Scaling Warehouse with Flink, Parquet & Kubernetes

Aditi Verma &
Ramesh Shanmugam

Aditi Verma

Sr Software Engineer

@aditiverma89 <averma@branch.io>

Ramesh Shanmugam

Sr Data Engineer

@rameshs01 <rshanmugam@branch.io>

branch

Agenda

- Background
- Moving data with Flink @ Branch
- Scale & Performance
- Flink on Kubernetes
- Auto Scaling & Failure Recovery

branch

12B requests per day (+70% y/y)

3B user sessions per day

50 TB of data per day

200K events per second

60+ Flink pipelines

5+ Kubernetes cluster

branch

druid

Flink

HIVE

APACHE Spark™

presto

kubernetes

Apache Airflow

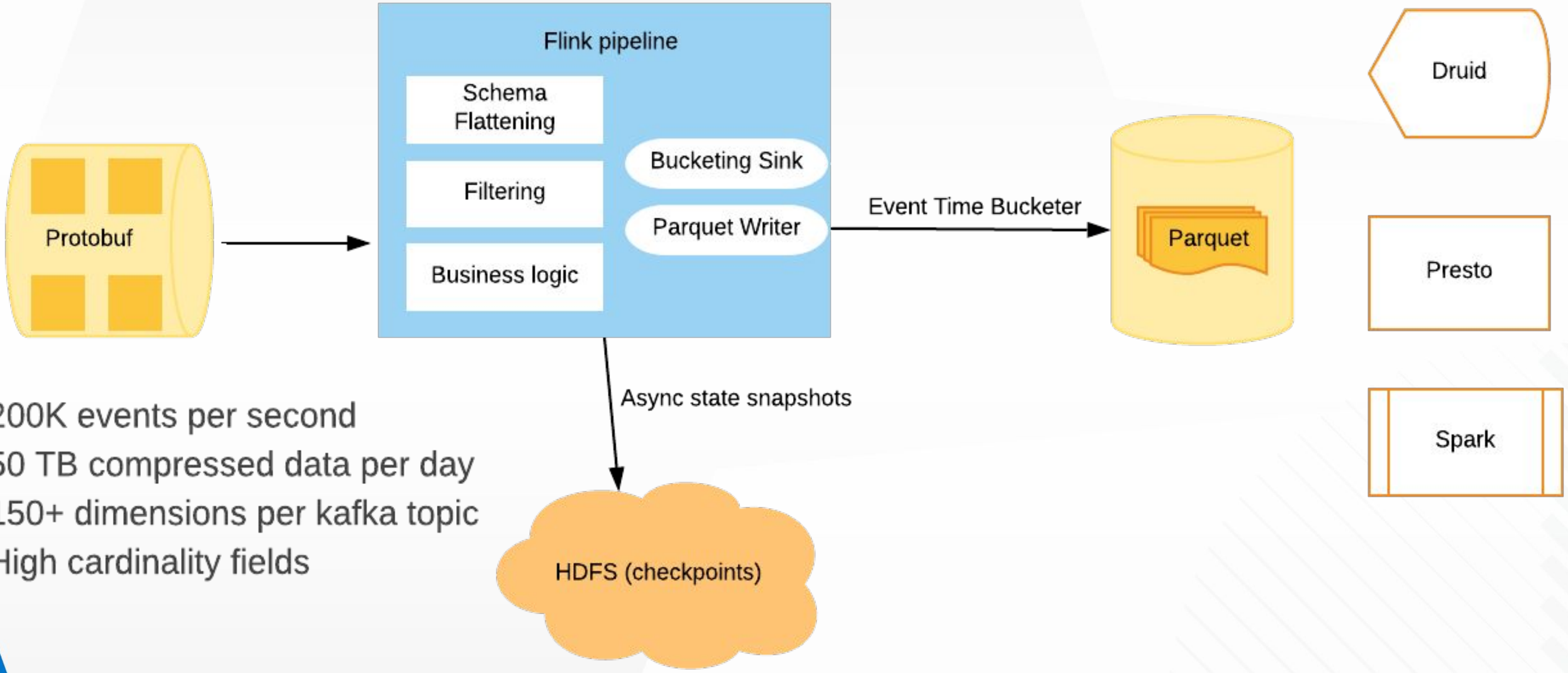
Moving data with Flink @ Branch

“Life is 10% what happens to you and 90% how you react to it.”
— Charles R. Swindoll

Receive information
Process it
React to it

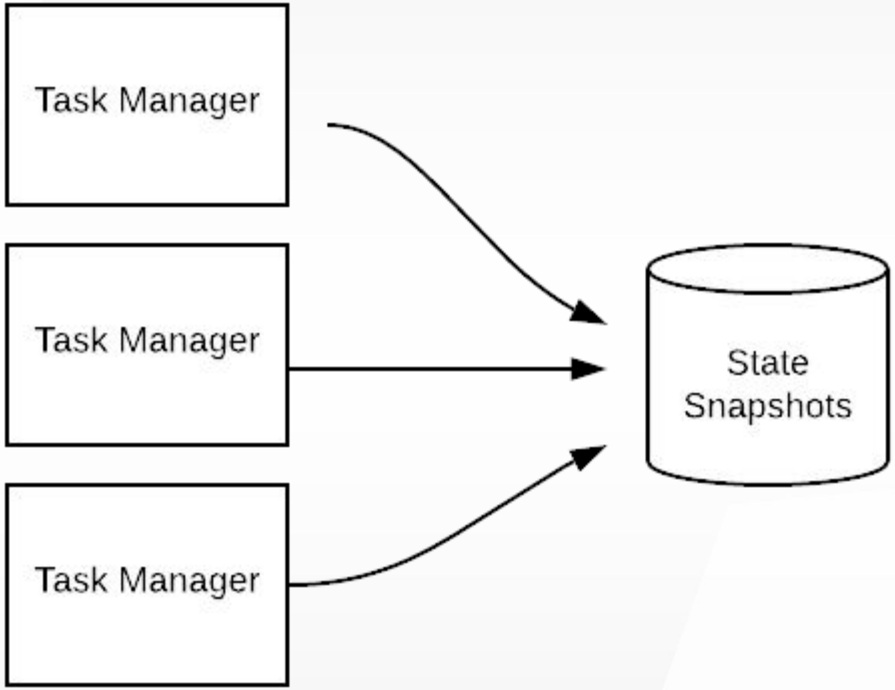
FAST!!

Flink @ Branch



200K events per second
50 TB compressed data per day
150+ dimensions per kafka topic
High cardinality fields

State Backend



- Relatively small state backend
- File system backed state

Parquet

- Higher compression
- Read heavy data set: ingested to Druid and Presto (3M+ queries/day)
- Avro data format
- Memory intensive writes

Writing parquet with Flink

Two approaches:

1) Close the file with checkpointing

Writing parquet with Flink

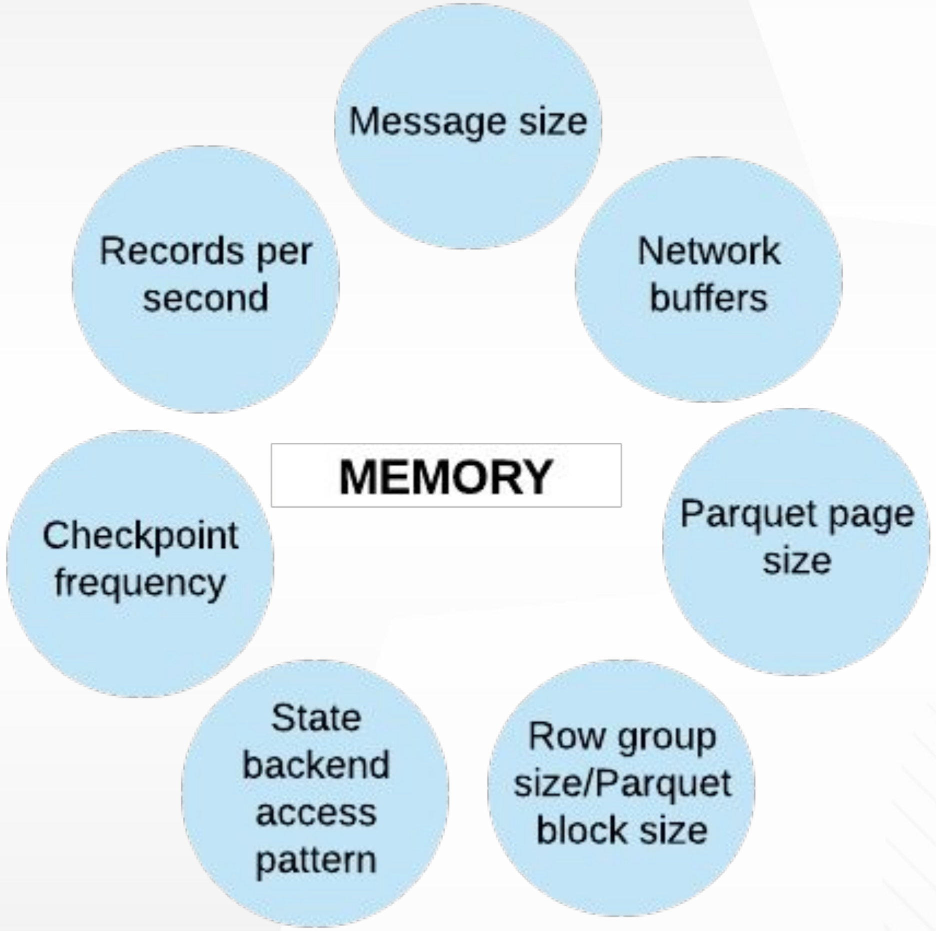
Two approaches:

a) ~~Close the file with checkpointing~~
b) Bucketing file sink
   i) Configured with custom event-time bucketer, parquet writer and batch size
   ii) Files are rolled out with a timeout of 10 min within a bucket

Performance and Scale

- 100% traffic increase each year
- Higher parallelism impacts application performance and state size
- Kafka partitions < Flink parallelism requires rebalance on the input stream
- Task manager timeouts

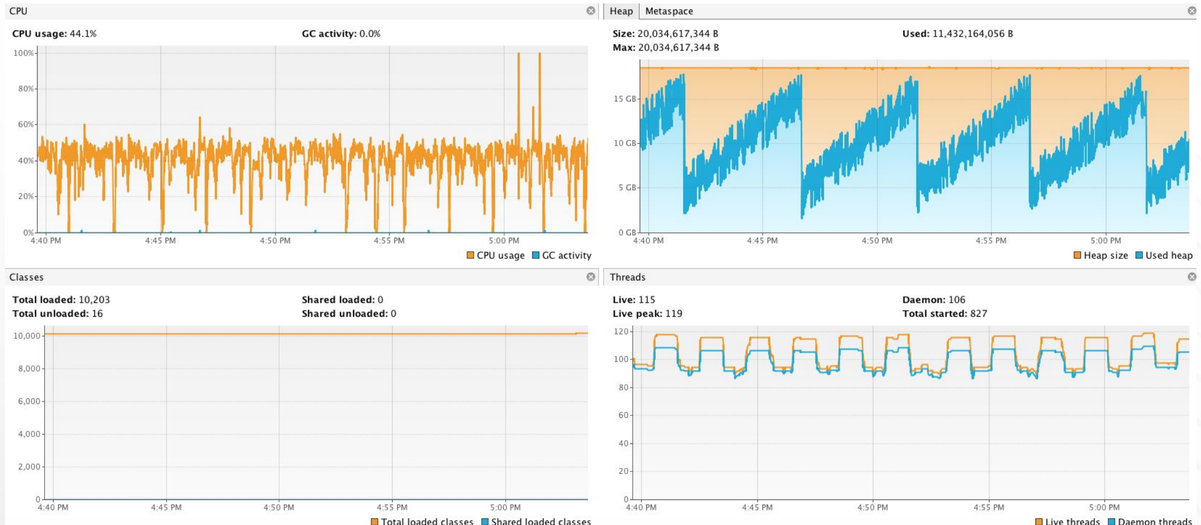
Performance
Scalability
Stability


MEMORY
Message size
Network buffers
Parquet page size
Row group size/Parquet block size
State backend access pattern
Checkpoint frequency
Records per second

Analyzing memory usage

- ❖ Network Buffers
- ❖ Memory Segments
- ❖ User code

- ❖ Memory and GC stats
- ❖ JVM parameters



CPU

CPU usage: 44.1% **GC activity:** 0.0%

Heap | Metaspace

Size: 20,034,617,344 B **Used:** 11,432,164,056 B
Max: 20,034,617,344 B

Classes

Total loaded: 10,203 **Shared loaded:** 0
Total unloaded: 16 **Shared unloaded:** 0

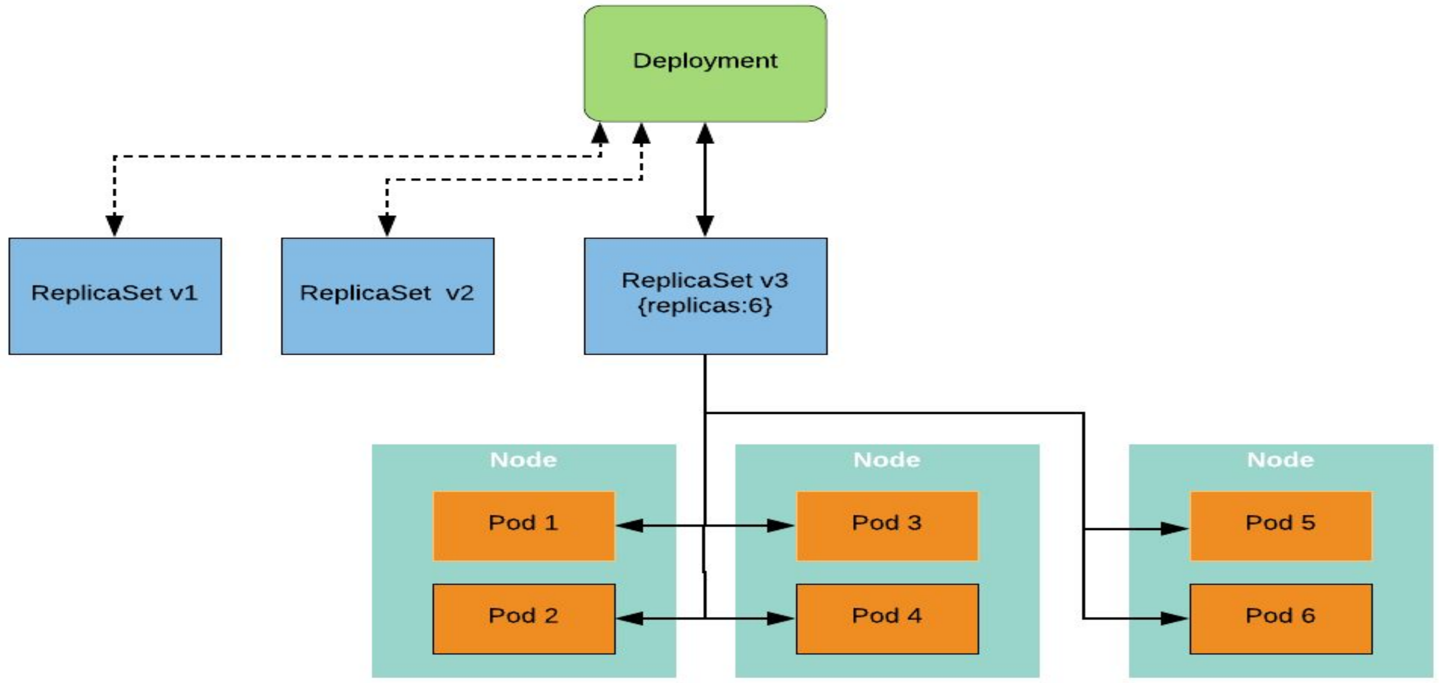
Threads

Live: 115 **Daemon:** 106
Live peak: 119 **Total started:** 827

Containerizing Flink - Mesos

- Longer start-up time on Mesos
- Moved to containerizing Flink application on Kubernetes
- Kubernetes is resource oriented, declarative

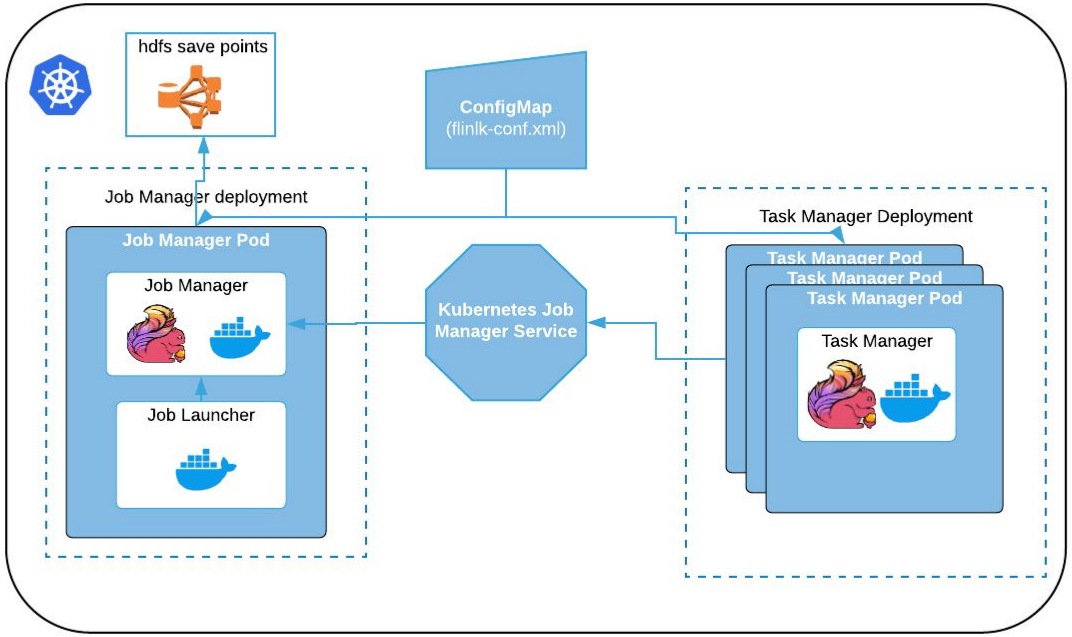
Kubernetes Terms



Flink

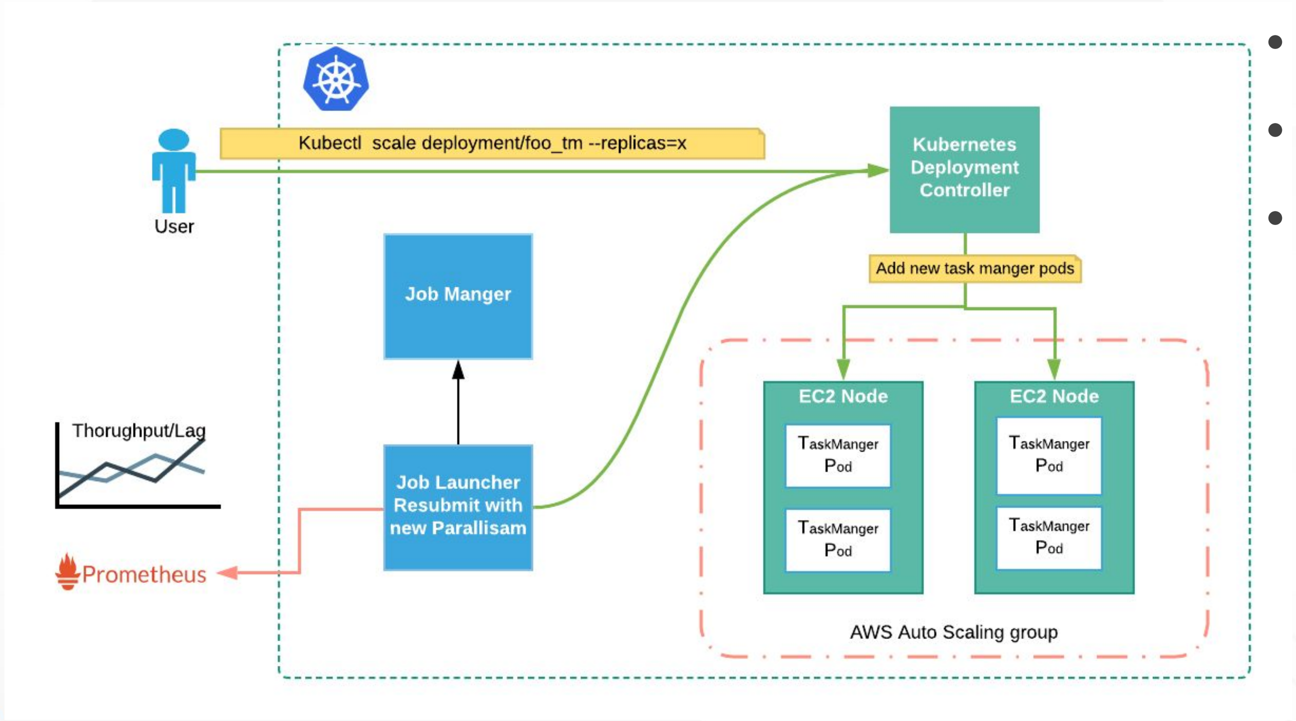
⊕

Flink on Kubernetes @ Branch



- Single job per cluster
- Docker image
  - flink image - Task manager + job manager
  - Job launcher - custom launcher + job jar
- Job launcher
  - Application jar
  - Uploads jar
- Config map - flink config.xml
  - jobmanager.rpc.address

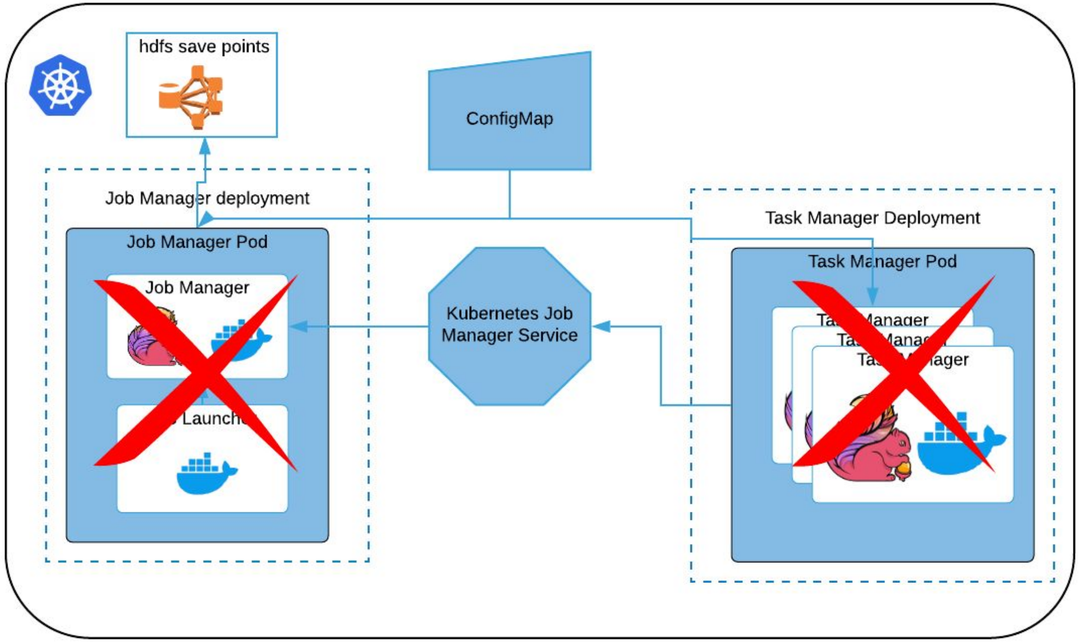
Auto Scaling



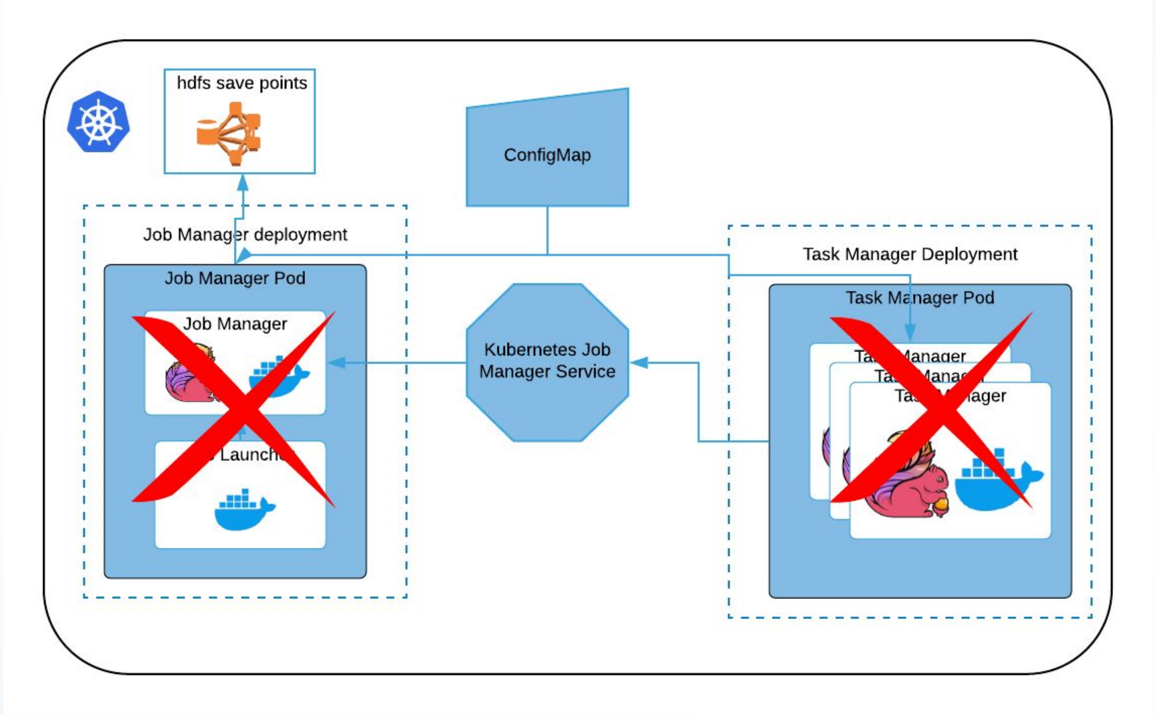
- When & How much scale
  - Auto - Joblauncher
- Scale
  - Replica Set
- Flink job with new parallelism

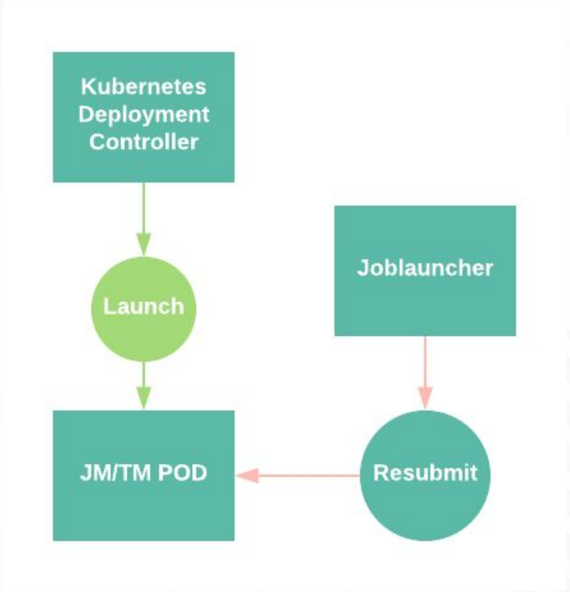
Failure Recovery

Job / Task Manager Goes Down?

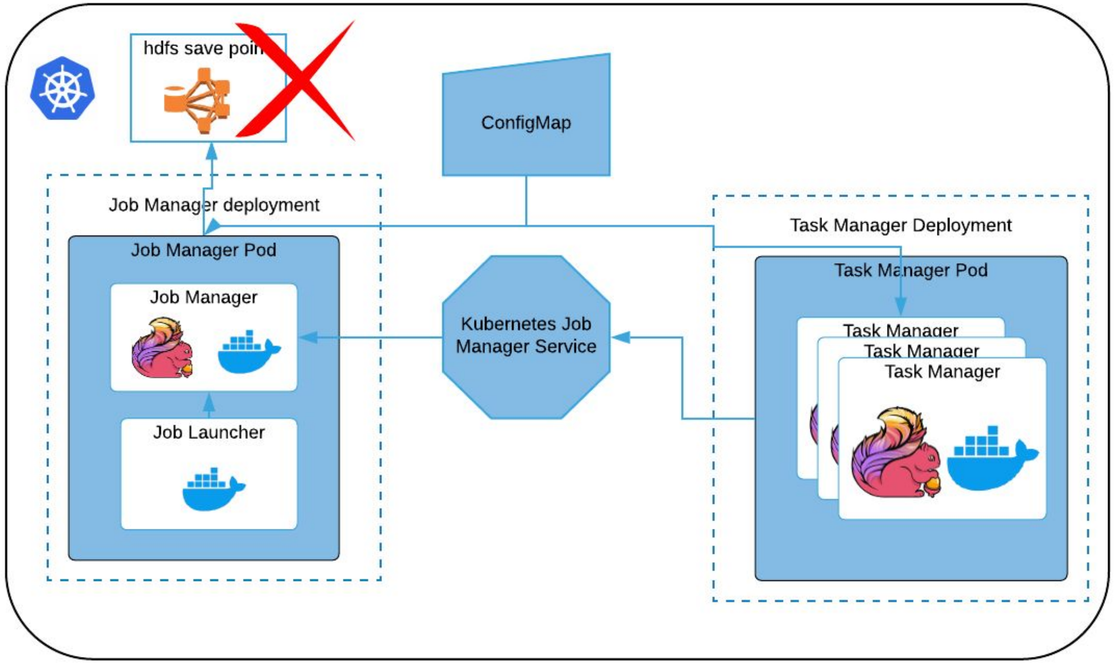


Job / Task Manager Goes Down?



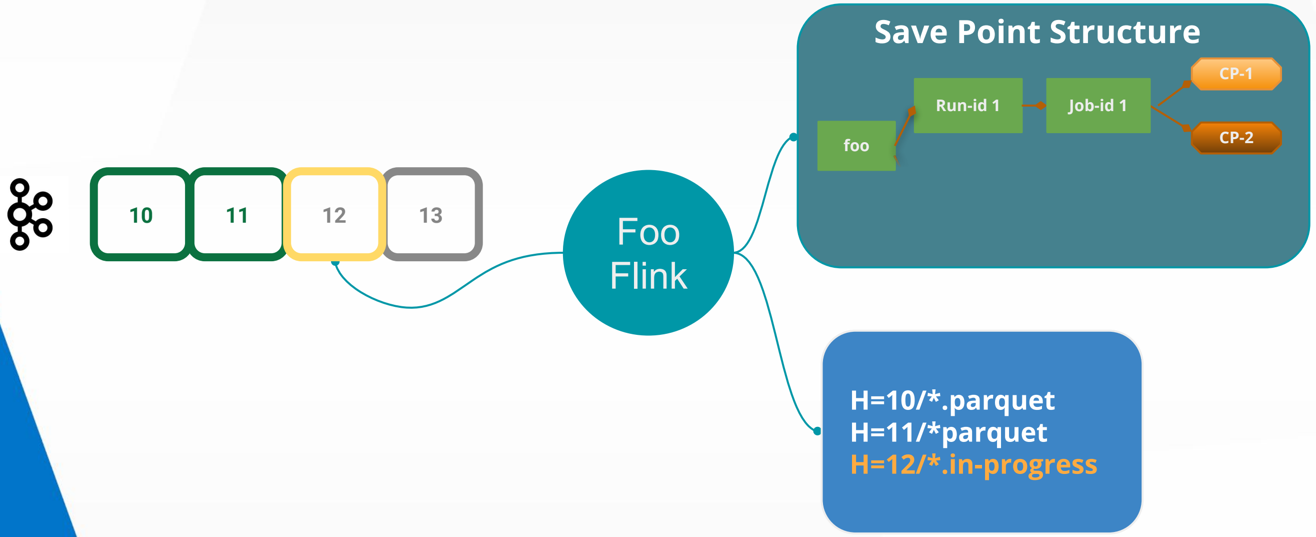


Savepoint Failure



- Reasons
  - Truncation
  - Schema mismatch
  - Hdfs outage

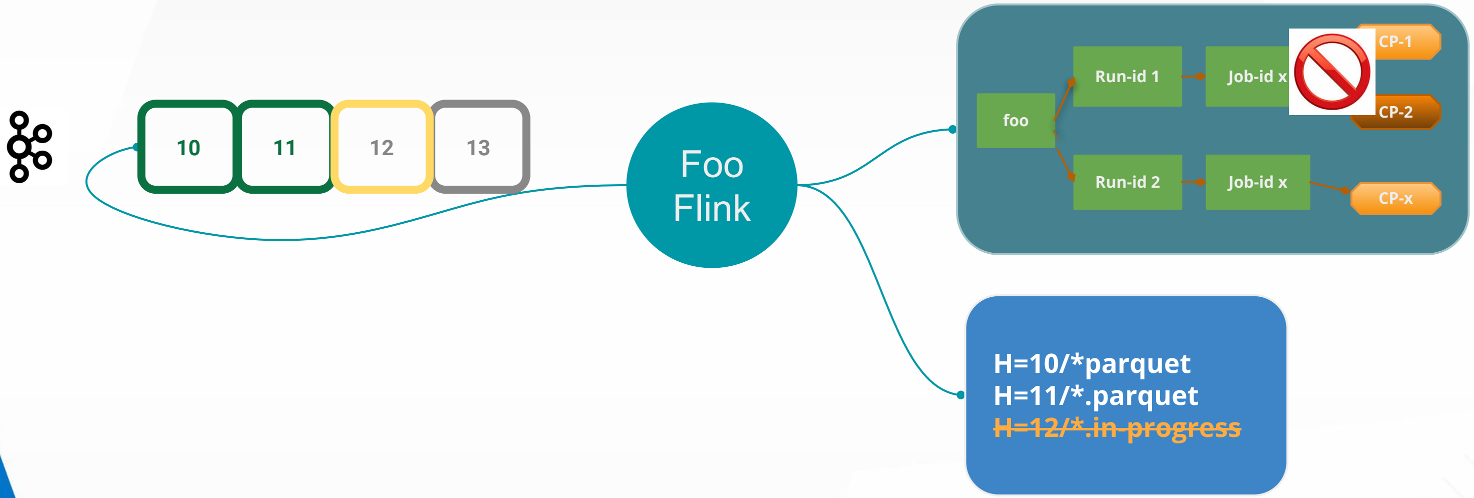
Savepoint Structure



H=10/*.parquet
H=11/*parquet
H=12/*.in-progress

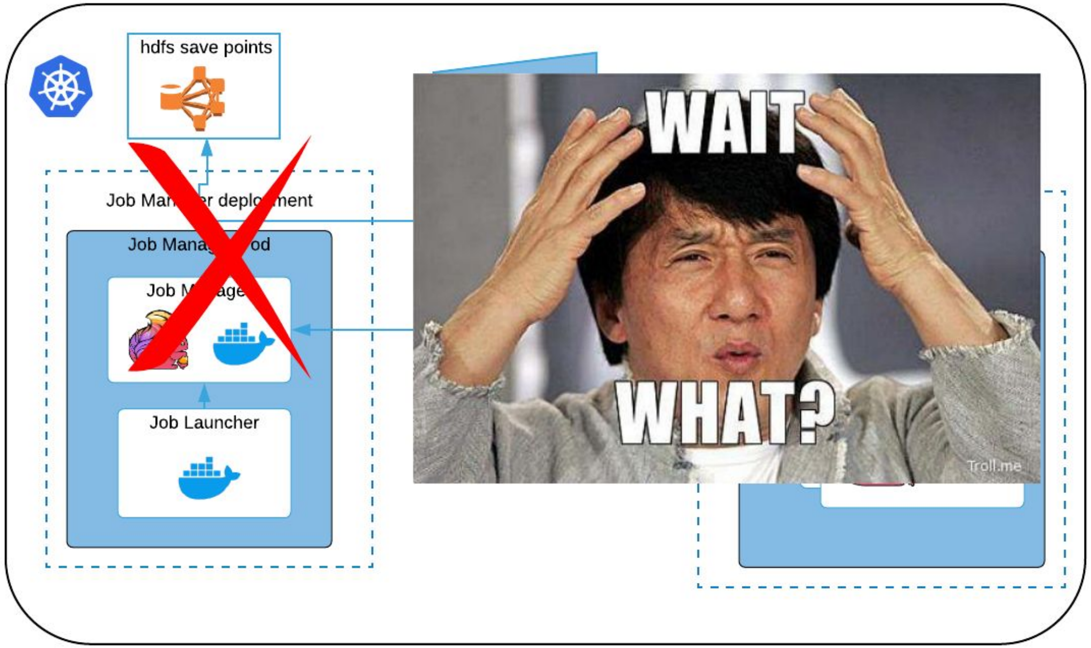
- job/run-id/flink-job-id/cp-x
- Run id - incremental number
- Job id - flink job name

Savepoint failure recovery



H=10/*parquet
H=11/*.parquet
~~H=12/*.in-progress~~

Auto Recovery does not work?



- Continuous monitoring and proper alerts
- start job from latest offset
- Have different backfill route

Next Steps....

- Parquet memory consumption (when too many buckets open)
  - Window + Rocks db => Parquet
  - Two stage process
    - row oriented streaming
    - batch to convert columnar

Q & A

branch