Relatório Técnico

1. Objetivo do Case

O objetivo do case é desenvolver uma solução de engenharia de dados para consumir informações de partidas do jogo League of Legends em tempo real utilizando a API oficial da Riot Games. O fluxo de dados deve ser processado e armazenado em um Data Lake, estruturado em camadas (bronze, silver, gold). Além disso, a solução deve permitir monitoramento e observabilidade, garantindo a rastreabilidade do fluxo de dados, detecção de falhas e identificação de gargalos de desempenho.

Github: https://github.com/diegomeyer/datamaster

1.1. Conta Developer Riot

- Crie uma conta no portal Riot Games
- Faça a verificação do email cadastrado
- Aceite os termos
- Crie de fato a conta de developer
- Acesse o link Developer Portal
- Gere uma nova chave da API

1.2. Configurando o Ambiente

- Execute o comando
  ./start_services.sh **API_KEY**

2. Arquitetura de Solução e Arquitetura Técnica

2.1. Arquitetura de Solução

A solução foi projetada em um pipeline de dados com as seguintes etapas principais:

- Extração:
  i. Dados de partidas são consumidos da API da Riot Games.
  ii. As informações são enviadas para o tópico summoners no Kafka.
- Processamento em Tempo Real:
  i. Dados do Kafka são consumidos pelo Spark.
  ii. Os dados são escritos na camada bronze do Data Lake.
- Transformação e Limpeza:
  i. Dados da camada bronze são processados e normalizados para a camada silver.
- Agregação:
  i. A camada gold é criada com agregações e métricas prontas para análise, como KPIs.
- Monitoramento:

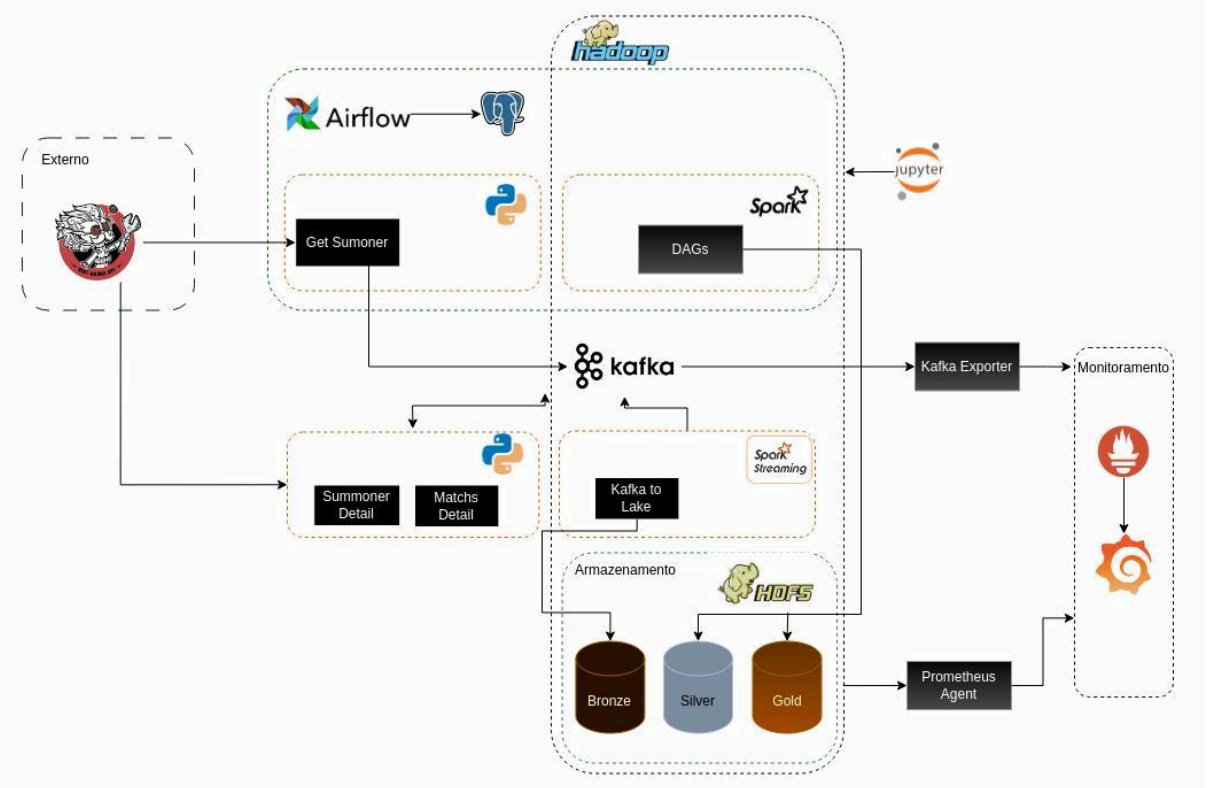
i. Métricas de todo o pipeline são monitorados usando Prometheus e Grafana.

2.2. Arquitetura Técnica

Tecnologias Utilizadas:

Tecnologia	Função
Kafka	Sistema de mensageria para ingestão de dados em tempo real.
Kafka Exporter	Export as métricas do Kafka para o Prometheus
Spark	Processamento distribuído de dados.
HDFS	Armazenamento em Data Lake com suporte a grandes volumes.
Jupyther	Ambiente para exploração dos dados
Airflow	Schedular Jobs
Prometheus	Coleta de métricas para monitoramento.
Grafana	Visualização de métricas em dashboards.
Docker Compose	Orquestração dos serviços.

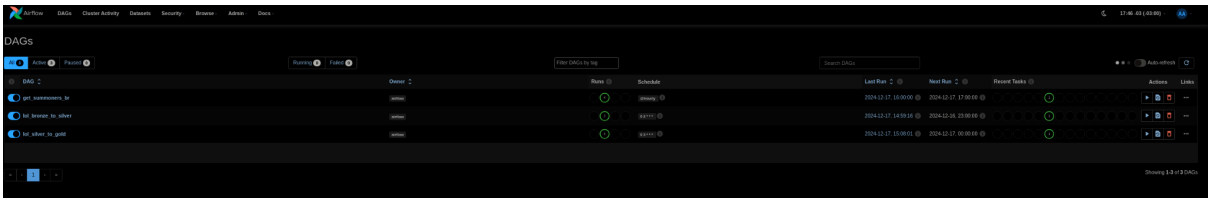
Arquitetura:



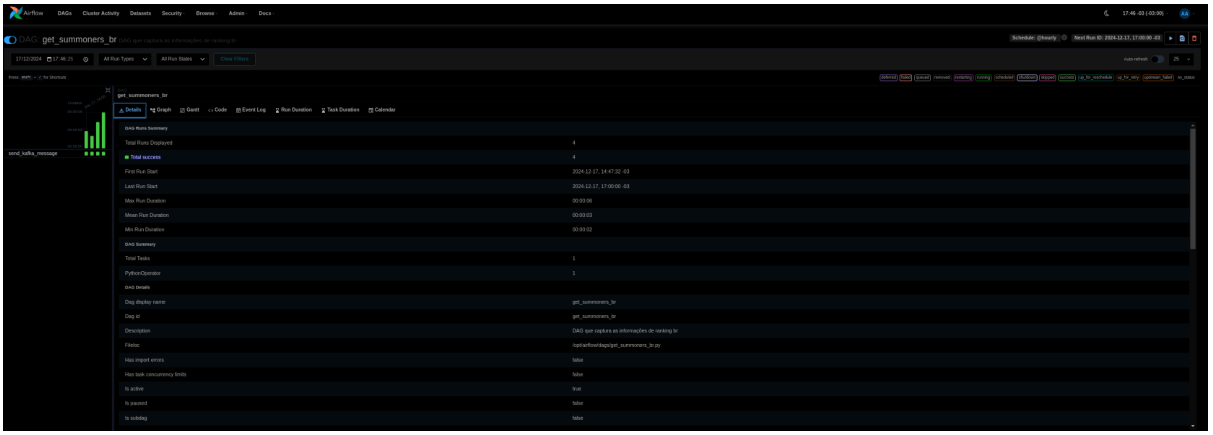
3. Explicação sobre o Case Desenvolvido

3.1. Extração de Dados

O script get_summoners_br.py consulta a API da Riot Games e está agendado para executar de hora em hora, onde coleta os IDs dos jogadores da região do brasil e envia informações sobre partidas para o tópico Kafka *summoners*.
O script streaming kafka_summoner_details.py recebe o ID do jogador e buscas os IDs de das últimas 3 partidas e envia esses IDs para o topico Kafka *summoners_details*
O script streaming kafka_matchs.py recebe o ID da partida e busca as informações da partida e envia o json para o tópico Kafka *matchs*.



DAGs



DAG - Extração de Dados

3.2. Processamento de Dados

O script consumer_kafka_to_lake.py:

- Consome mensagens do Kafka em tempo real no tópico match.
- Escreve os dados brutos e as métricas de cada etapa na camada bronze do Data Lake em formato Parquet.
- Assegura tolerância a falhas com checkpoints.

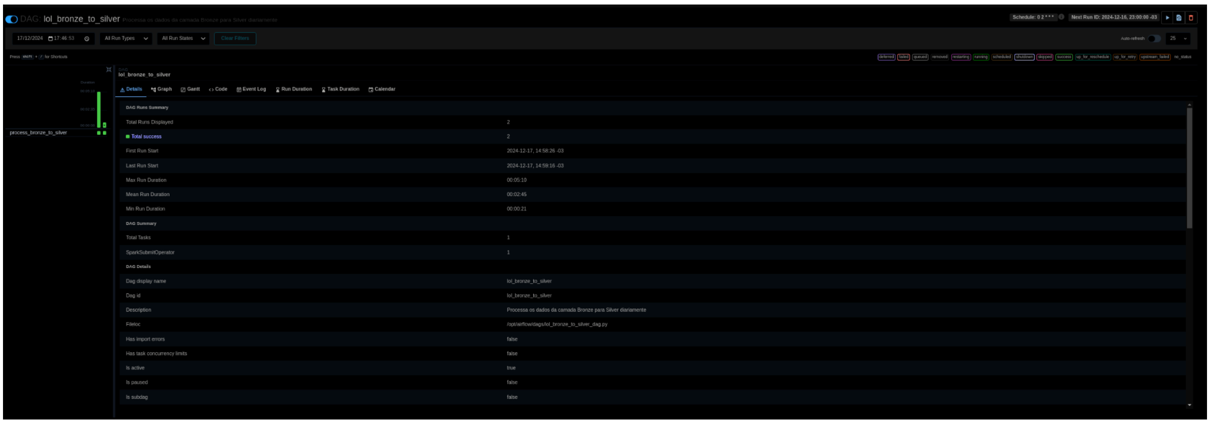
```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/bronze/matchs/*/*.parquet").printSchema()
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
root
 |-- timestamp_summoner: string (nullable = true)
 |-- timestamp_summoner_details: string (nullable = true)
 |-- timestamp_match: string (nullable = true)
 |-- match_details: string (nullable = true)
```

Schema - Tabela Bronze

O script bronze_to_silver.py

- DAG que consome a camada bronze e extrai e estrutura as informações mais relevantes.



DAG - Bronze to Silver

```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/silver/games/*/*.parquet").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+-------------+-------------+---------+-------------+------+--------+-----------+--------------------+------------------+
|match_id  |game_creation|game_duration|game_mode|game_version  |map_id|queue_id|platform_id|game_start_timestamp|game_end_timestamp|
+----------+-------------+-------------+---------+-------------+------+--------+-----------+--------------------+------------------+
|3038395782|1734120237863|1893         |CLASSIC  |14.24.643.5128|11    |420     |BR1        |1734120279105       |1734122172672     |
|3039998212|1734381028131|1863         |CLASSIC  |14.24.643.5128|11    |420     |BR1        |1734381055205       |1734382918667     |
+----------+-------------+-------------+---------+-------------+------+--------+-----------+--------------------+------------------+
only showing top 2 rows
```

```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/silver/participants/*/*.parquet").orderBy("win").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+--------------+-----------+-------------+-------+-------------------+------+-----+------+-------+-----+-----+-----+-----+-----+-----+-----+----------+---------------+-----------+------------+------------------+------------------+-----------+-----+
|match_id  |participant_id|champion_id|champion_name|team_id|individual_position|lane  |kills|deaths|assists|item0|item1|item2|item3|item4|item5|item6|allInPings|itemsPurchased|wardsKilled|wardsPlaced|total_damage_dealt|total_damage_taken|gold_earned|win  |
+----------+--------------+-----------+-------------+-------+-------------------+------+-----+------+-------+-----+-----+-----+-----+-----+-----+-----+----------+---------------+-----------+------------+------------------+------------------+-----------+-----+
|3033314834|4             |61         |Orianna      |100    |MIDDLE             |MIDDLE|8    |8     |4      |3158 |6655 |2031 |1058 |4645 |1056 |3363 |0         |23             |3          |11          |85093             |21842             |10366      |false|
|3033314834|9             |516        |Ornn         |100    |TOP                |TOP   |2    |5     |6      |3047 |3076 |3068 |1031 |1011 |1057 |3340 |0         |17             |2          |8           |98327             |30697             |7836       |false|
+----------+--------------+-----------+-------------+-------+-------------------+------+-----+------+-------+-----+-----+-----+-----+-----+-----+-----+----------+---------------+-----------+------------+------------------+------------------+-----------+-----+
only showing top 2 rows
```

```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/silver/teams_bans/*/*.parquet").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+-------+--------+-----------+
|match_id  |team_id|ban_turn|champion_id|
+----------+-------+--------+-----------+
|3040166917|100    |5       |55         |
|3040295332|200    |9       |20         |
+----------+-------+--------+-----------+
only showing top 2 rows
```

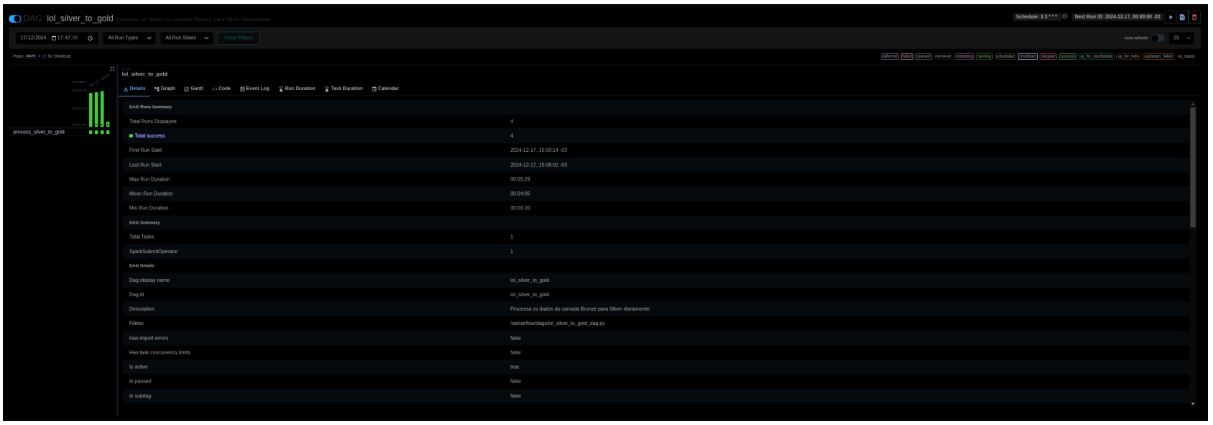
```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/silver/teams_stats/*/*.parquet").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+-------+----+-----------+------------+-----------+
|match_id  |team_id|win |baron_kills|dragon_kills|tower_kills|
+----------+-------+----+-----------+------------+-----------+
|3040250383|200    |true|0          |2           |8          |
|3040202159|200    |true|1          |4           |11         |
+----------+-------+----+-----------+------------+-----------+
only showing top 2 rows
```

Saídas das tabelas Silvers

O script silver_to_gold.py

- DAG que consome a camada silver e realiza agregações.



DAG - Silver To Gold

```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/gold/match_summary/*/*.parquet").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+-------------+-------------+---------+-------------+------+--------+-----------+--------------------+------------------+
|match_id  |game_creation|game_duration|game_mode|game_version  |map_id|queue_id|platform_id|game_start_timestamp|game_end_timestamp|
+----------+-------------+-------------+---------+-------------+------+--------+-----------+--------------------+------------------+
|3040321295|1734439306043|940          |CLASSIC  |14.24.644.2327|11    |420     |BR1        |1734439336861       |1734440277100     |
|3040298464|1734425464429|1207         |CLASSIC  |14.24.644.2327|11    |420     |BR1        |1734425509870       |1734426716776     |
+----------+-------------+-------------+---------+-------------+------+--------+-----------+--------------------+------------------+
only showing top 2 rows
```

```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/gold/player_performance/*/*.parquet").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+--------------+-----------+-------------+-------+-----------+------------+-------------+-----------------+-----------------+-----------------+-------------+------------+--------------+-----------+----------+
|match_id  |participant_id|champion_id|champion_name|team_id|total_kills|total_deaths|total_assists|total_damage_dealt|total_damage_taken|total_gold_earned|wards_placed|wards_killed|items_purchased|win_status|
+----------+--------------+-----------+-------------+-------+-----------+------------+-------------+------------------+------------------+-----------------+------------+------------+---------------+----------+
|3034445102|3             |64         |LeeSin       |200    |6          |5           |2            |165393            |31323             |9173             |18          |3           |26             |false     |
|3037609799|2             |16         |Soraka       |100    |2          |4           |16           |14612             |12079             |7581             |27          |5           |21             |true      |
+----------+--------------+-----------+-------------+-------+-----------+------------+-------------+------------------+------------------+-----------------+------------+------------+---------------+----------+
only showing top 2 rows
```

```
try:
    spark.read.parquet("hdfs://hadoop-namenode:8020/datalake/gold/team_performance/*/*.parquet").show(2,False)
except AnalysisException as e:
    print(f"Erro do Spark ao acessar os dados: {e}")
```

```
+----------+-------+----------+-----------------+-----------------+-----------------+
|match_id  |team_id|win_status|total_baron_kills|total_dragon_kills|total_tower_kills|
+----------+-------+----------+-----------------+-----------------+-----------------+
|3034483961|200    |false     |0                |2                |6                |
|3038387397|200    |false     |0                |1                |2                |
+----------+-------+----------+-----------------+-----------------+-----------------+
only showing top 2 rows
```

Saídas das tabelas Gold

3.3. Estrutura do Data Lake

- Camada Bronze: Dados brutos conforme recebidos do Kafka.
- Camada Silver: Dados transformados e normalizados, por exemplo, extraindo estatísticas individuais dos jogadores.
- Camada Gold: Dados agregados, como KPIs (ex.: total de kills, mortes e assistências por jogador).

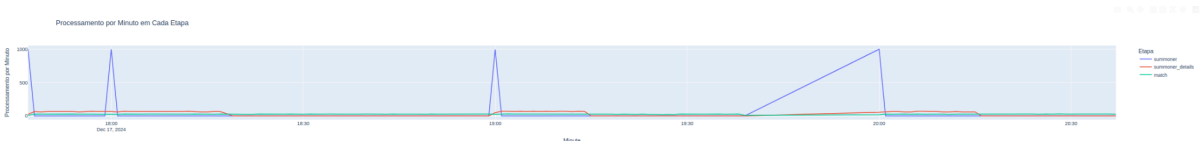
3.4. Monitoramento

Métricas:

- Kafka, HDFS e Spark expõem métricas para o Prometheus, que são visualizadas no Grafana.

Hadoop Métricas

Kafka Métricas



Throughput - Etapas

3.5. LGPD

Para os dados que trabalhamos na API, não temos dados sensíveis.
Caso existisse dados sensíveis poderíamos utilizar os métodos:

- Generalização
- Supressão
- K-Anonimidade
- Tokenização

4. Melhorias

4.1. Escalabilidade:

- Implementar particionamento no Kafka e paralelismo no Spark para suportar maior volume de dados.
- Configurar múltiplos nós no cluster HDFS para maior capacidade de armazenamento.

4.2. Observabilidade Avançada:

- Adicionar rastreamento distribuído com OpenTelemetry para monitorar o tempo de processamento em cada componente do pipeline.

4.3. Governança de Dados:

- Adicionar políticas de retenção em cada camada.
- Adicionar expurgo dos dados

4.4. Segurança:

- Configurar autenticação e autorização no Kafka.
- Criptografar os dados sensíveis armazenados no Data Lake.
- Chaves de API devem ser armazenadas em um cofre de senhas.

5. Considerações Finais

A solução desenvolvida apresenta um pipeline robusto para ingestão, processamento e armazenamento de dados em tempo real, com suporte a monitoramento e observabilidade. As tecnologias utilizadas garantem escalabilidade e flexibilidade, atendendo às demandas do case.

Com as melhorias sugeridas, a solução pode ser ainda mais eficiente e confiável, garantindo maior governança e capacidade de lidar com volumes crescentes de dados.