Introduction à l’ événementiel

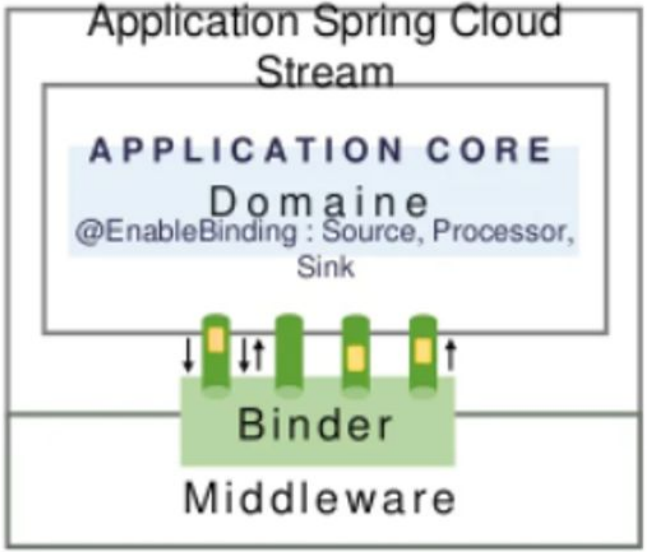
Spring Batch & Spring Cloud Stream et Kafka

Général

- Initiation à l'événementiel
- Écosystème du cours / projet
  - Spring Boot
  - Spring Data
  - Spring Batch
  - Spring Cloud
  - Kafka

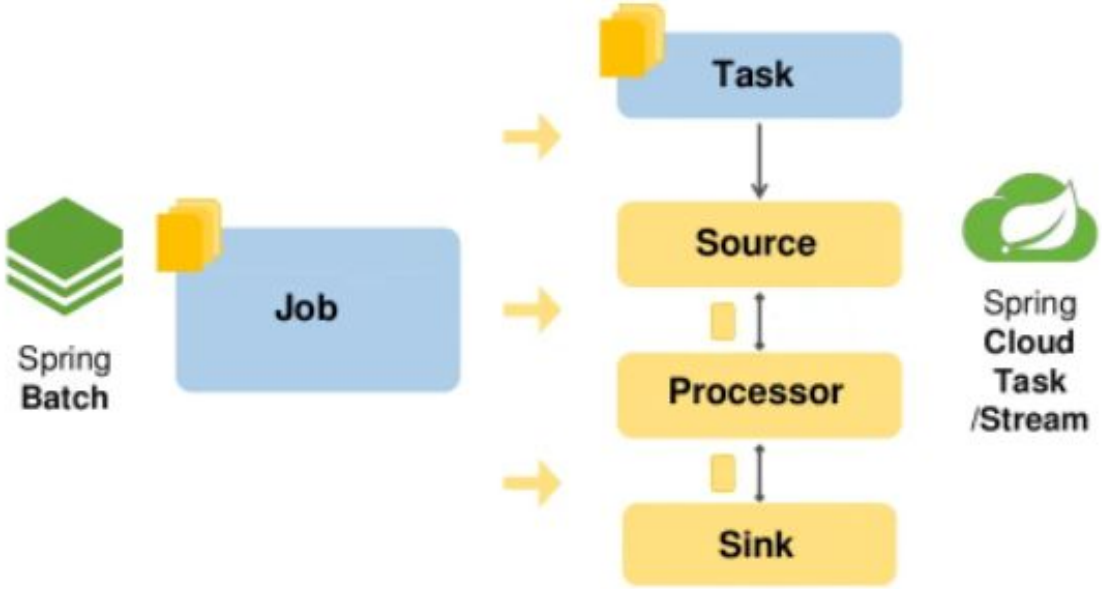
Écosystème

Écosystème

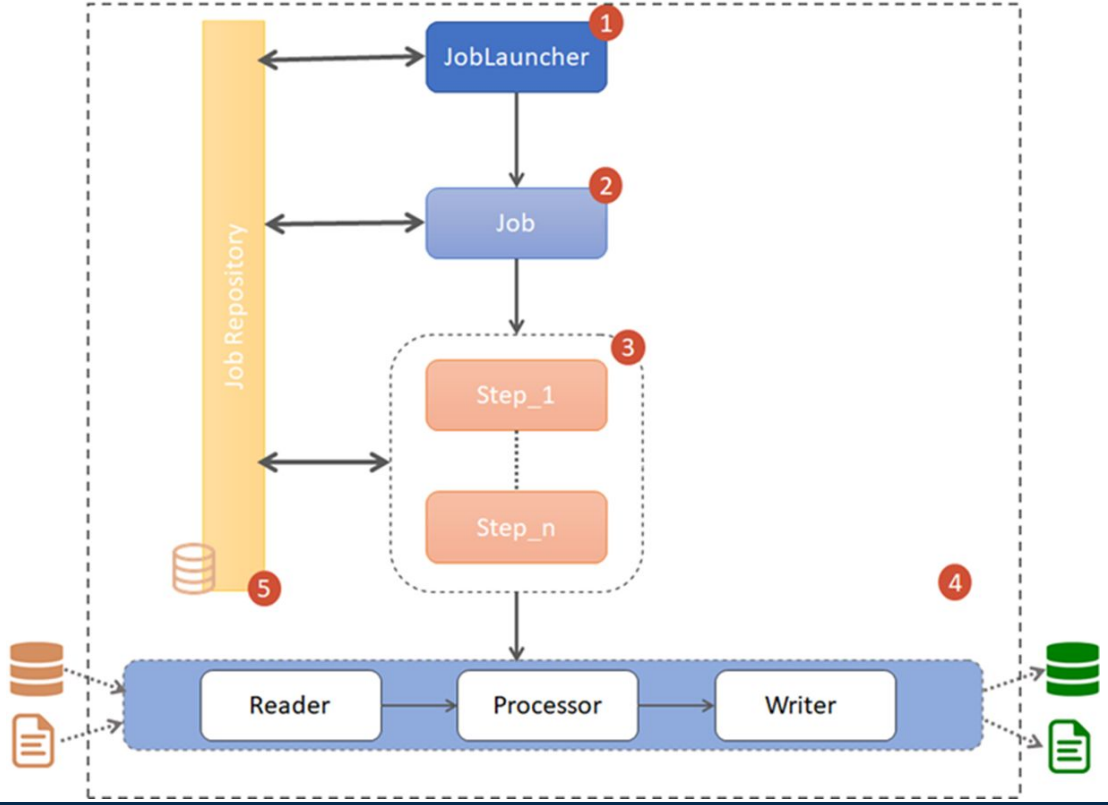


Kafka Binder

Processus

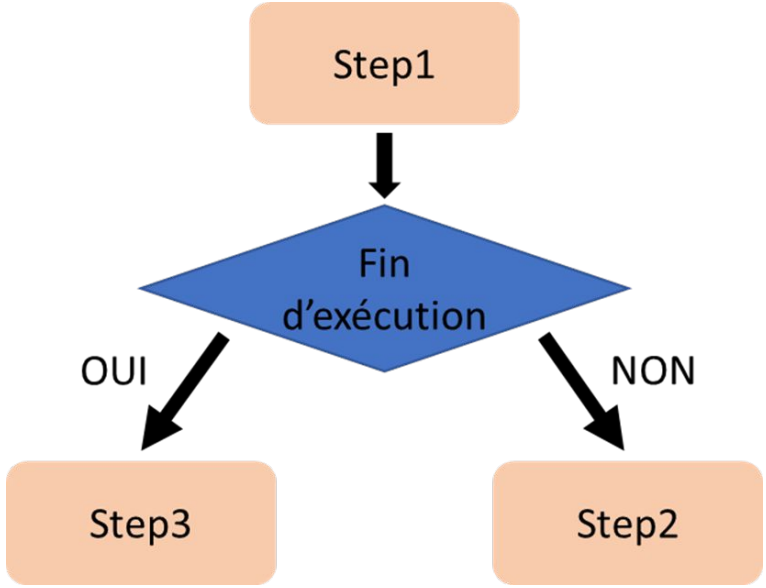

Task
Source
Processor
Sink
Job
Spring
Batch
Spring
Cloud
Task
/Stream

Spring Batch - Architecture



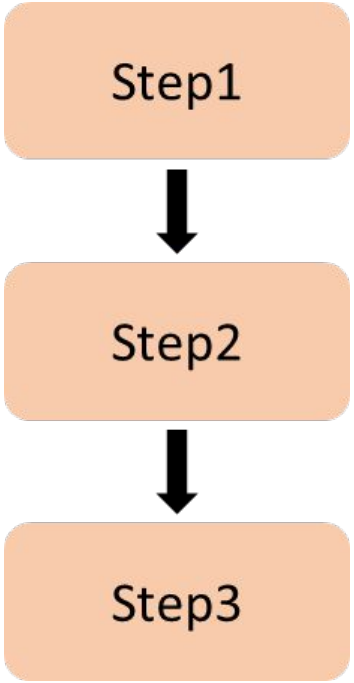
Job - Définition / Création

```
@Bean
public Job jobExample() {
   return _jobBuilder.get(" jobExample")
         .start(step1())
         .on("*").to(step3())
         .from(step1()).on("FAILED" ).to(step2())
         .end()
         .build();
}
```



Job - Définition / Création

```
@Bean
public Job jobExample() {
   return _jobBuilder.get(" jobExample")
         .start(step1())
         .nextstep2())
         .next(step3())
         .end()
         .build();
}
```



Création d’un Step

```
@Bean
public Step step1() {
 return _jobBuilder.get("step1")
        .reader(new ReaderExample())
        .processor(new ProcessorExample())
        .writer(new WriterExample())
        .build();
}
```

```
@Bean
public Step step2() {
 return _jobBuilder.get("step2")
        .tasklet( new TaskletExample())
        .build();
}
```

Création d'un Step - Item<>

```
public interface ItemReader<T> {
   T read() throws Exception, UnexpectedInputException, ParseException, NonTransientResourceException;
}
```

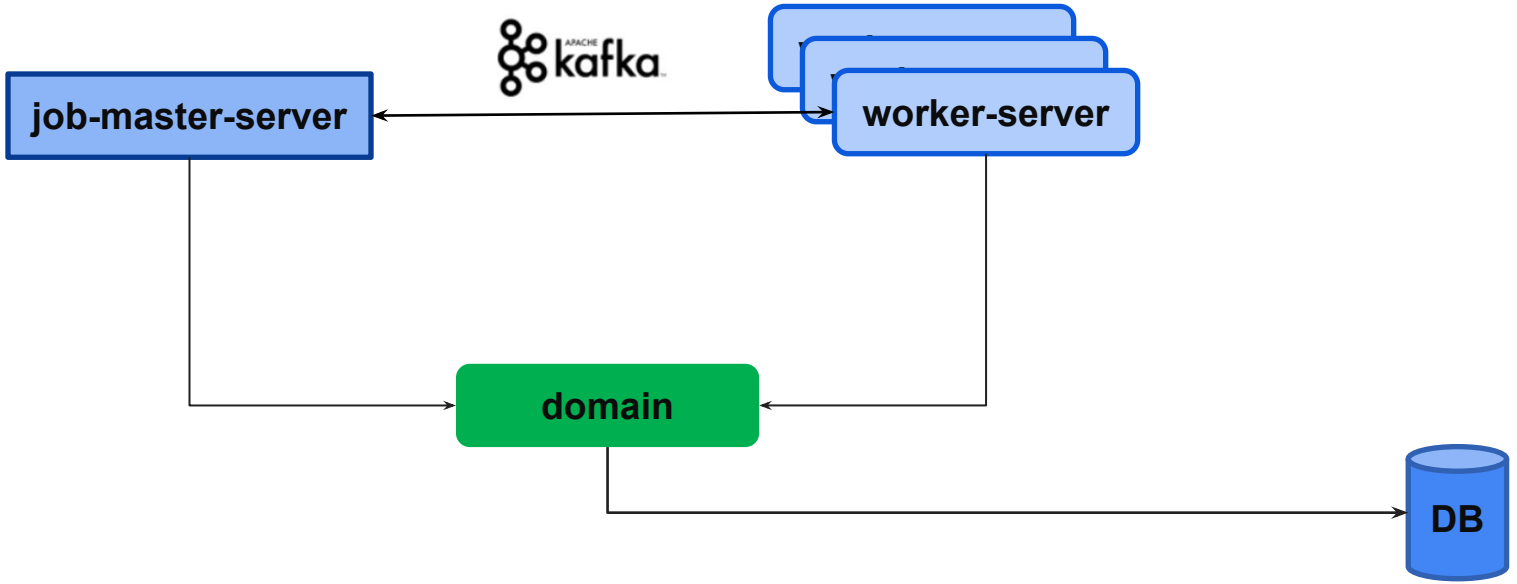
```
public interface ItemProcessor<I, O> {
   O process(I obj) throws Exception;
}
```

```
public interface ItemWriter<T> {
   void write(List<? Extends T> items) throws Exception;
}
```

Création d’un Step - Tasklet

```
public Step TaskletExample implements Tasklet {
   @Override
   public RepeatStatus execute(final StepContribution stepContribution, final ChunkContext chunkContext) {
      return RepeatStatus.FINISHED;
   }
}
```

Architecture du projet



Architecture du projet

- domain
  - Configuration de la base
  - Objets métiers
- job-master-server : permet de gérer le lancement des Jobs
  - Définition de création d‘un Job
  - JobExecutionSink qui permet d’écouter des messages du « worker-server »
- worker-server : permet d’exécuter les Jobs
  - JobProcessor qui permet d’écouter / d’envoyer des messages au « job-master »

Fonctionnement

- Lancement du serveur **Kafka**
- Lancement du serveur **domain**
- Lancement du serveur **server-queue**
- Lancement de n serveurs **worker-service**

Spring Boot - Spring Cloud- Kafka