Spring Batch

Framework for Batch Processing

What is Batch Processing?

Processing of data without interaction or interruptions.

Use case -

- Generation of Bank Statements
- ETL processing - transferring of data
- Processing Big Data
- Research and Analysis

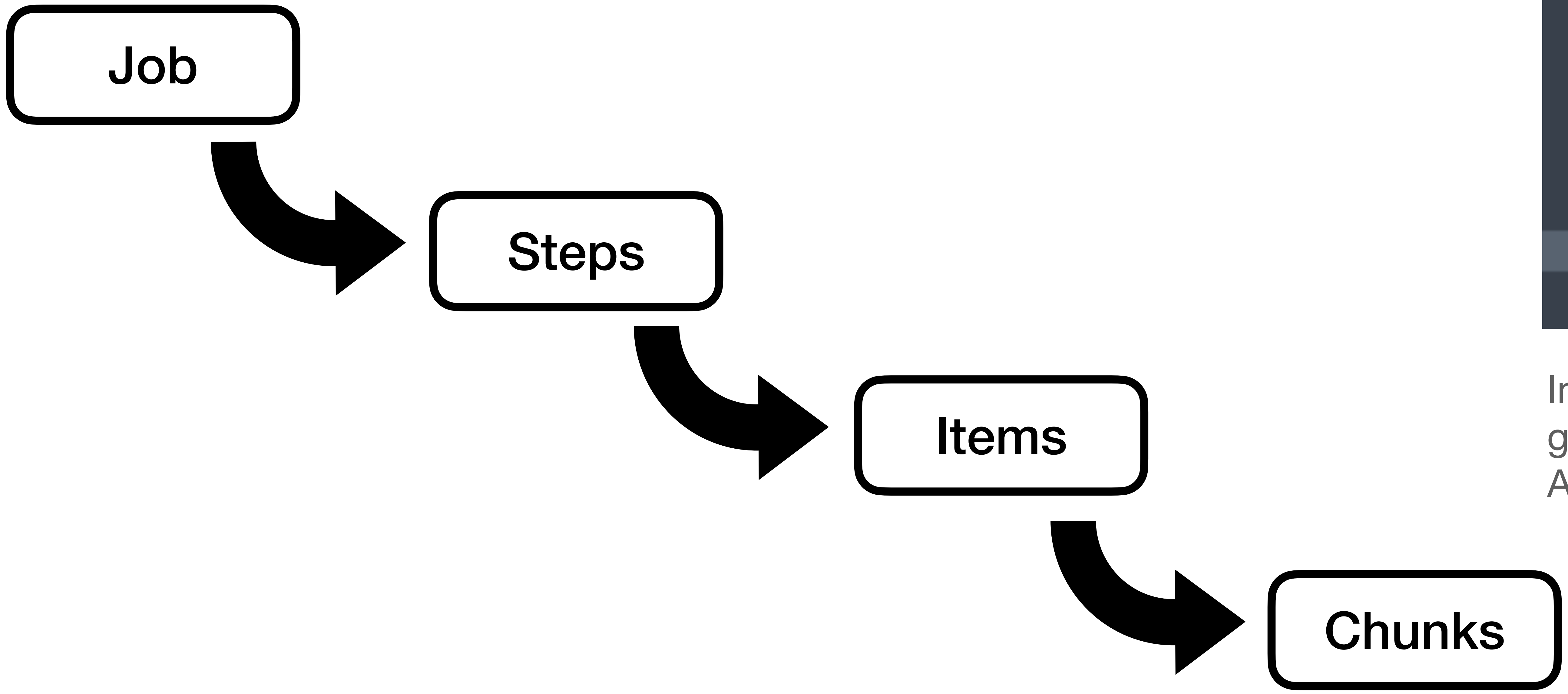
Spring Batch

A lightweight, comprehensive batch framework designed to enable the development of robust batch applications vital for daily operations of enterprise systems.

WHY?

- Restartability - easy to restart from where it failed
- Start, restart, skip, retry capabilities
- Different Readers and Writers - support of JMS, JDBC, CSV, File, Hibernate and more
- Chunk Processing
- Transaction management - rollback/skip capabilities
- Parallel processing

HOW?

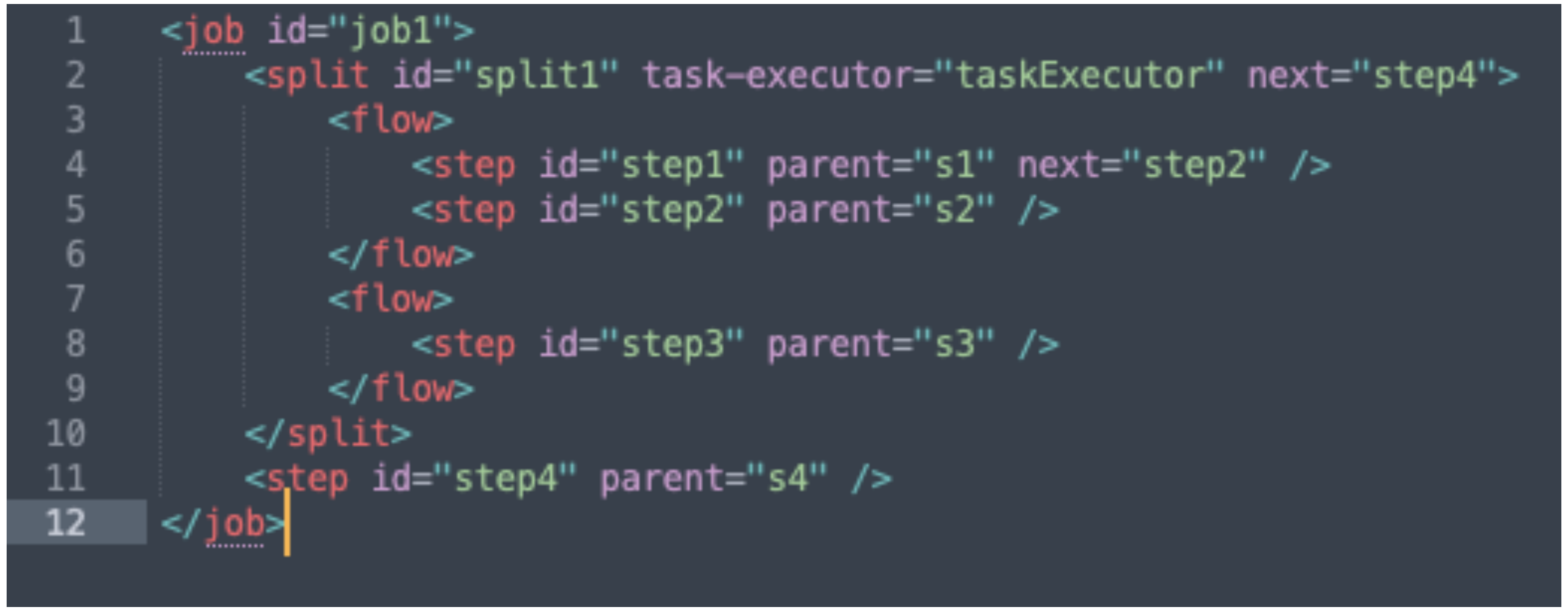


Job

Steps

Items

Chunks



```
<job id="job1">
    <split id="split1" task-executor="taskExecutor" next="step4">
        <flow>
            <step id="step1" parent="s1" next="step2" />
            <step id="step2" parent="s2" />
        </flow>
        <flow>
            <step id="step3" parent="s3" />
        </flow>
    </split>
    <step id="step4" parent="s4" />
</job>
```

In this example, we are splitting the task and using **<flow>**, the processing is going to be parallel because of **taskExecutor**
And after that step4 will execute.

A Job in a spring batch is a sequence of steps. Each step can be configured with -

- **Next** —> next step to execute
- **Tasklet** —> task or chunk to execute (It can be configured with ItemReader, ItemWriter, ItemProcessor)
- **Decision** —> decide which steps need to execute

Important points to remember

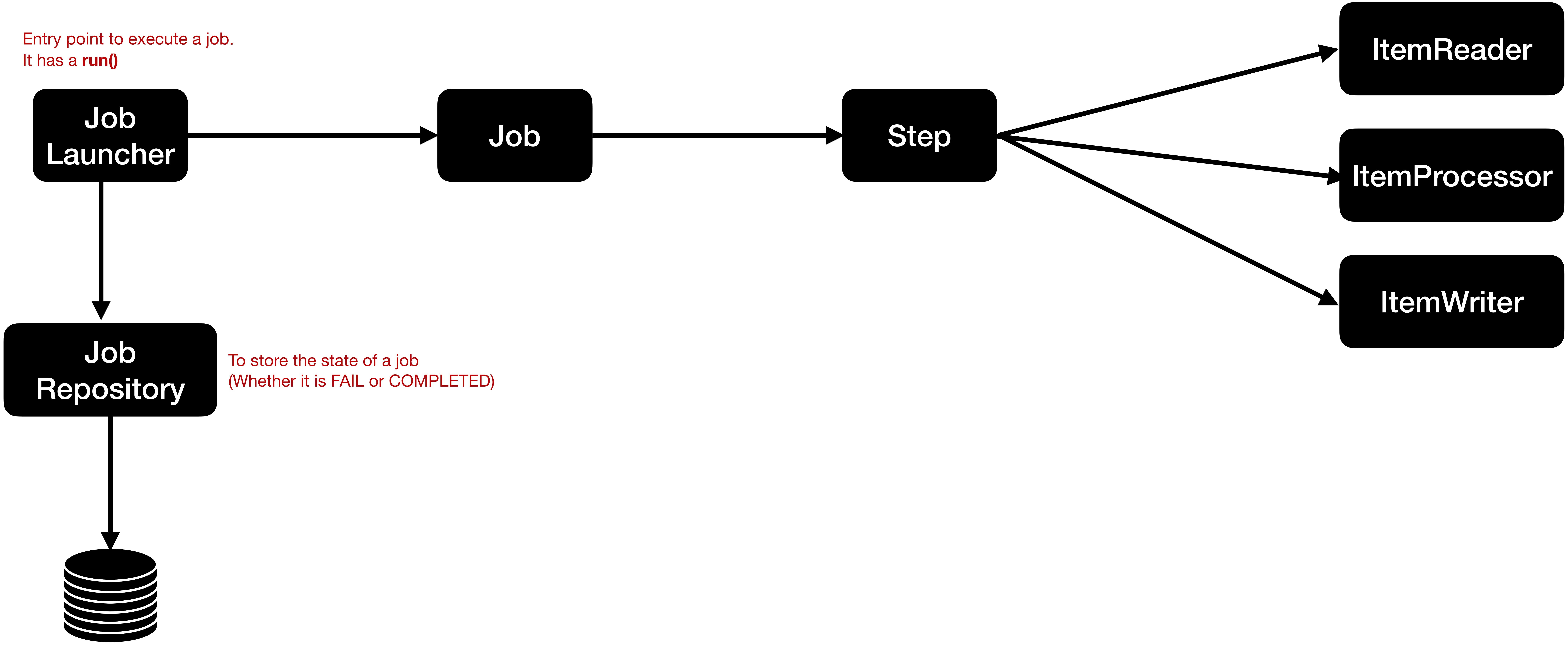
- A Job Launcher can be used to execute a Spring Batch job. It can also be launched/scheduled in a web container as well.
- Spring Batch does not use any Schedular by design.
- Each execution of a job is called a Job Instance. Each Job Instance is provided with an execution id which can be used to restart the job (if needed).
- Job can be configured with params which can be passed to it from the Job Launcher.

How to Scale Batch Processing

- Multi-threaded steps
- Parallel Steps
- Remote Chunking
- Remote Partitioning - it follow Master-Slave flow

Reference - https://docs.spring.io/spring-batch/docs/current/reference/html/scalability.html#:~:text=Remote%20Chunking,-In%20remote%20chunking&text=The%20manager%20is%20an%20implementation,to%20the%20middleware%20as%20messages

Spring Batch components & Architecture flow



Entry point to execute a job.
It has a **run()**

Job Launcher

Job

Step

ItemReader

ItemProcessor

ItemWriter

Job Repository

To store the state of a job
(Whether it is FAIL or COMPLETED)