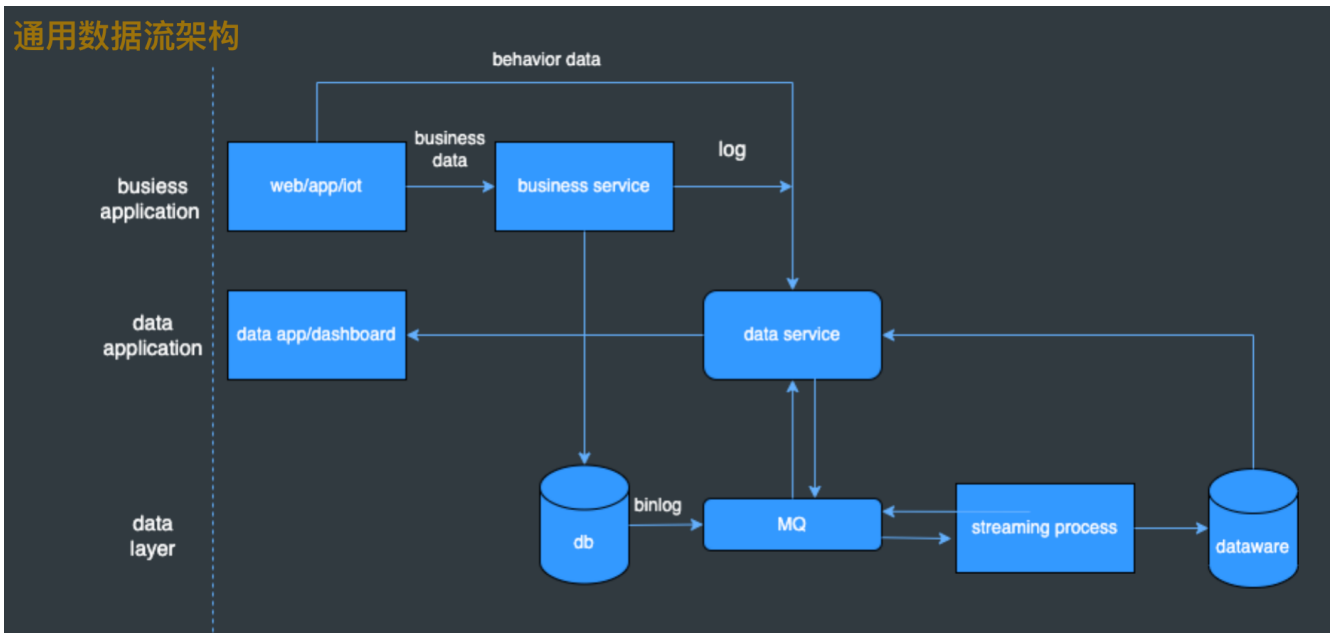
使用亚马逊无服务器技术轻松构建数据实时大屏

> ✊ 经验没有压缩算法。 —— 安迪 · 贾西

本文档以构建实时数据分析和数据大屏为场景，介绍如何使用aws相关服务轻松，快速实现流式开发以及过程中的痛点难点

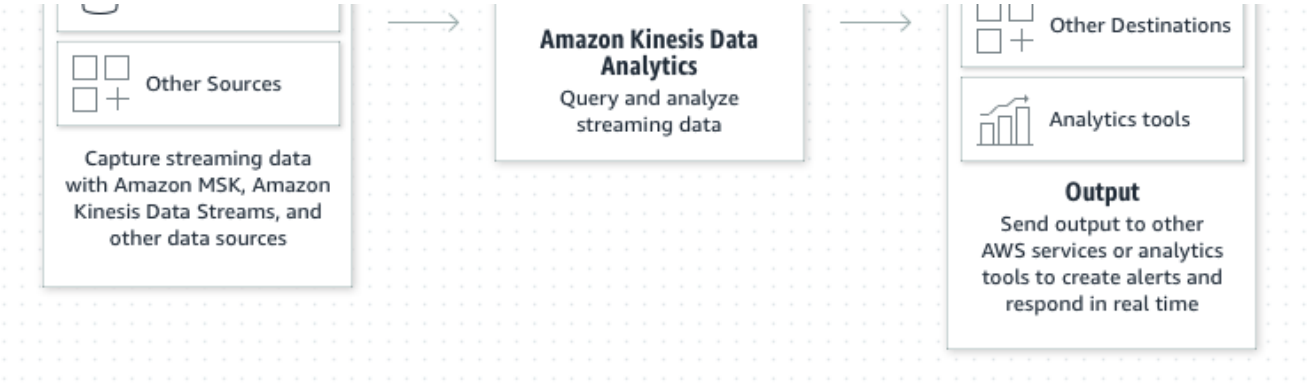
一、通用方案



二、服务介绍

Amazon Kinesis Data Analytics

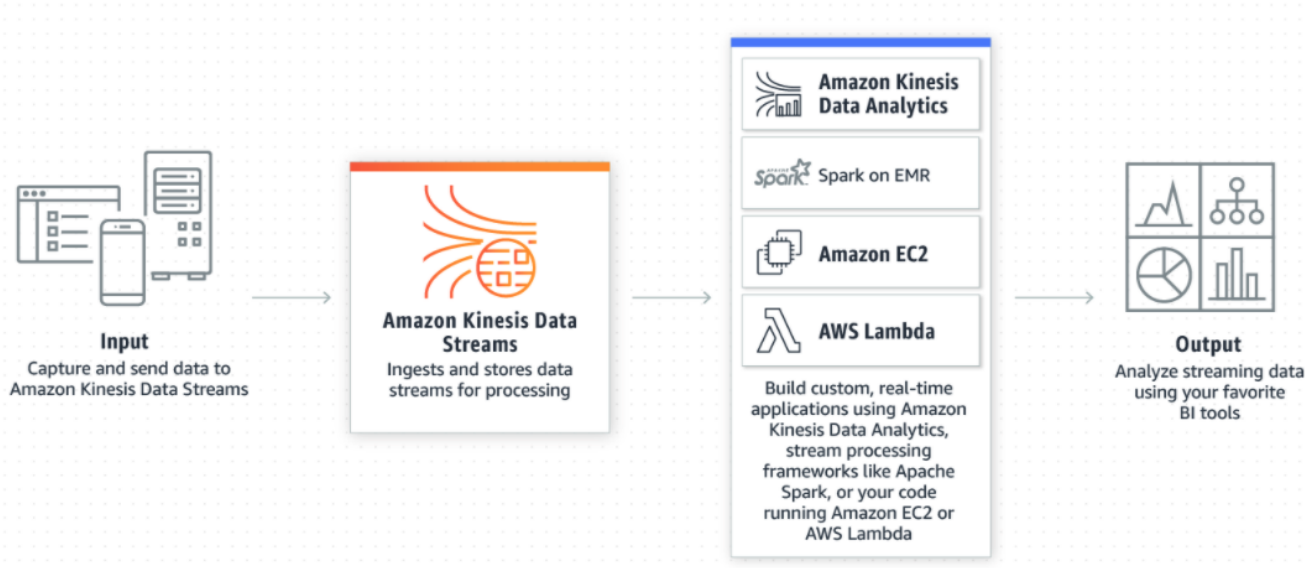
> 使用无服务器的完全托管式 Apache Flink 从串流数据中获得可行见解



使用文档

Amazon Kinesis

> Amazon Kinesis 可让您轻松收集、处理和分析实时流数据，以便您及时获得见解并对新信息快速做出响应。Amazon Kinesis 提供多种核心功能，可以经济高效地处理任意规模的流数据，同时具有很高的灵活性，让您可以选择最符合应用程序需求的工具。借助 Amazon Kinesis，您可以获取视频、音频、应用程序日志和网站点击流等实时数据，也可以获取用于机器学习、分析和其他应用程序的 IoT 遥测数据。借助 Amazon Kinesis，您可以即刻对收到的数据进行处理和分析并做出响应，无需等到收集完全部数据后才开始进行处理



使用文档

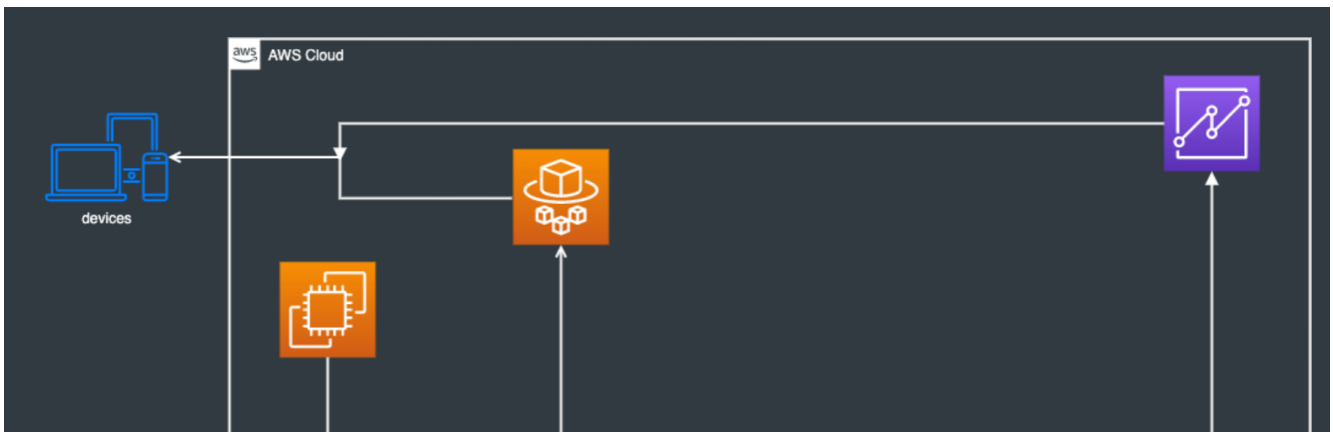
Amazon Kinesis Data Firehose

> Amazon Kinesis Data Firehose 是将流数据可靠地加载到数据湖、数据存储和分析服务中的最简单方式。该服务可以捕获和转换流数据并将其传输给 Amazon S3、Amazon Redshift、Amazon OpenSearch Service（接替 Amazon Elasticsearch Service）、通用 HTTP 终端节点和服务提供商（如 Datadog、New Relic、MongoDB 和 Splunk）。这是一项完全托管的服务，会自动扩展以匹配数据吞吐量，并且无需持续管理。该服务还可以在加载数据前对其进行批处理、压缩、转换和加密，从而最大程度地减少所用存储量，同时提高安全性。

使用文档

三、方案设计

> 从业务数据库通过maxwell采集binlog以后,发送到名为order_ds的kinesis data stream中
>
> kinesis data analystic实时处理order_ds的数据，进行ETL
>
> 一方面，kinesis data analystic 将处理好的中间模型表通过Kinesis data firehouse 写入redshift
>
> 另一方面 kinesis data analystic 将高度聚合好的指标数据直接写入kinesis datastream
>
> 最后，数据服务实时消费指标数据流并推送到前端，实现实时数据大屏
>
> BI 分析工具查询redshift,实现即席查询和看板制作



三、准备工作

> 我们这里将模拟用户的下单流程，所以我们用一个mysql业务库demo的order表，用来保存用户的订单数据。
>
> 我们的flink程序将处理订单的binlog流数据，并和数仓中的维度表进行关联，生成一个dwd模型宽表customer_order,所以我们还需要一个redshift集群，用来保存维度表customer,该表存放在redshift中dev库中。

1. 创建 工作环境

- 建议在您的vpc环境起一台ec2用来跑相关脚本
- 创建ec2过程可参考 这里 或者咨询架构师
- 安装 并配置 aws cli

Python

```
pip3 install awscli

aws configure
```

配置过程请参考 使用文档

❤️ 后续步骤中的脚本都将运行在该机器上

- 拉去workshop 脚本

Bash

```
git clone git@github.com:tingxin/kinesis-workshop.git
```

- 运行如下脚本安装依赖包

Bash

```
pip3 install -r requirement.txt
```

❤️ 如果在您的工作机上遇到 permission deny 相关的错误，请在所有的命令前添加 sudo

2. rds mysql 数据库集群

> mysql 作为测试数据源

- 主库访问的终端节点。下面训练将用模拟程序写入数据到mysql

Java

```
demo.c6lwjjfhbm6a.rds.cn-northwest-1.amazonaws.com.cn
```

- 确保您的mysql数据库主库开启了binlog，并设置更改 binlog_format 参数的值为row。具体方法请参考 文档
- 在ec2 的页面创建 容许对3306端口流量进出的安全组，配置给mysql服务
- 在vpc页面创建对mysql可以访问的终端节点
- 进入mysql 主库，并创建订单表

Bash

```
mysql -h demo.c6lwjjfhbm6a.rds.cn-northwest-1.amazonaws.com.cn -P 3306 -u
admin -p
```

SQL

```sql
use demo;

CREATE TABLE IF NOT EXISTS `order` (
    order_id INT AUTO_INCREMENT NOT NULL,
    user_mail varchar(20) NOT NULL,
    status char(10) NOT NULL,
    good_count INT NOT NULL,
    city varchar(20) NOT NULL,
    amount FLOAT NOT NULL,
    create_time datetime NOT NULL,
    update_time datetime NOT NULL,
    PRIMARY KEY (`order_id`)
);
```

-

3. redshift 数仓集群

- 在集群页面找到jdbc链接字符串备用
- 配置redshift的安全组，容许 5439（redshift的默认端口） 的流量访问
- 在vpc页面创建对redshift可以访问的终端节点

四、处理流数据

1. 创建 kinesis datastream

- 创建 名为order_ds 的 kinesis datastream， 用于接收从mysql 采集到的订单表的binlog数据流
- 创建 名为metric_ds 的 kinesis datastream， 用于接收后面flink程序处理好的指标数据流
- 创建 名为dwd_customer_order_ds 的 kinesis datastream， 用于接收后面flink程序处理好的模型表。这条流将作为firehouse 的输入，被firehouse写入数仓redshift中
- 创建过程都比较简单，不再赘述

2. 使用maxwell伪装成mysql 从库，采集binlog数据流

> maxwell 在本文中将采用docker 部署方案。你可以开启一台ec2来运行，也可以部署在EKS (K8S在aws的托管方案)上。由于EKS 和 K8S 也是一个比较庞大的话题，本文不在此赘述，将采用部署在ec2的简单方案。

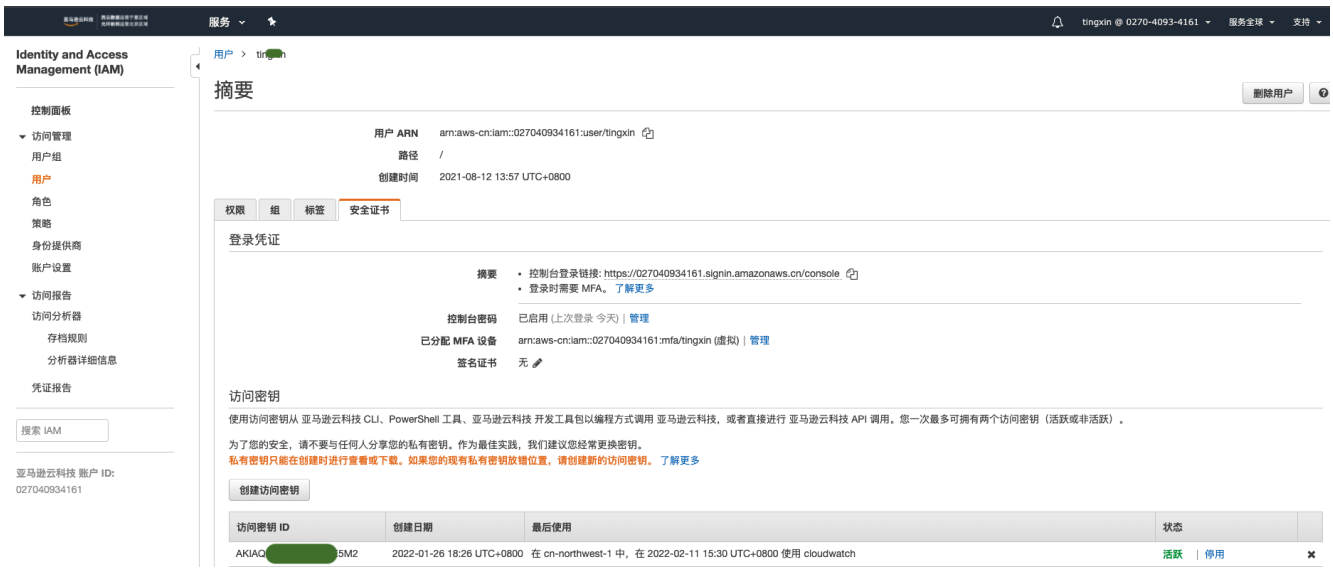
- 检查当前工作环境的ec2是否安装docker,如果没有安装，请参考 安装docker 进行安装

- 运行如下脚本获取maxwell 的docker image

Bash

```
docker pull zendesk/maxwell
```

- 在当前工作环境配置如下环境变量



> Aws access key 和 aws secret key 是一组代表某个aws用户的凭证。在我们的workshop中，需要该用该用户具有读rds数据库并把binlog写入kinesis data stream的权限
>
> 这两个key 你可以按上图在IAM中获取

Bash

```
export AWS_ACCESS_KEY_ID={您的access key}
export AWS_SECRET_KEY={您的access secret}
```

> export 的方式保存环境变量，只在当天终端起作用，这样是为了安全.相对的，你也可以通过修改.bash_profile的方式，永久将AWS_ACCESS_KEY_ID 环境变量写入ec2,但相对不安全

- 创建如下mysq了用户，并赋权。maxwell将使用该用户访问你的mysql数据库，抽取binlog

```
mysql> CREATE USER 'maxwell'@'%' IDENTIFIED BY 'XXXXXX';
mysql> CREATE USER 'maxwell'@'localhost' IDENTIFIED BY 'XXXXXX';

mysql> GRANT ALL ON maxwell.* TO 'maxwell'@'%';
mysql> GRANT ALL ON maxwell.* TO 'maxwell'@'localhost';

mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'maxwell'@'%';
mysql> GRANT SELECT, REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'maxwell'@'localhost';
```

- 修改如下脚本，将花括号包裹起来的参数替换成您的配置,然后在控制台运行

Dockerfile

```
docker run -it --rm --name maxwell-kinesis -e AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID -e AWS_SECRET_KEY=$AWS_SECRET_KEY -v `cd && pwd`/.aws:/root/.aws zendesk/maxwell  sh -c 'cp /app/kinesis-producer-library.properties.example /app/kinesis-producer-library.properties && echo "Region={AWS_REGION}" >>/app/kinesis-producer-library.properties && /app/bin/maxwell --user={DB_USERNAME} --password={DB_PASSWORD} --host={MYSQL_RDS_URI} --producer=kinesis --kinesis_stream={KINESIS_NAME}'
```

Dockerfile

```
## 修改后应该类似如下
docker run -it  --name maxwell -e AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID -e AWS_SECRET_KEY=$AWS_SECRET_KEY -v `cd && pwd`/.aws:/root/.aws zendesk/maxwell sh -c 'cp /app/kinesis-producer-library.properties.example /app/kinesis-producer-library.properties && echo "Region=cn-northwest-1" >> /app/kinesis-producer-library.properties && bin/maxwell --user=maxwell --password=Demo1234 --host=demo.c6lwjjfhbm6a.rds.cn-northwest-1.amazonaws.com.cn --producer=kinesis --kinesis_stream=order_ds'
```

- 脚本 write_db.py 将模拟用户下单，将订单数据插入数据库order表中。运行如下命令执行该脚本

> 请注意修改setting.py的相关配置，例如mysql地址，账号密码，以及数据库名称和表名

Bash

```
python3 write_db.py
```

- 使用脚本read_kinesis.py 验证binlog数据流是否已经加载到 order_ds 数据流中

Bash

```
python3 read_kinesis.py order_ds
```

3. 使用kinesis data analystic 运行 flink程序

- 在工作ec2上拉取代码： git@github.com:tingxin/dongpeng-metrics.git
- 代码运行在 Amazon Corretto 11 ,默认ec2安装的是jdk 8, 所以需要修改默认jdk. 修改请参考使用文档
- 根据需要修改代码

Python

```
src/main/java/com/amazonaws/param # 该目录下存放一些组件和服务的基础配置，需要根据您的实际情况修改
src/main/java/com/amazonaws/components/input # 存放输入流的逻辑，根据情况修改
src/main/java/com/amazonaws/components/output # 存放输出流的逻辑，根据情况修改
```

- 进入代码进行打包

Bash

```
mvn package -f pom.xml
```

- 上传代码到s3 ,上传前请先在s3上配置一个bucket用来存放jar文件

Bash

```
aws s3 cp target/aws-kinesis-analytics-java-apps-1.0.jar s3://<您的s3存储桶>/aws-kinesis-analytics-java-apps-1.3.jar
```

- 进入aws kinesis data analystic 控制台，创建任务，flink 版本选1.13。 过程比较简单，不再赘述
- 任务创建成功后，点击配置，进入任务配置页面
- 配置代码位置 （记下系统帮你创建好的IAM 角色，后续需要修改权限）

- 配置网络环境，即您的程序跑在那个网络。建议使用无vpc
- 配置运行时属性
- 保存配置

- 点击运行按钮运行代码，启动运行大约要1~2分钟
- 运行成功后在控制台点击按钮 “打开 apache flink 控制面板” 查看flink 任务详情

4. 验证flink输出数据

- 新建一个终端，回到 代码项目 kinesis-workshop
- 使用如下命令查看处理后的流数据

```
Nginx
1  python3 read_kinesis.py metric_ds
```

5. 将生成的流数据通过firehouse写入数仓redshift

- 新建一个传输流，根据flink 代码里编写的逻辑选择输入方式。目前demo代码，将计算的模型表写入了一条新的data stream，所以这里选择输入方式为kinesis data stream。如果代码中直接使用firehouse stream作为producer,则应该选择Direct PUT 方式
- 选择目标为redshift
- 按如下配置 Amazon Redshift COPY 命令。由于firehouse的工作方式是将缓冲的数据按照时间规则或大小规则暂存在S3,当规则满足后，触发COPY明细将数据批量从S3复制到redshift,所以需要配置这些参数。另外我们的数据格式是json ,所以需要复制的时候进行识别。由于需要json转换，copy 命令选项需要添加 json 'auto'
- 在redshift创建你的目标表

- 点开高级，配置缓冲策略

- 配置role,默认会帮你自动创建

五、制作实时数据大屏

1. 构建数据服务推送实时指标

- 运行如下脚本启动数据服务

```
Bash
1  python3 data_server.py metric_ds
```

- 数据服务默认监听端口3008，您可以根据需要修改。配置安全组，容许外网访问
- 打开浏览器，输入 您工作机的ip 地址:3008 即可查看一个简单的实时看板

六、制作看板

1. 使用quicksight 制作看板(国内客户暂时无法使用quicksight)

> Amazon QuickSight 是一项快速的业务分析服务，用于构建可视化、执行临时分析以及从数据中快速获取业务洞察。Amazon QuickSight 无缝地发现 AWS 数据源，使企业能够扩展到数十万用户，并通过使用 Amazon QuickSight Super-fast, Parallel, In-Memory, Calculation Engine (SPICE) 提供快速响应的查询性能。

这里有比较详尽的教程：

https://docs.aws.amazon.com/zh_cn/quicksight/latest/user/quickstart-createanalysis.html

七、常见问题

1. 启动maxwell遇到

Bash

```
com.github.shyiko.mysql.binlog.network.ServerException: Could not find first log file name in binary log index file
        at com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:944) [mysql-binlog-connector-java-0.25.3.jar:0.25.3]
```

解决方法参考文档

2. 配置kinesis data analys时，如果选择以后的vpc,访问kinesis data stream失败[需进一步研究]
3. Redshift 配置了公网访问，如何容许kinesis data analystic访问 【需要细化明确】

八、优化

1. 使用 AWS Secrets Manager 创建密码托管

> 由于flink 要访问redshift的customer维度表，无论是将redshift的密码硬编码在代码里，还是通过应用程序参数传递进去，都不安全，建议使用Secrets Manager