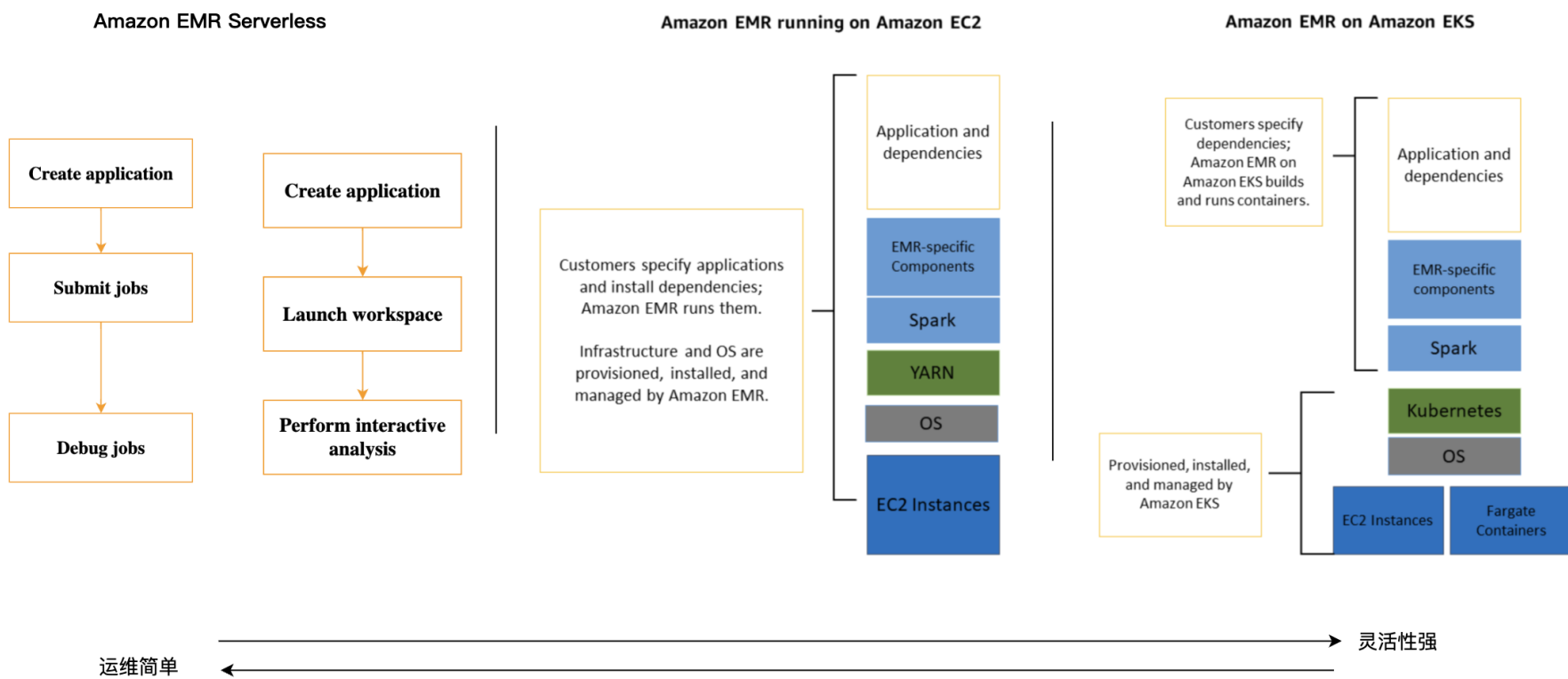
与RFC有关的主要组件：

类别	AWS产品	内网或腾讯云产品
虚拟机	EC2	CVM
对象存储	S3	COS/HDFS
消息队列	*MSK	TDBank
KV存储	DynamoDB	-
	ElasticCache for Redis	Redis
	MemoryDB	Redis
大数据/湖仓查询	Athena	IDEX/WeData SQL
	RedShift	IDEX/WeData SQL
	GLUE	DLA
大数据计算	*EMR Serverless	太极Spark
	EMR on EC2	Yarn
	EMR on EKS	峰峦
	*MSF ( Managed Service for Apache Flink )	Oceanus
Notebook	*EMR Studio	太极Notebook
	*SageMaker Unified Studio	WeData Notebook
任务调度	*MWAA ( Managed Workflows for Apache Airflow )	US

注：\*表示在应用上不能完全对标

## EMR

EMR ( Elastic MapReduce ) 有三种形态，EMR Serverless、EMR on EC2和EMR on EKS：



EMR Serverless是托管服务，不需要运维集群，可以直接提交Spark作业，或在Notebook内交互式操作Spark，不支持Flink。

EMR Studio > Applications > rfc-batch > Submit job run

Submit batch job run

Job details Info

Name

My\_First-Spark-Job

Runtime role

The IAM role assumed by the job. This role must have permissions to access your data sources, targets, scripts, and any libraries used by the job. [Learn more](#)

AmazonEMRStudio\_RuntimeRole\_1742975255257

↺

Script location

The location of the main JAR or Python script in Amazon S3 that you want to run.

🔍 s3://bucket/prefix/object

View

Browse S3

Script arguments

An array of arguments passed to your main JAR or Python script. Your code should handle reading these parameters. Each argument in the array must be separated by a comma.

["argument\_value\_2", "argument\_value\_2" ... ]

▼ Spark properties - optional Info

Additional configuration properties that you can specify for each job. Amazon EMR uses default application properties to help you get started quickly.

Edit in table

Edit in text

Key

Value - optional

🔍 Enter key

🔍 Enter value

Remove

Add new property

You can add up to 49 more properties.

EMR on EC2是Yarn集群，在启动集群时可以指定集群中的机型和要安装的组件，如Spark、Flink，然后通过命令向集群提交作业：

Create cluster [Info](#)

Name

My cluster

Amazon EMR release


[Info](#)

A release contains a set of applications which can be installed on your cluster.


emr-7.8.0

Application bundle


Spark  
Interactive




Core  
Hadoop



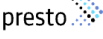
Flink




HBase




Presto



Trino



Custom



☐ AmazonCloudWatchAgent 1.300032.2

☐ HCatalog 3.1.3

☐ Hue 4.11.0

☐ Livy 0.8.0

☐ Pig 0.17.0

☐ TensorFlow 2.16.1

☒ Zeppelin 0.11.1

☒ Flink 1.20.0

☒ Hadoop 3.4.1

☐ JupyterEnterpriseGateway 2.6.0

☐ Oozie 5.2.1

☐ Presto 0.287

☐ Tez 0.10.2

☐ ZooKeeper 3.9.3

☐ HBase 2.6.1

☒ Hive 3.1.3

☐ JupyterHub 1.5.0

☐ Phoenix 5.2.1

☐ Spark 3.5.4

☐ Trino 467

AWS Glue Data Catalog settings

Use the AWS Glue Data Catalog to provide an external metastore for your application.

☐ Use for Hive table metadata

☐ Use for Spark table metadata

Operating system options

[Info](#)

☒ Amazon Linux release

☐ Custom Amazon Machine Image (AMI)

☒ Automatically apply latest Amazon Linux updates

EMR on EKS，需要先准备好装有所需组件的k8s容器，再在上面通过EMR创建虚拟集群，来提交作业。

EMR Serverless价格会比on EC2/EKS贵，也不支持spot（竞价）机器实例，无法像on EC2/EKS一样进一步降低成本，但出于初期尽快跑通和减少运维代价的考虑，应尽可能选择EMR Serverless。

MSF Flink

AWS提供托管Flink服务：MSF（Managed Service Flink）。与EMR Serverless类似，不需要自己运维EMR集群，但功能不完全对标Oceanus。

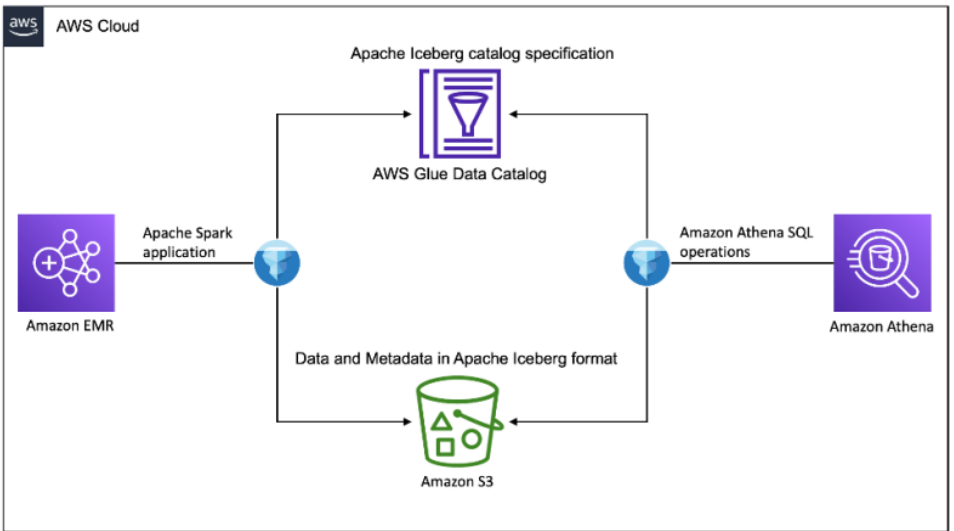
也可以选择EMR on EC2或EMR on EKS。

同样的，应尽可能选择MSF以减轻运维负担。

update: **MSF修改JVM配置需要提工单，只能选择EMR on EC2。**

特征存储

湖仓



通过GLUE Catalog + S3实现特征湖仓存储，用户可以通过serverless的Athena进行查询。

[Working with Amazon S3 Tables and table buckets - Amazon Simple Storage Service](#)

KV存储

服务类型	MemoryDB	DynamoDB
延迟	读：微秒级，写：毫秒级	毫秒级

数据格式	Valkey/Redis OSS，兼容Redis	非结构化数据
数据限制	与Redis相同，无特殊限制	<b>单行不超过400KB（含列名）</b> ，如超过需考虑压缩或通过SortKey扩展成一个Partition对多个Item
备份	天级自动snapshot + 按需snapshot	持续备份 + point in time restore

选择MemoryDB：Redis协议，数据限制少，简化应用层业务逻辑。

// todo: 对比成本

🔗 [与 Valkey 和 Redis OSS 兼容的内存数据库 — Amazon MemoryDB 定价 — AWS](#)

消息队列

选择MSK ( Managed Service for Apahce Kafka ) 即可。AWS尚未提供托管的Pulsar。

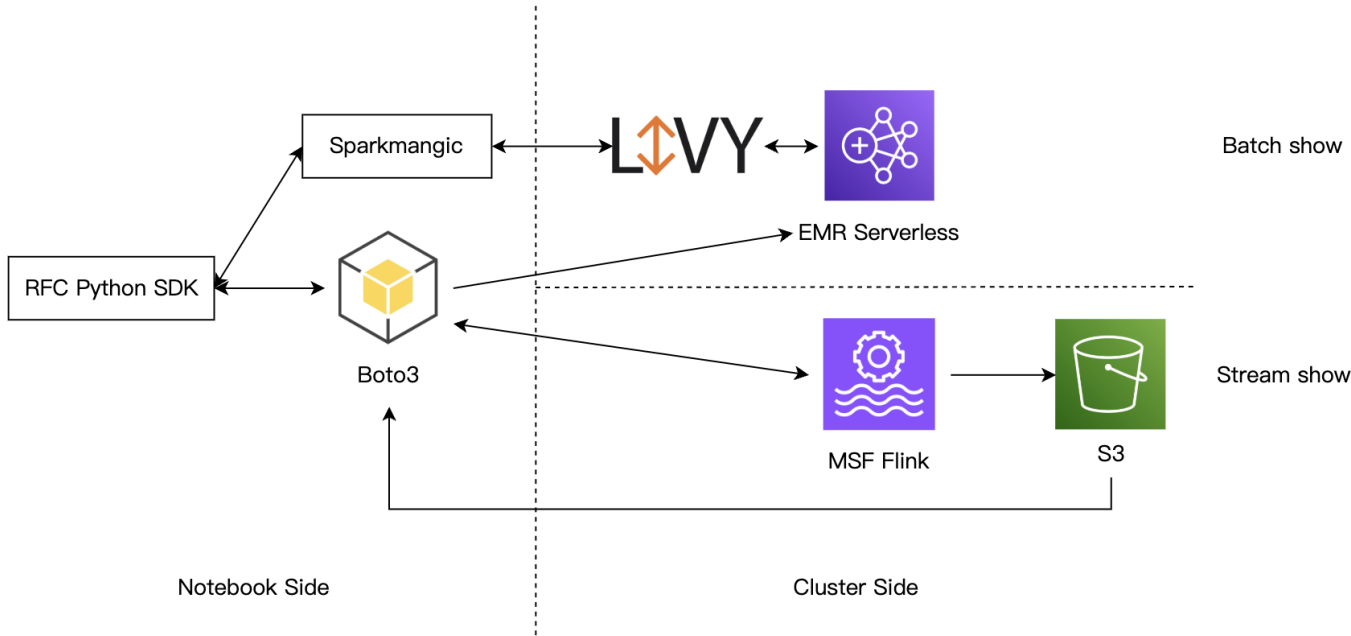
Notebook交互式开发

AWS提供两种Notebook，都不能直接对标太极notebook。

Notebook类型	AWS EMR Studio	AWS SageMaker Unified Studio	太极Notebook
Kernel Python版本	3.7.16	3.10.16	3.6 - 3.9
PySpark连接方式	livy	livy + EMR	client mode，kernel与Driver在一起
Flink连接方式	EMR API	EMR API	Oceanus API

update: **第三种选择：EMR JupyterHub，自主可控** 🔗 [JupyterHub - Amazon EMR](#)

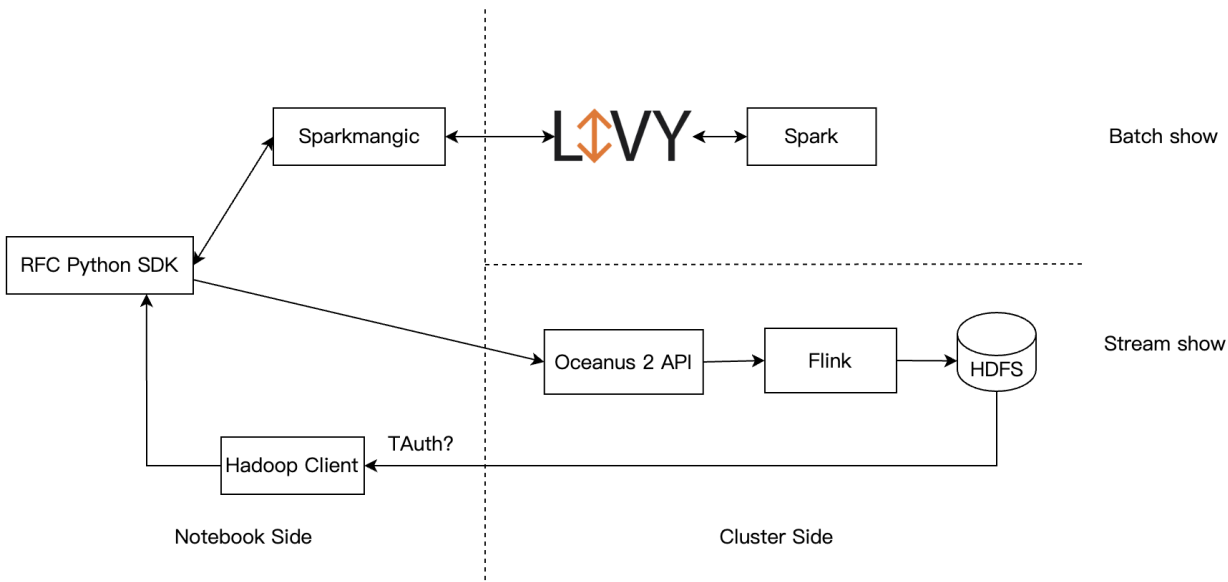
两种Notebook的Python版本都只有一种，注意EMR Studio的Python版本是3.7.16，RFC Python库如要适应这个版本，会有兼容性改造的开发量。所以考虑使用SageMaker Unified Studio，在Notebook中操作EMR Serverless或MSF。



太极Notebook上的PySpark是client mode，Driver就运行在kernel所在本地，可以直接本地操作SparkSession，而SageMaker Unified Studio需要通过livy连接到EMR上的Driver，要通过spark magic来操作（PCG的Venus Notebook也是livy方案），需要对RFC Python库进行适配改造。

对于Flink，没有直接的Notebook交互操作途径，除非直接EMR on EC2，使用集群上Zeppelin的Notebook，但是那样要对RFC Runtime做改造，使它变成一个PyFlink应用，改造代价大，因此一期交互式调试沿用现有做法：通过S3中转show的结果。

update：天穹将在AWS部署大数据组件，但因人力问题无法支持WeData，可能的实现方式：



hadoop client需要较多安装、配置流程：🔗 [https://km.woa.com/articles/show/447290?kmref=search&from\\_page=1&no=2](https://km.woa.com/articles/show/447290?kmref=search&from_page=1&no=2)

update: RFC只使用AWS组件，但有读经营数仓tdw表的需要，需要从EMR读数仓的hive表，涉及权限认证和SparkSession内使用多个hive metastore或iceberg catalog量大问题。

Yes, it is possible to use multiple Hive metastores and Iceberg catalogs within the same SparkSession, but it requires careful configuration.

### 1. Multiple Hive Metastores:

- Spark can be configured to connect to different Hive metastores by specifying the appropriate configuration properties. You can
- However, you cannot directly switch between multiple Hive metastores within the same SparkSession. Instead, you would typically

### 2. Iceberg Catalogs:

- Iceberg supports multiple catalog types (e.g., Hive, Hadoop, and others). You can configure different Iceberg catalogs in your SparkSession
- You can specify the catalog configuration in the SparkSession using the ``spark.sql.catalog.<catalog_name>`` properties. For example:

```
scala
spark.conf.set("spark.sql.catalog.my_catalog", "org.apache.iceberg.spark.SparkCatalog")
spark.conf.set("spark.sql.catalog.my_catalog.type", "hive")
spark.conf.set("spark.sql.catalog.my_catalog.uri", "thrift://<hive-metastore-uri>")
...
```

### 3. Using Multiple Catalogs:

- You can define multiple Iceberg catalogs in the same SparkSession by using different catalog names. For example:

```
scala
spark.conf.set("spark.sql.catalog.catalog1", "org.apache.iceberg.spark.SparkCatalog")
spark.conf.set("spark.sql.catalog.catalog1.type", "hive")
spark.conf.set("spark.sql.catalog.catalog1.uri", "thrift://<hive-metastore-uri-1>")

spark.conf.set("spark.sql.catalog.catalog2", "org.apache.iceberg.spark.SparkCatalog")
spark.conf.set("spark.sql.catalog.catalog2.type", "hive")
spark.conf.set("spark.sql.catalog.catalog2.uri", "thrift://<hive-metastore-uri-2>")
...
```

### 4. Switching Context:

- When you want to use a specific Iceberg catalog, you can switch the context by using SQL commands. For example:

```
sql
USE catalog1.database_name;
...
```

- This allows you to work with tables in the specified catalog.

### 5. Considerations:

- Be mindful of the performance implications and potential conflicts when using multiple catalogs and metastores.
- Ensure that the configurations do not conflict with each other, especially when it comes to table names and database names.

### Example Code:

Here's a simple example of how you might set up a SparkSession with multiple Iceberg catalogs:

```
scala
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder()
  .appName("Multiple Iceberg Catalogs Example")
  .config("spark.sql.catalog.catalog1", "org.apache.iceberg.spark.SparkCatalog")
  .config("spark.sql.catalog.catalog1.type", "hive")
  .config("spark.sql.catalog.catalog1.uri", "thrift://<hive-metastore-uri-1>")
  .config("spark.sql.catalog.catalog2", "org.apache.iceberg.spark.SparkCatalog")
  .config("spark.sql.catalog.catalog2.type", "hive")
  .config("spark.sql.catalog.catalog2.uri", "thrift://<hive-metastore-uri-2>")
  .getOrCreate()
```



例行生产

任务调度

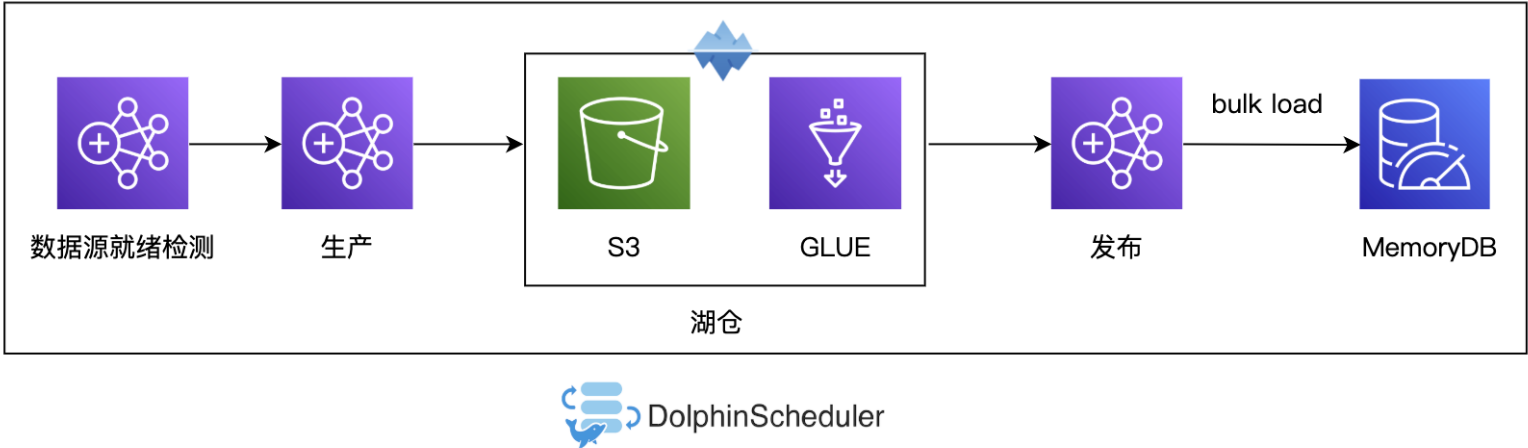
AWS没有直接对标US或太极的产品，需要有调度系统来周期性跑批处理任务。

AWS提供托管的MWAA（Managed Workflows for Apache Airflow），不过Airflow的DAG要用Python代码表达，考虑到易用性，选择搭建serverless Apahce DolphinScheduler：

[AWS 部署无服务器 DolphinScheduler | 亚马逊AWS官方博客](#)

update：可能使用AWS上的US。API鉴权方式与Oceanus相同，都是用个人cmk。<https://iwiki.woa.com/p/195788408>

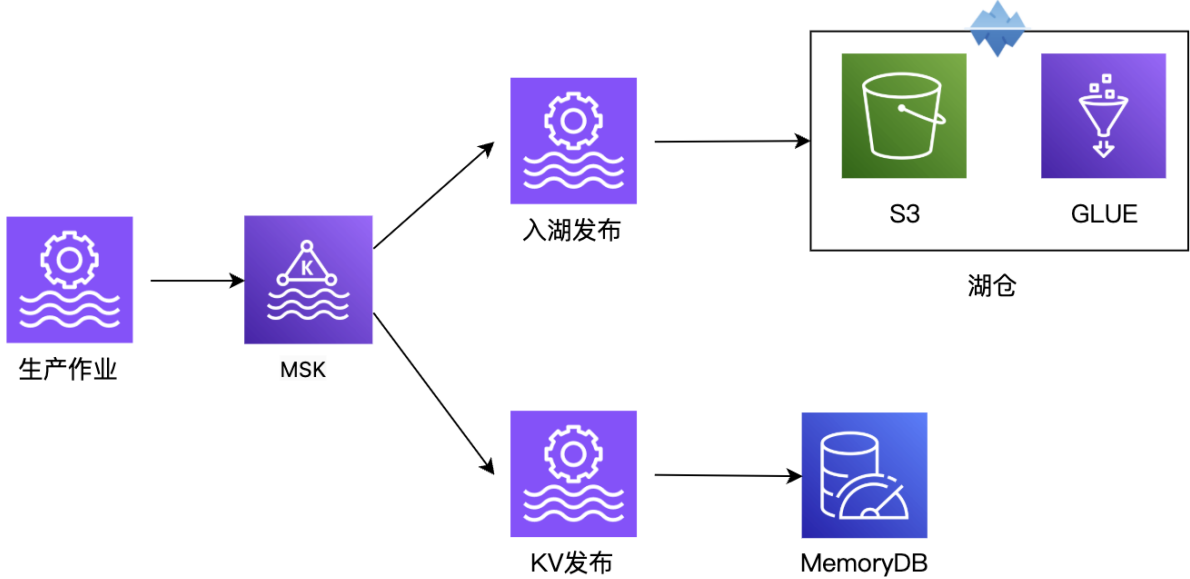
Batch



在DolphinScheduler调度下进行例行生产：特征先入湖仓，再bulk load到MemoryDB。

Redis bulk load，本质上是pipeline：[Bulk loading | Docs](#)

Stream



生产作业不直接写存储，而是写入Kafka topic，由发布作业进行入湖发布或KV发布，以提高生产作业的健壮性。

风险点：Pemja实现的Python UDF在社区版Flink上可能会有各种环境和jar冲突问题，如无法解决，要考虑将整个RFC作业改成PyFlink实现，改动较大，或禁用Python UDF，只使用Scala UDF。

监控与日志

业务监控：Prometheus + managed Grafana

任务监控：EMR Serverless与MSF自带，如使用EMR on EC2/EKS，需要另外建设

日志查询：AWS有CloudWatch，成本较高，或者可以使用OpenSearch（托管的ELK）

元数据管控

主要是特征ID管理，一期可以人工维护。但需要做好流程控制。

RFC框架改造

组件与环境兼容

内部环境	AWS
TDBank SDK	Kafka Connector
HDFS	S3
监控宝	Prometheus + Grafana
日志汇	CloudWatch/ELK

Spark tq系列	Spark社区版
Flink tq系列	Flink社区版
X-Stor	MemoryDB
Iceberg Runtime tq系列	Iceberg社区版
太极Notebook/WeData Notebook	SageMaker Unified Studio

算子支持

需新开发：

- Kafka Source
- Kafka Sink
- Redis Sink
- .....

发布任务

流和批都需开发新的发布任务：入湖与KV发布。

时间节点

时间点	事项
2024-04-10	跑通batch和stream作业提交，验证EMR Serverless与MSF功能
2024-04-22	跑通notebook交互式调试
2024-04-25	完成DolphinScheduler服务搭建
2025-05-20	完善算子功能，RFC框架具备生产能力