Apache Flink на примере задачи дедупликации

Бобряков Александр

@appp_master

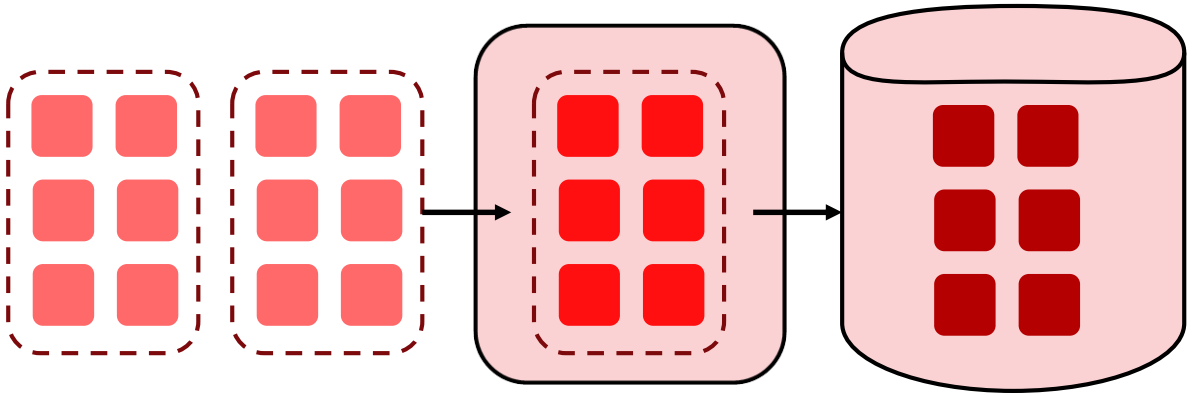
На какие вопросы ответим

- Как построить Flink-задачу по дедупликации?
- Как защититься от сбоев Flink-кластера? Exactly once?
- Как эволюционировать приложение?
- Как обеспечить непрерывную работу Flink-задания?
- Как настроить приложение под высокую нагрузку? RocksDB
- Оптимизации при работе с большим состоянием
- Как ускорить работу Flink-задания?

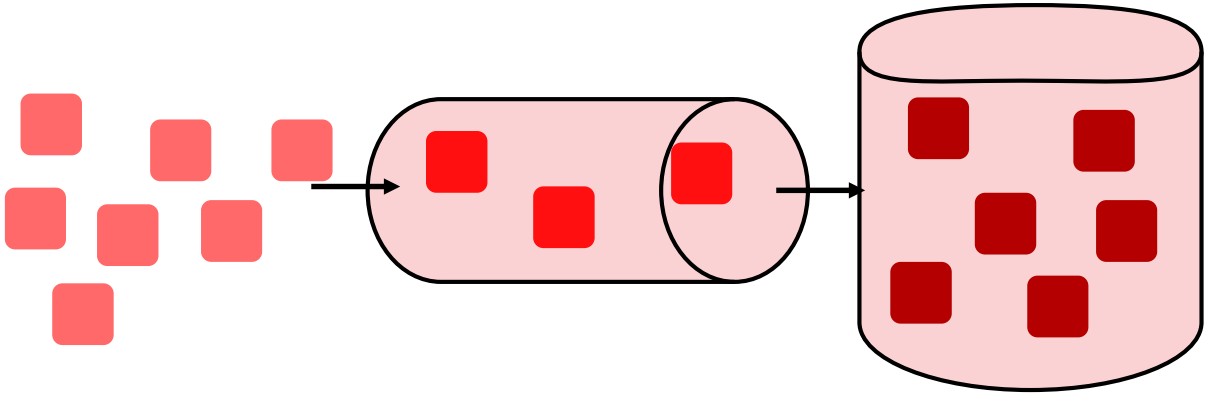
Потоковая обработка данных

Пакетная и потоковая обработка

- Данные собираются с течением времени
- После сбора данные отправляются на обработку

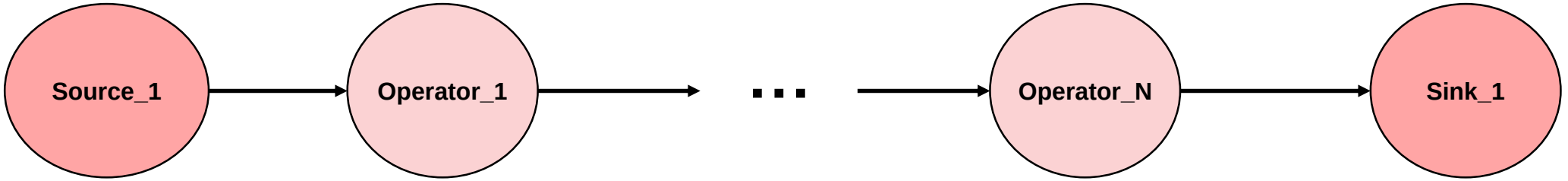


- Потоки данных поступают непрерывно
- События обрабатываются в режиме реального времени

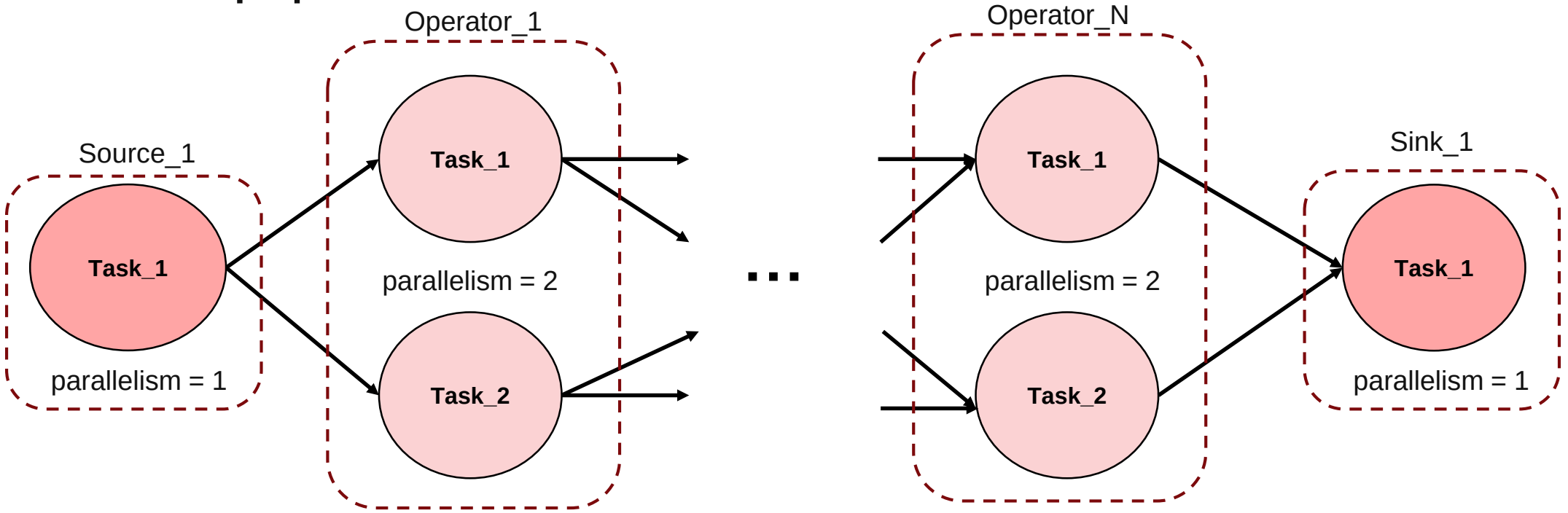


Графы потоков

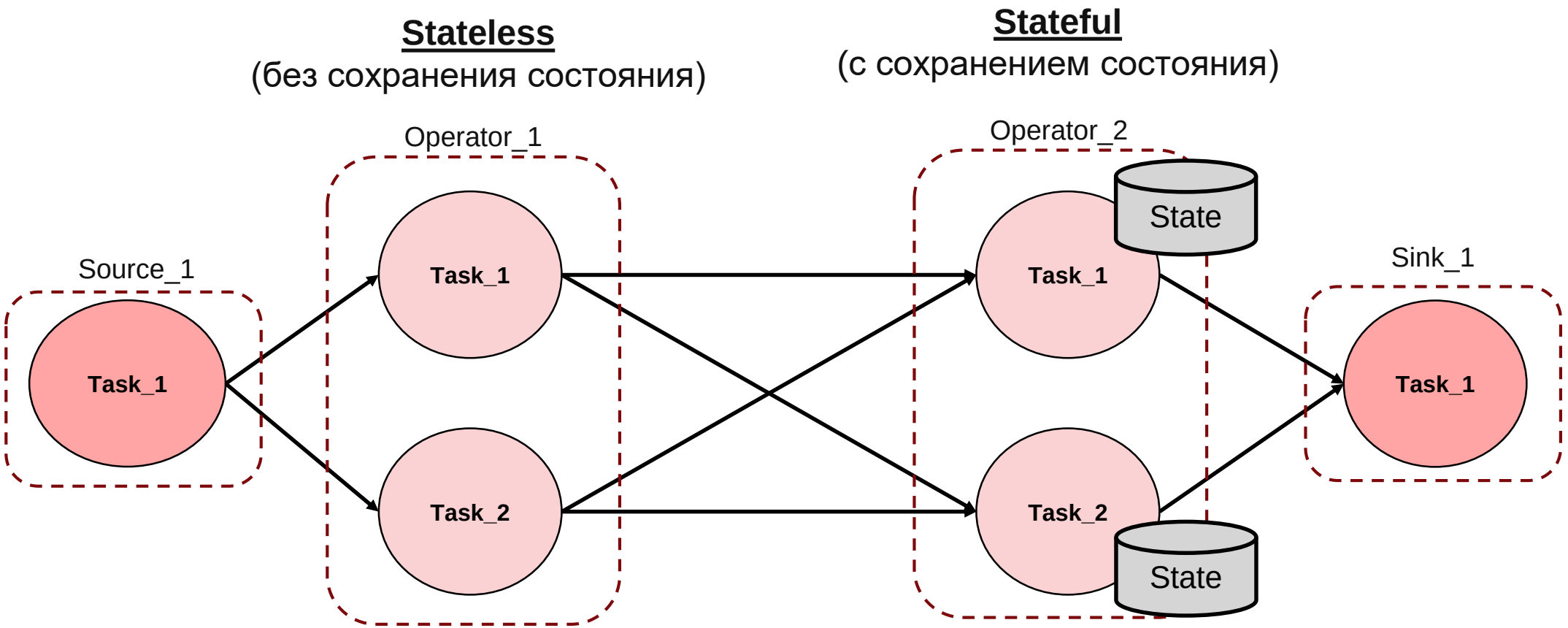
Логический граф



Физический граф

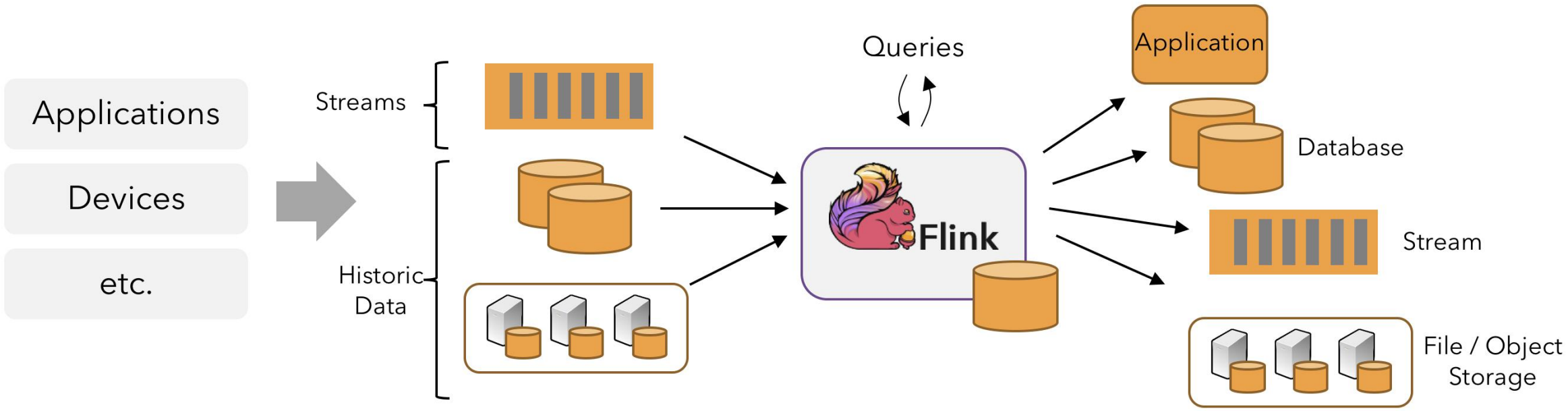


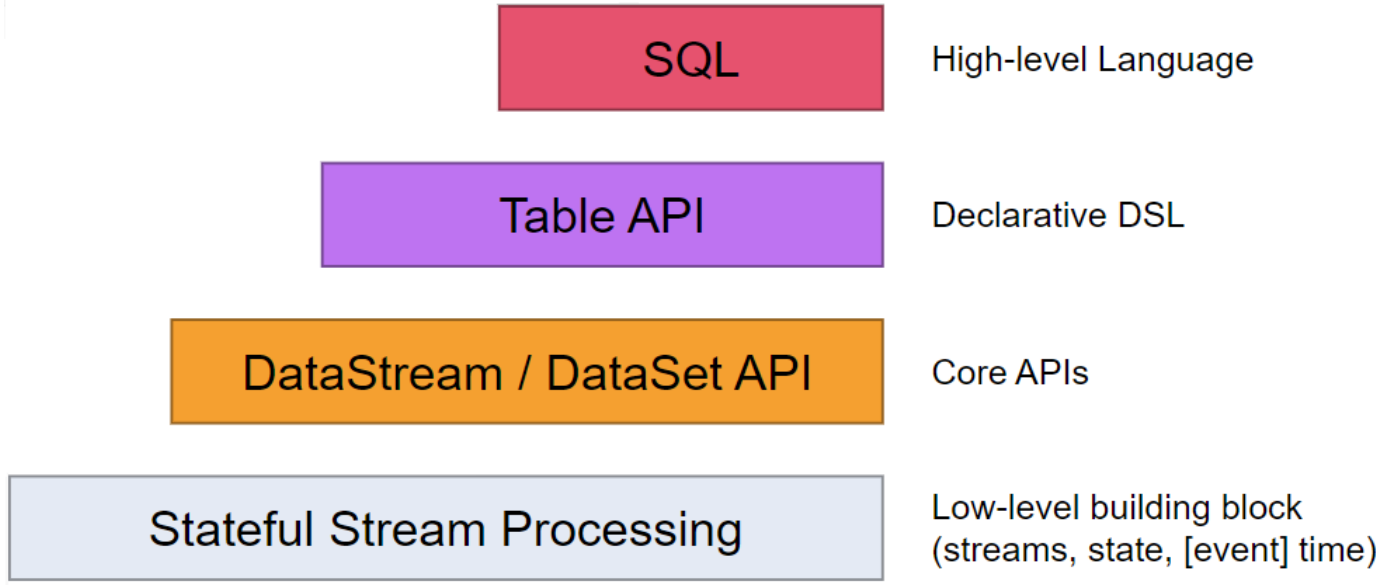
Типы потоковой обработки



Apache Flink

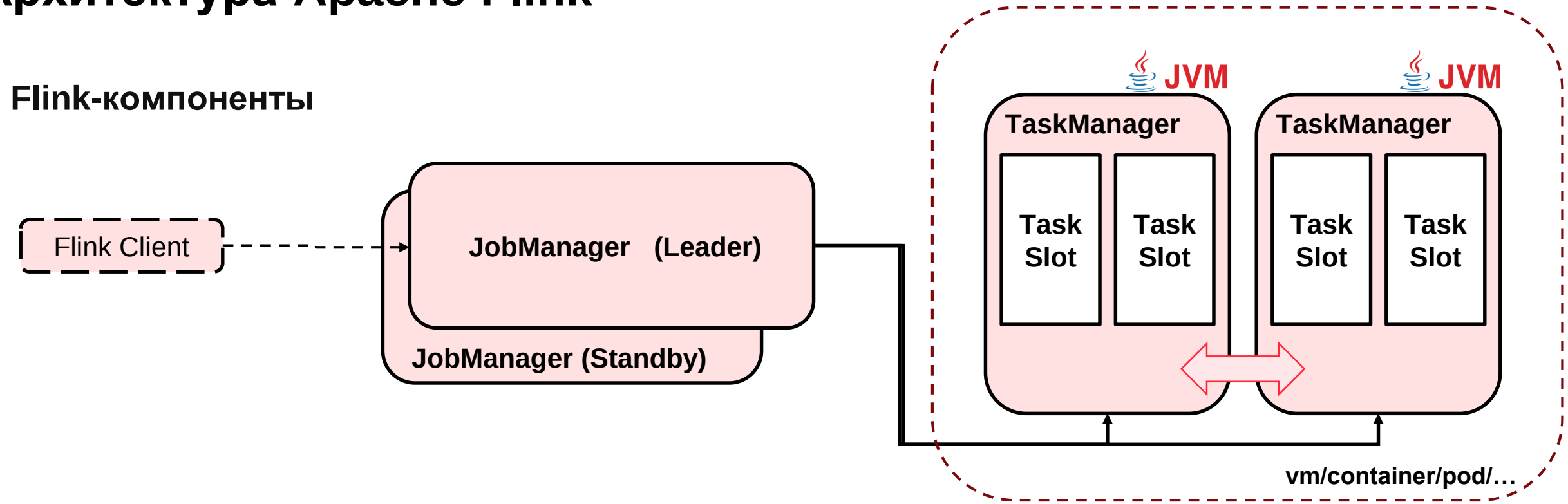
Apache Flink 1.17





Архитектура Apache Flink

Flink-компоненты



Внешние компоненты

HA Service Provider

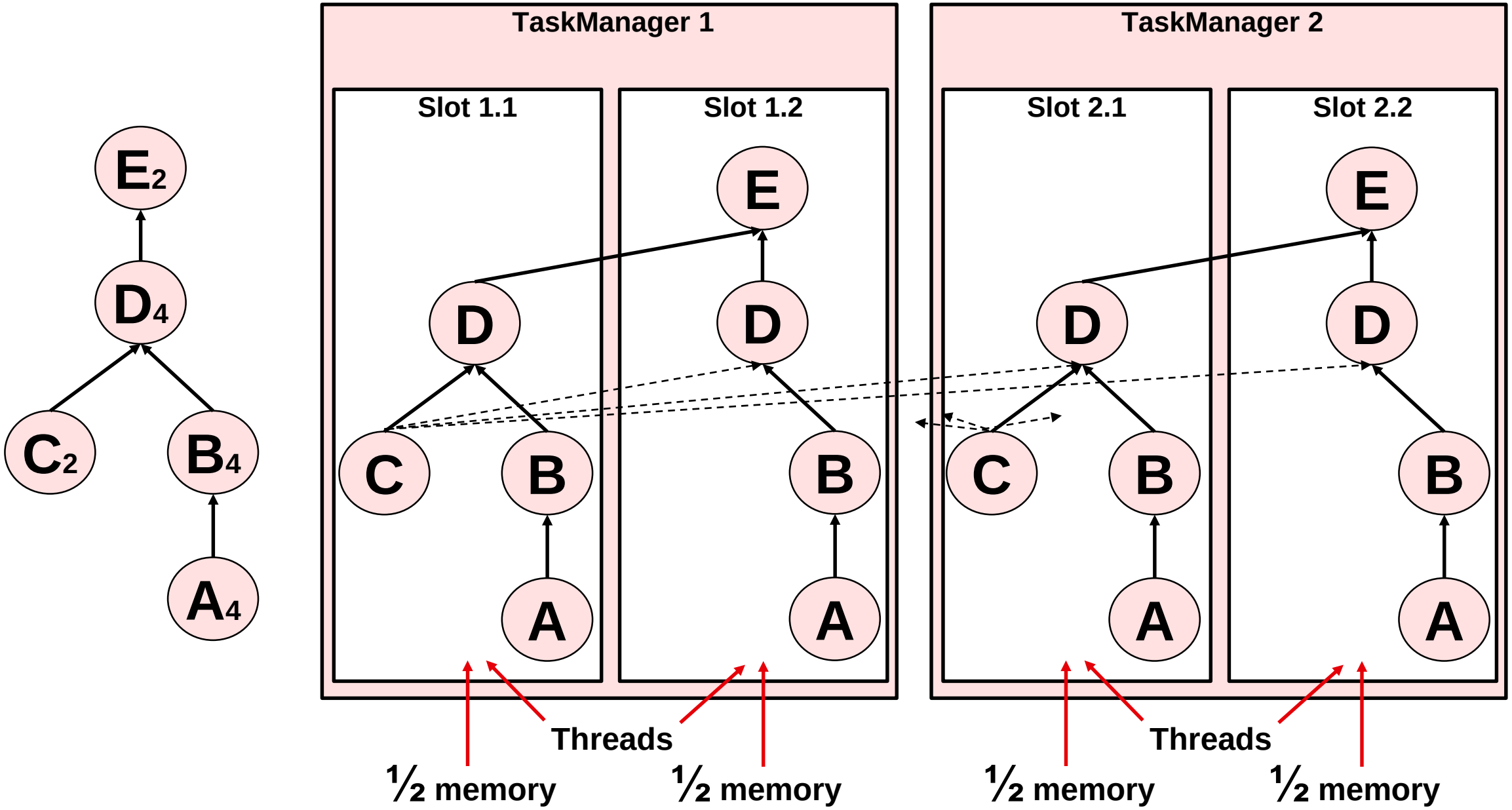
File Storage

Resource Manager

Metric Storage

Application-level sources/sinks

Выполнение Flink задачи



Что такое Flink-задание?

Flink-задание

```
final var env : StreamExecutionEnvironment =
        StreamExecutionEnvironment.getExecutionEnvironment();

env.fromElements( ...data: "qwe", "asd", "zxc") DataStreamSource<String>
        .filter() SingleOutputStreamOperator<String>
        .map() SingleOutputStreamOperator<Object>
        .process()
        .sinkTo(new PrintSink<>());

env.executeAsync();
```

Flink-задание

```
final var env : StreamExecutionEnvironment =
        StreamExecutionEnvironment.getExecutionEnvironment();

env.fromElements( ...data: "qwe", "asd", "zxc") DataStreamSource<String>
        .filter() SingleOutputStreamOperator<String>
        .map() SingleOutputStreamOperator<Object>
        .process()
        .sinkTo(new PrintSink<>());

env.executeAsync();
```

Flink-задание

```
final var env : StreamExecutionEnvironment =
        StreamExecutionEnvironment.getExecutionEnvironment();

env.fromElements( ...data: "qwe", "asd", "zxc") DataStreamSource<String>
        .filter() SingleOutputStreamOperator<String>
        .map() SingleOutputStreamOperator<Object>
        .process()
        .sinkTo(new PrintSink<>());

env.executeAsync();
```

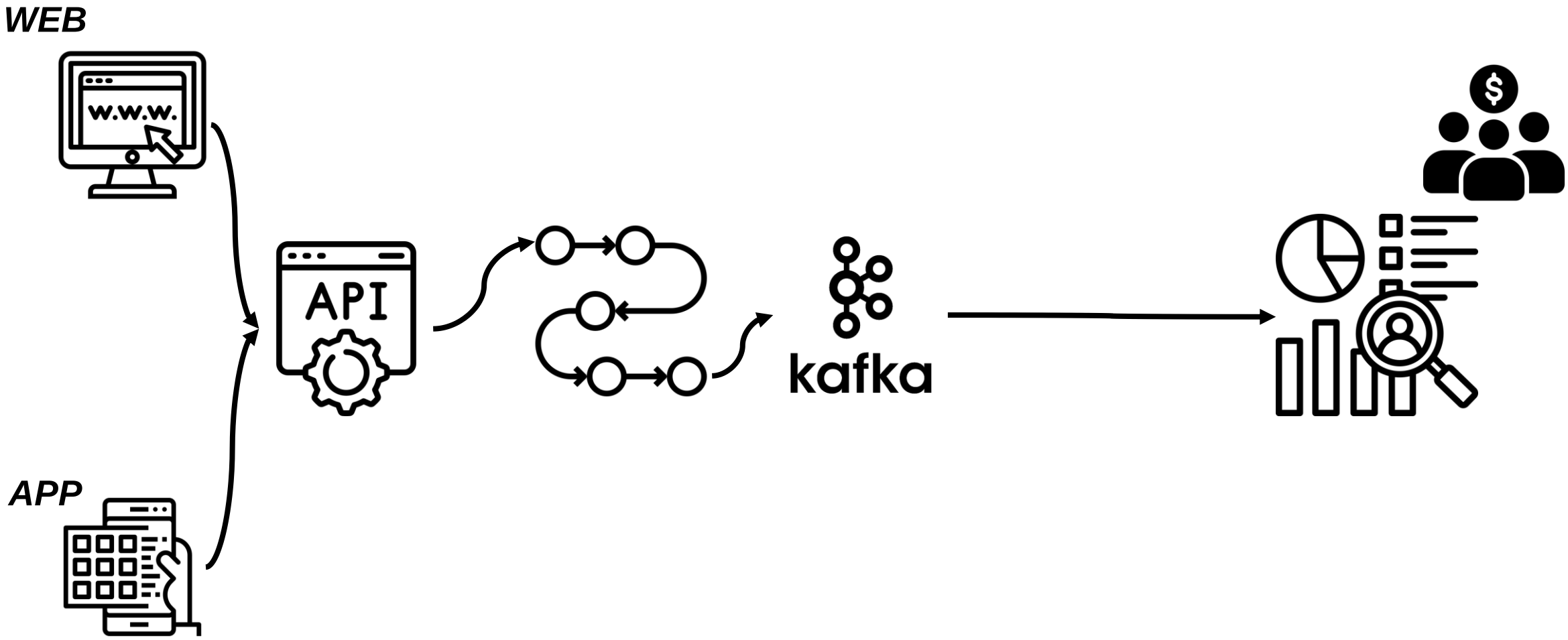
Flink-задание

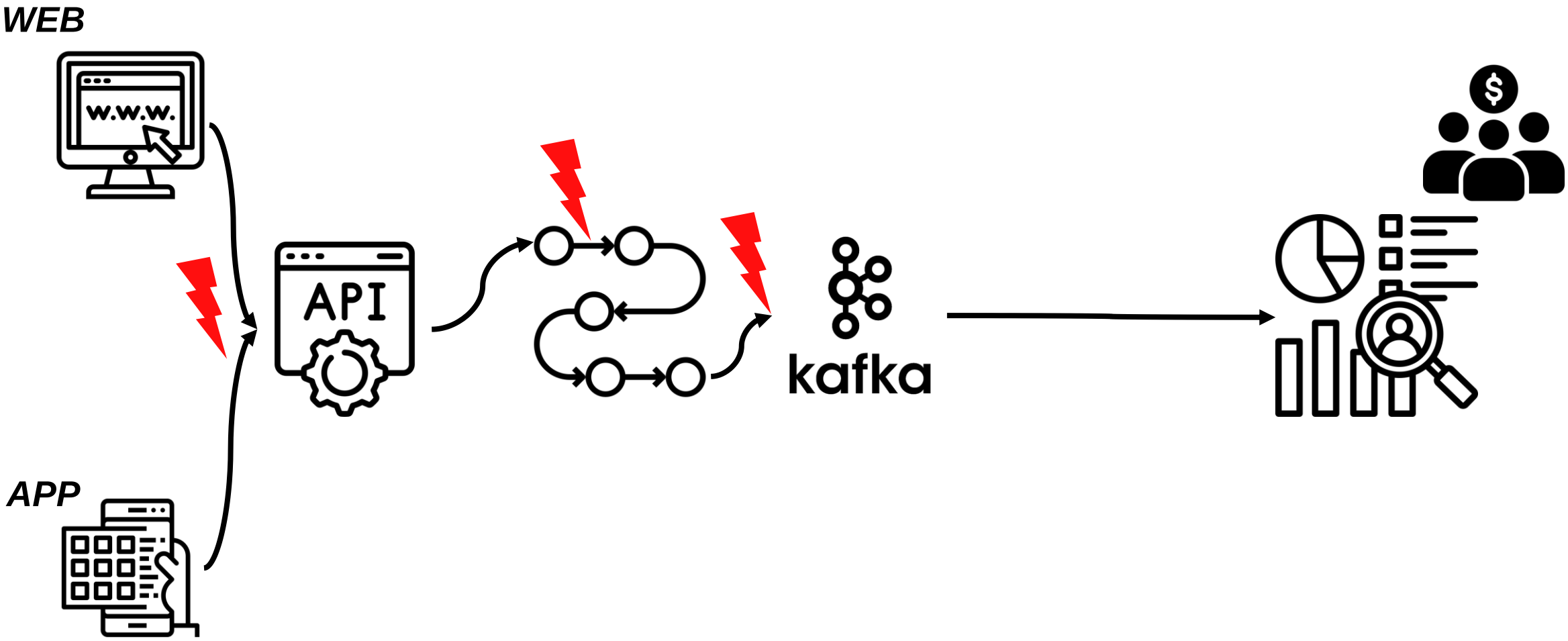
```
final var env : StreamExecutionEnvironment =
        StreamExecutionEnvironment.getExecutionEnvironment();

env.fromElements( ...data: "qwe", "asd", "zxc") DataStreamSource<String>
        .filter() SingleOutputStreamOperator<String>
        .map() SingleOutputStreamOperator<Object>
        .process()
        .sinkTo(new PrintSink<>());

env.executeAsync();
```

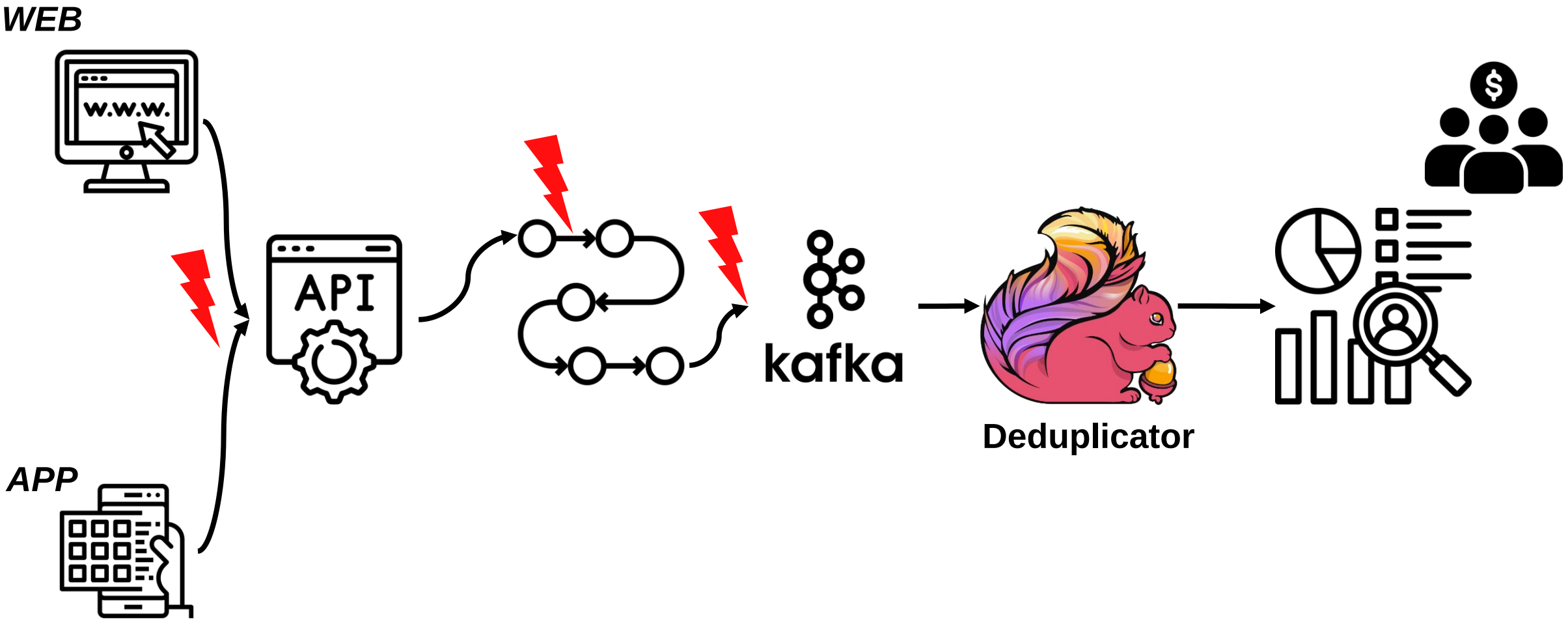
Flink-задание

```
final var env : StreamExecutionEnvironment =
        StreamExecutionEnvironment.getExecutionEnvironment();

env.fromElements( ...data: "qwe", "asd", "zxc") DataStreamSource<String>
        .filter() SingleOutputStreamOperator<String>
        .map() SingleOutputStreamOperator<Object>
        .process()
        .sinkTo(new PrintSink<>());

env.executeAsync();
```

Дедупликация


WEB
w.w.w.
API
kafka
APP


WEB
w.w.w.
API
kafka
APP


WEB
w.w.w.
API
kafka
Deduplicator
APP

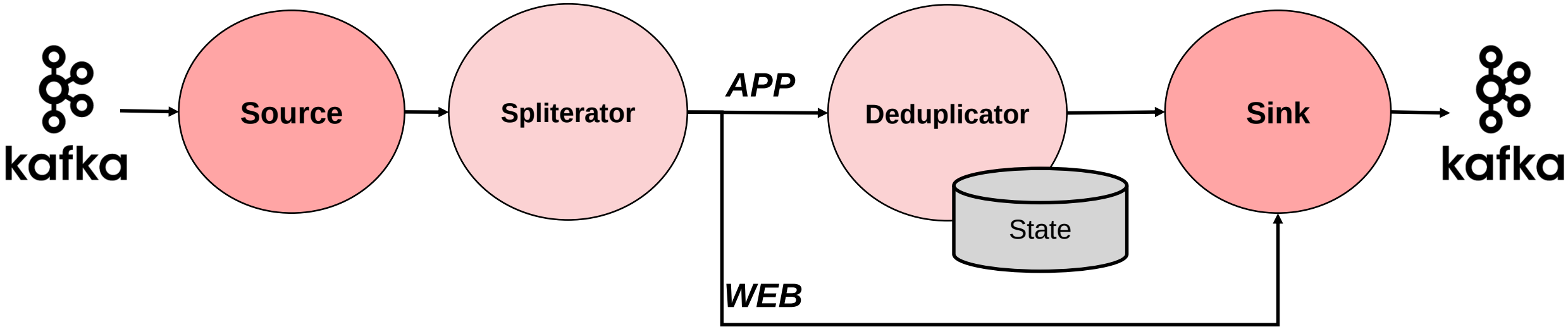
Задача дедупликации данных

Задача дедупликации

Исходные данные:

- Дедупликация APP в окне 30 s
- Нагрузка ~100 rps


```
{
  "id": "9ea7f44f-822b-4569-84ea-6dc6ffcaf19f",
  "uniqueValue": "656b4183-282c-1689019596",
  "type": "WEB",
  "timestamp": 1689019717
}
```

Задача дедупликации

```
final var env = StreamExecutionEnvironment
        .getExecutionEnvironment();

env.fromSource(kafkaSource)
        // .split()
        .flatMap() // deduplicator
        .sinkTo(kafkaSink);

env.executeAsync();
```

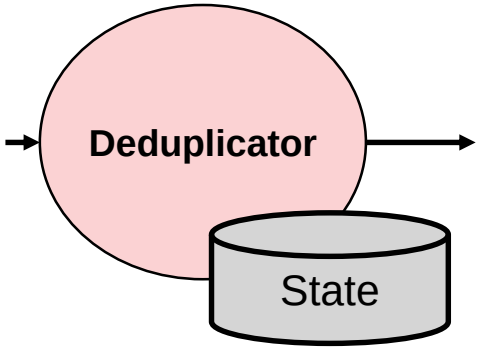
Задача дедупликации

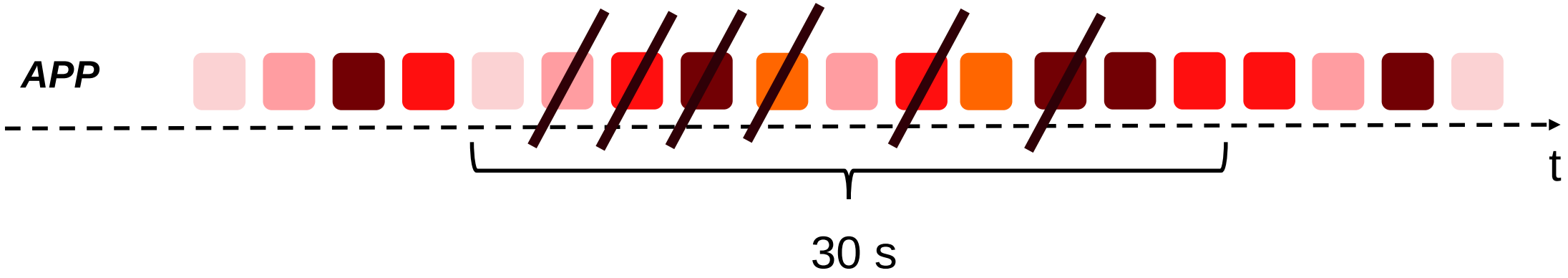
```
final var env = StreamExecutionEnvironment
        .getExecutionEnvironment();

env.fromSource(kafkaSource)
        // .split()
        .flatMap() // deduplicator
        .sinkTo(kafkaSink);

env.executeAsync();
```

Задача дедупликации

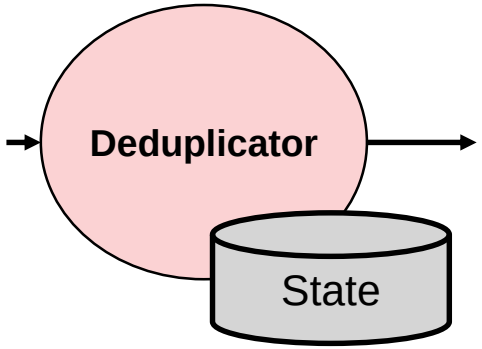
```
final var env = StreamExecutionEnvironment
        .getExecutionEnvironment();

env.fromSource(kafkaSource)
        // .split()
        .flatMap() // deduplicator
        .sinkTo(kafkaSink);

env.executeAsync();
```

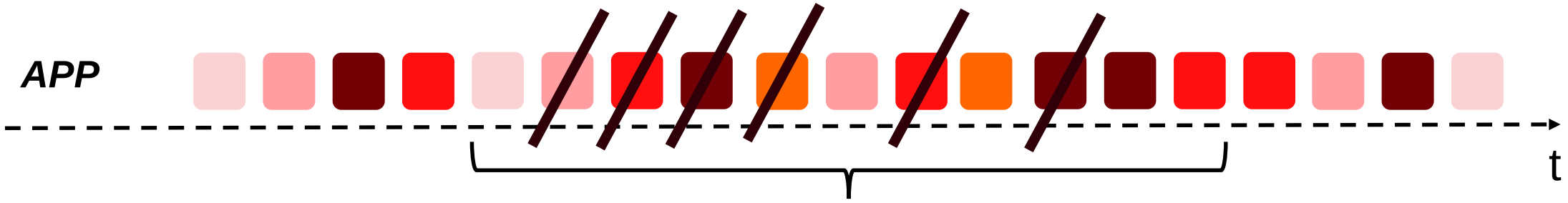
Дедупликатор





Дедупликатор



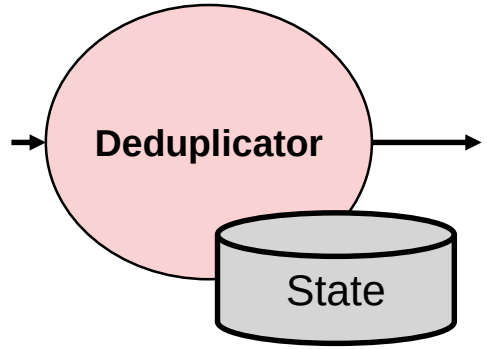


30 s

stream.window…()

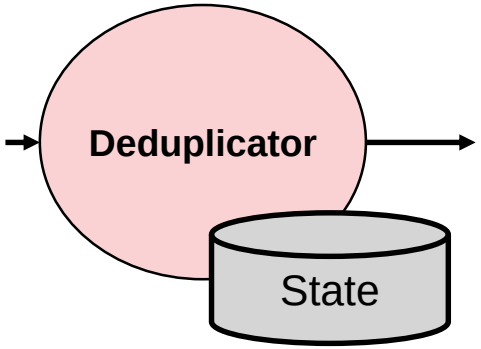
Дедупликатор

```
public class Deduplicator<T> extends RichFlatMapFunction<T, T> {
    private static final long serialVersionUID = 1L;

    private final Time ttl;
    private transient ValueState<Boolean> keyHasBeenSeen;

    public Deduplicator(Time ttl) { this.ttl = notNull(ttl, message: "Deduplicator TTL is null"); }

    @Override
    public void open(Configuration parameters) {...}

    @Override
    public void flatMap(T event, Collector<T> out) throws Exception {
        if (keyHasBeenSeen.value() == null) {
            out.collect(event);
            keyHasBeenSeen.update( value: true);
        }
    }
}
```



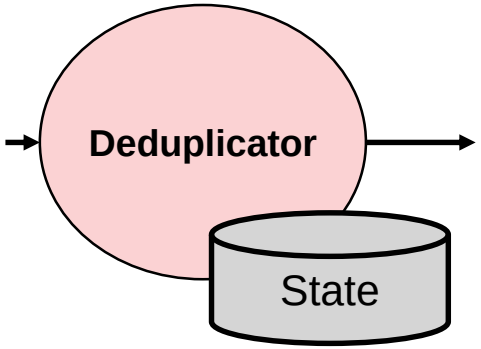
Дедупликатор

```java
public class Deduplicator<T> extends RichFlatMapFunction<T, T> {
    private static final long serialVersionUID = 1L;

    private final Time ttl;
    private transient ValueState<Boolean> keyHasBeenSeen;

    public Deduplicator(Time ttl) { this.ttl = notNull(ttl, message: "Deduplicator TTL is null"); }

    @Override
    public void open(Configuration parameters) {...}

    @Override
    public void flatMap(T event, Collector<T> out) throws Exception {
        if (keyHasBeenSeen.value() == null) {
            out.collect(event);
            keyHasBeenSeen.update( value: true);
        }
    }
}
```



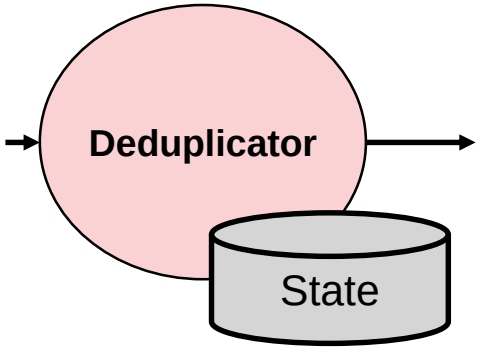
Дедупликатор

```
public class Deduplicator<T> extends RichFlatMapFunction<T, T> {
    private static final long serialVersionUID = 1L;

    private final Time ttl;
    private transient ValueState<Boolean> keyHasBeenSeen;

    public Deduplicator(Time ttl) { this.ttl = notNull(ttl, message: "Deduplicator TTL is null"); }

    @Override
    public void open(Configuration parameters) {...}

    @Override
    public void flatMap(T event, Collector<T> out) throws Exception {
        if (keyHasBeenSeen.value() == null) {
            out.collect(event);
            keyHasBeenSeen.update( value: true);
        }
    }
}
```

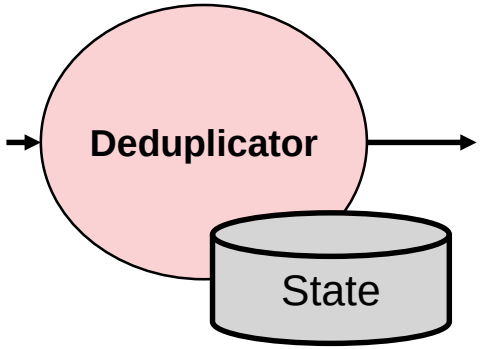


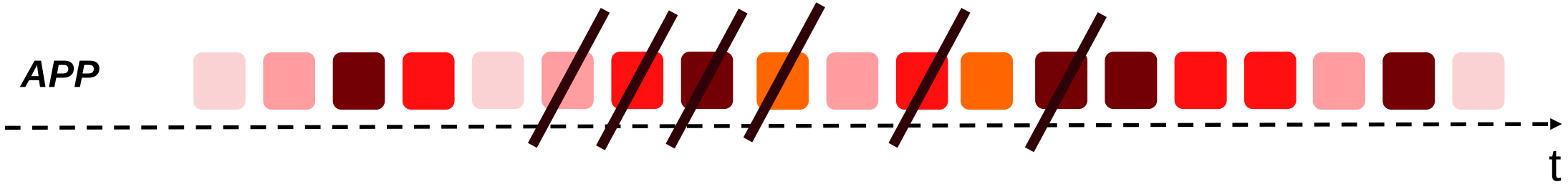
Дедупликатор

```
public class Deduplicator<T> extends RichFlatMapFunction<T, T> {
    private static final long serialVersionUID = 1L;

    private final Time ttl;
    private transient ValueState<Boolean> keyHasBeenSeen;

    public Deduplicator(Time ttl) { this.ttl = notNull(ttl, message: "Deduplicator TTL is null"); }

    @Override
    public void open(Configuration parameters) {...}

    @Override
    public void flatMap(T event, Collector<T> out) throws Exception {
        if (keyHasBeenSeen.value() == null) {
            out.collect(event);
            keyHasBeenSeen.update( value: true);
        }
    }
}
```

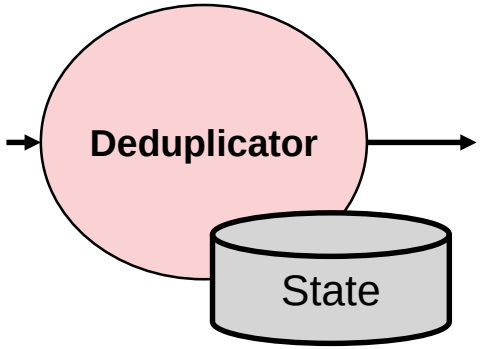


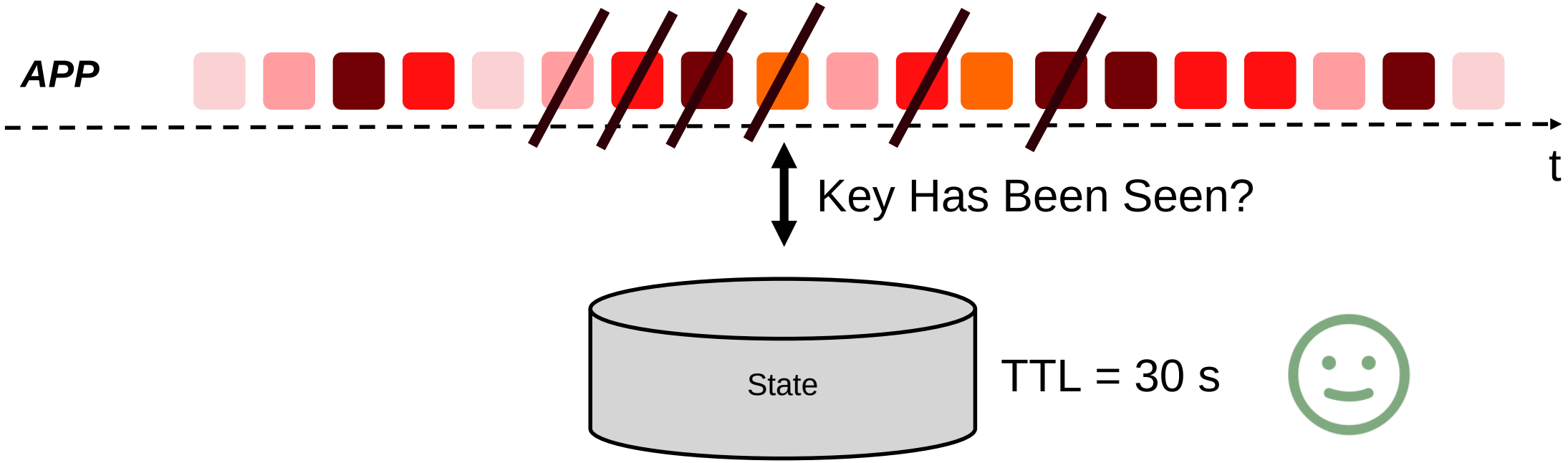
Дедупликатор



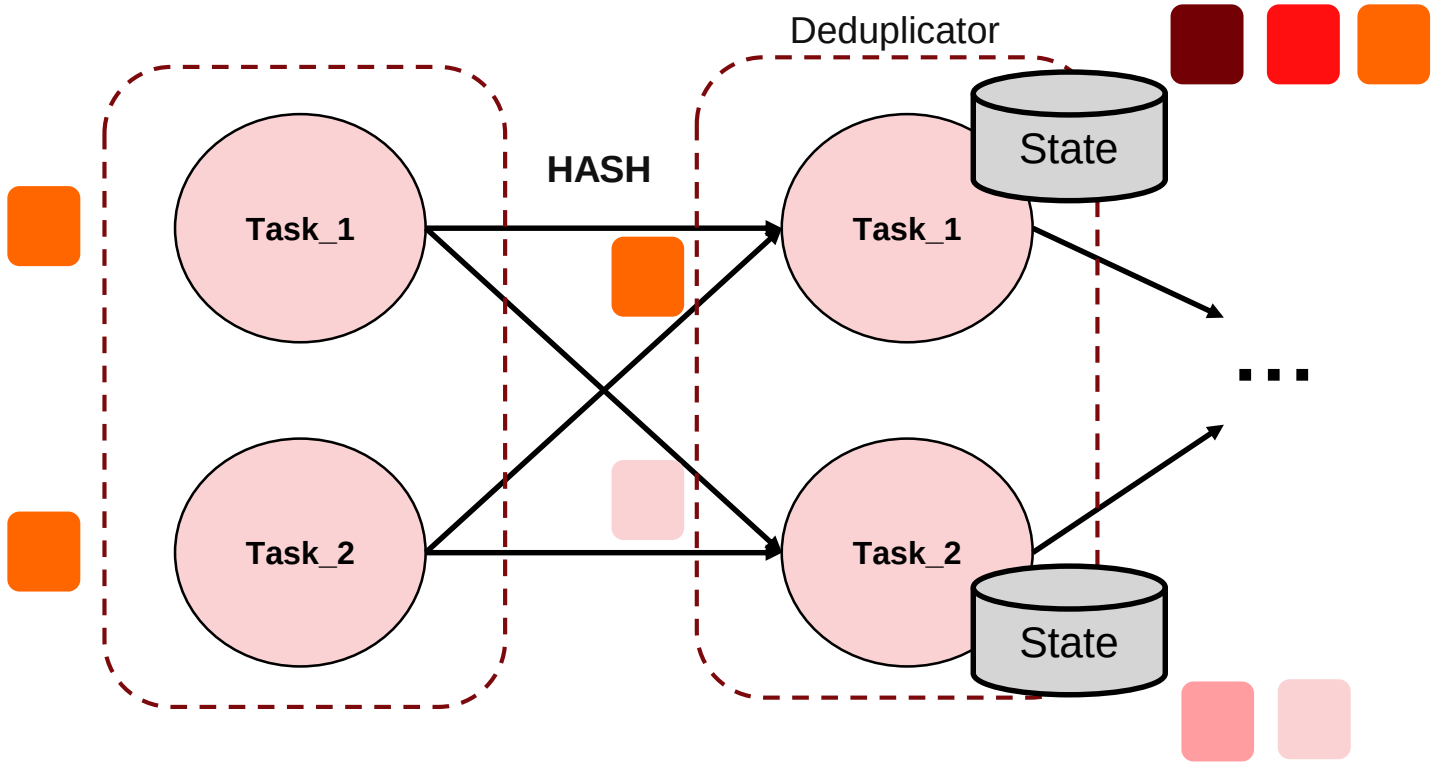


Дедупликатор



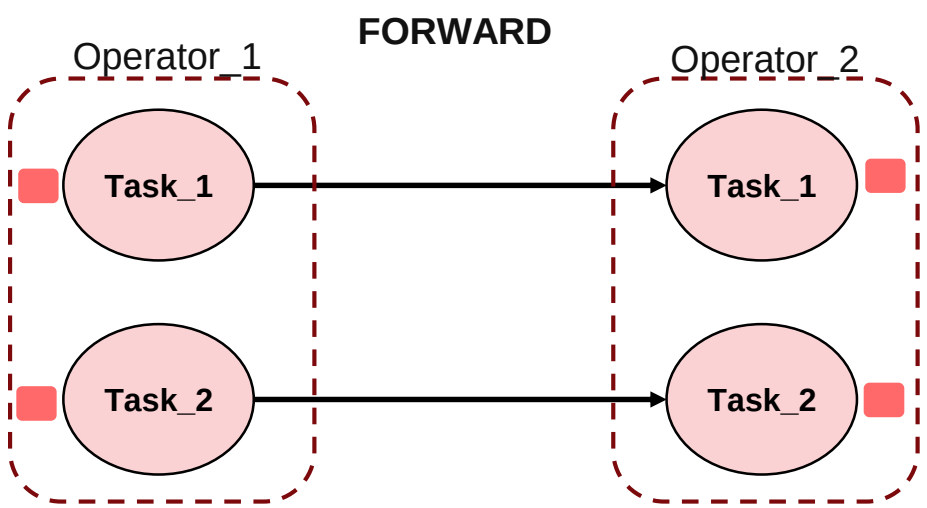


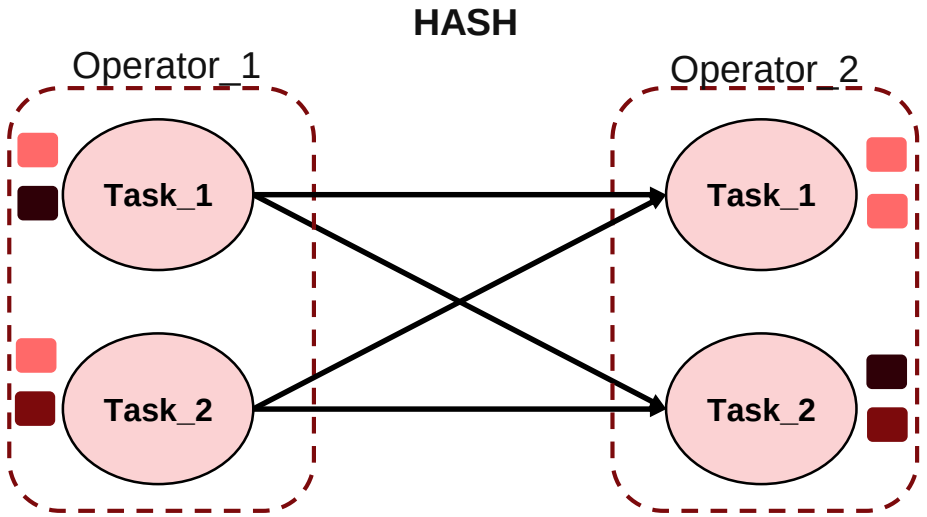
Дедупликатор и keyBy

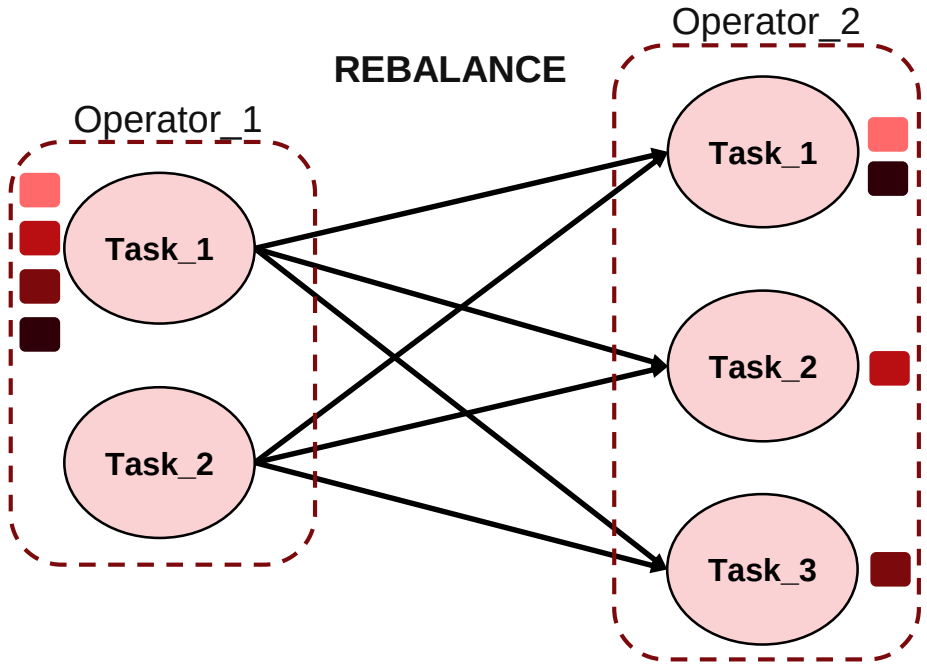


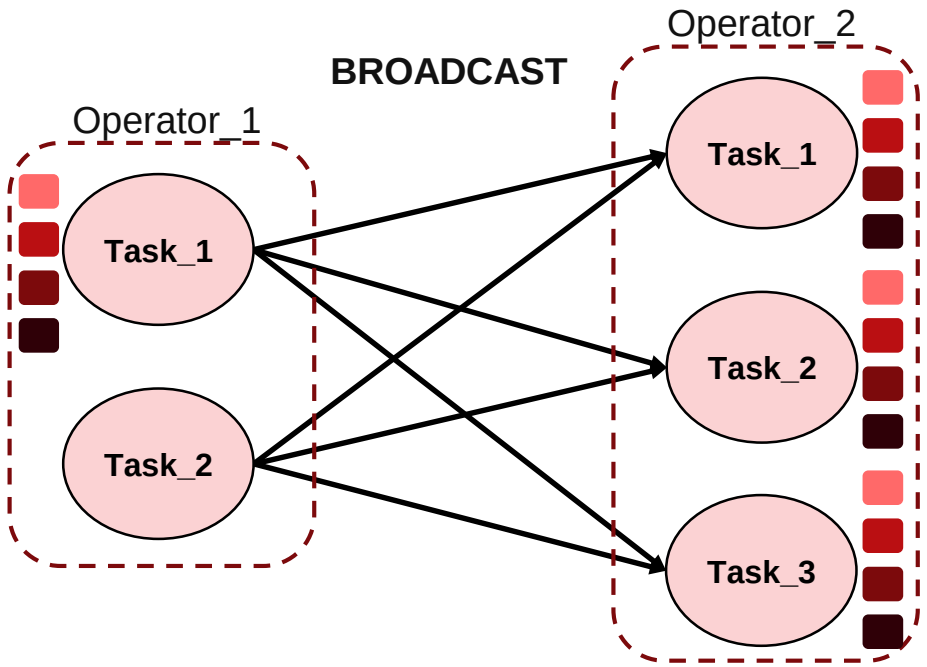

```
appStream
    .keyBy(value -> value.getUniqueValue())
    .flatMap(new Deduplicator<>(Time.seconds(30)))
```

Распределение событий

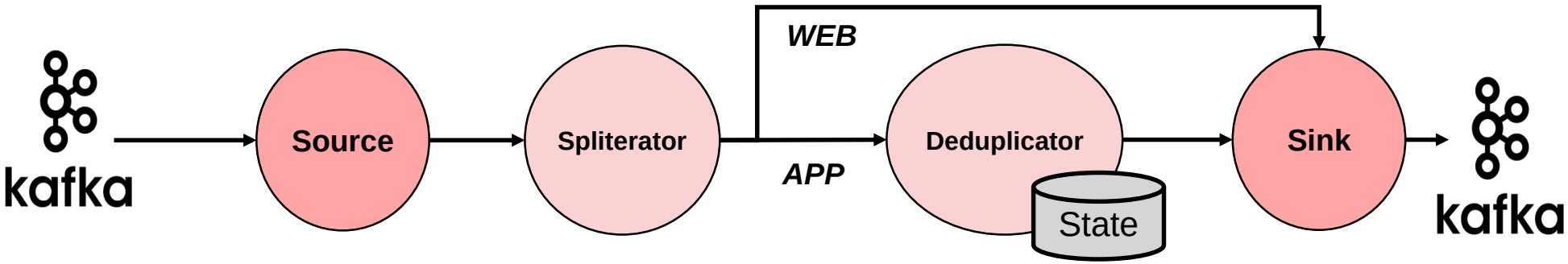




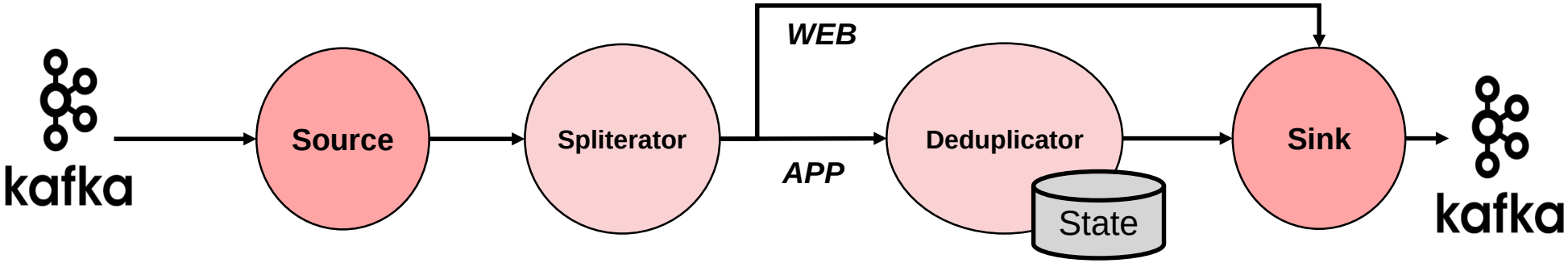




Текущая реализация задания

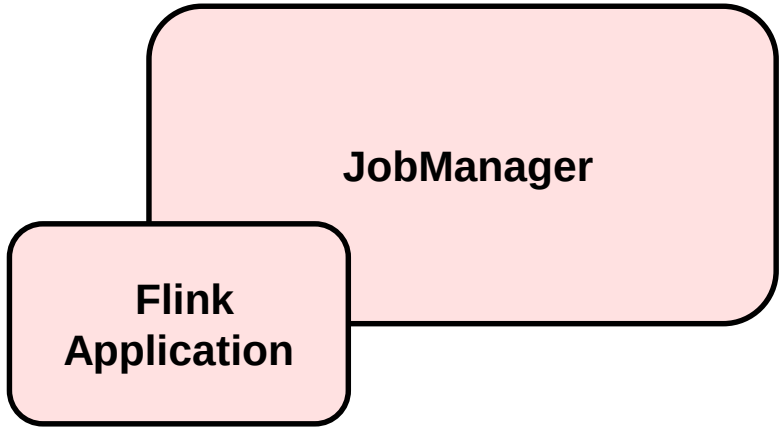


Текущая реализация задания



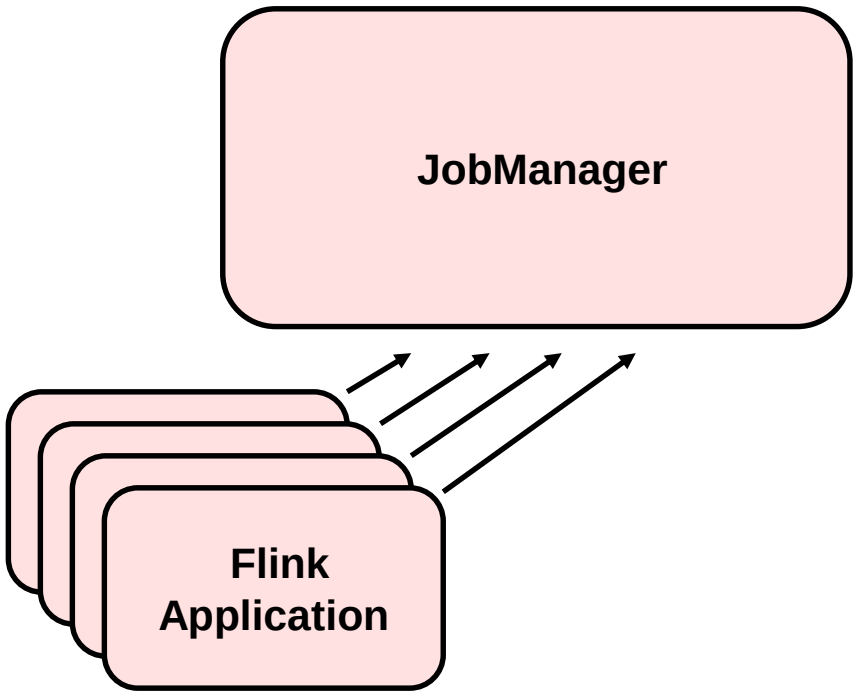
Режимы развертывания

Application Mode



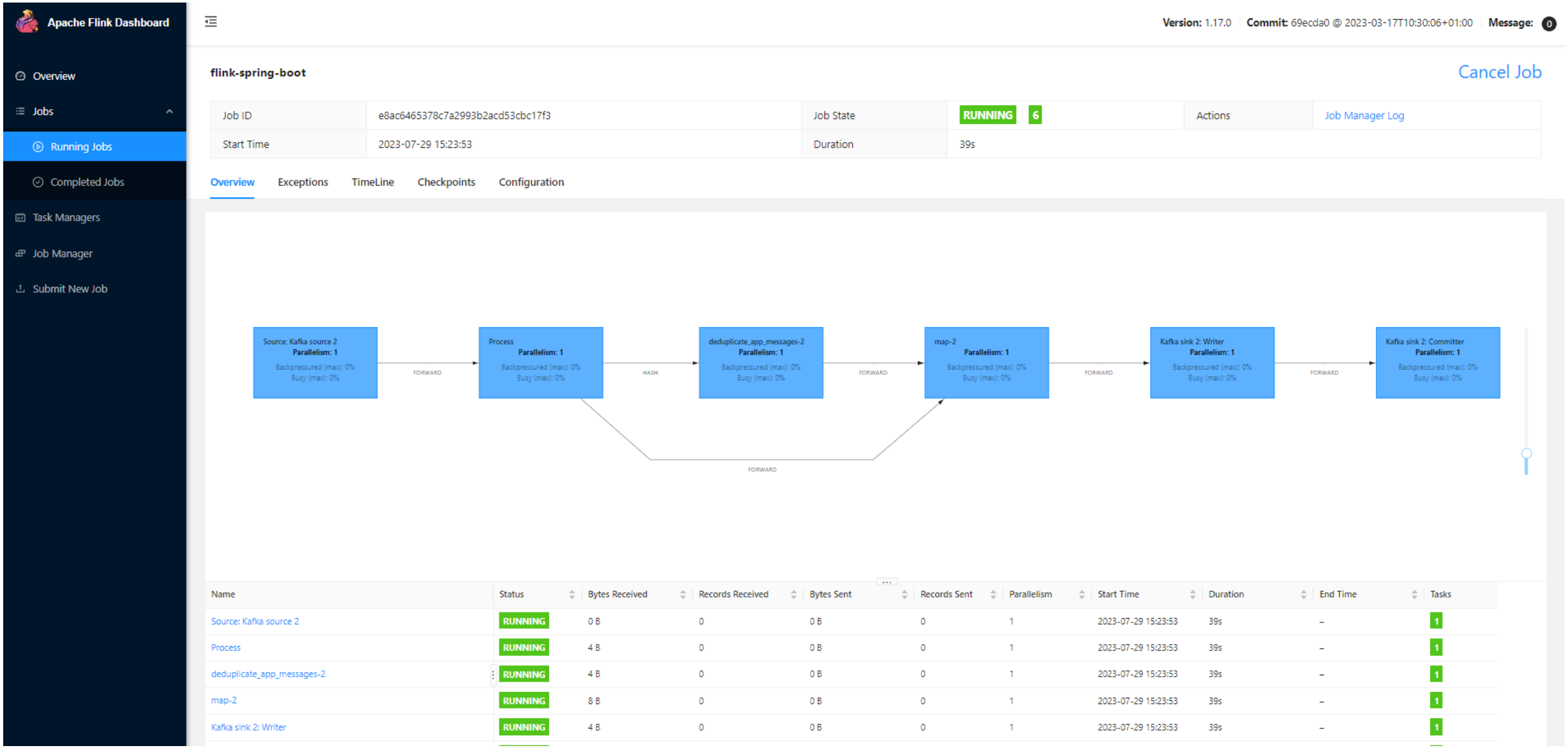
- Отдельный кластер под задание
- Задание поставляется с Flink-дистрибутивом

Session Mode



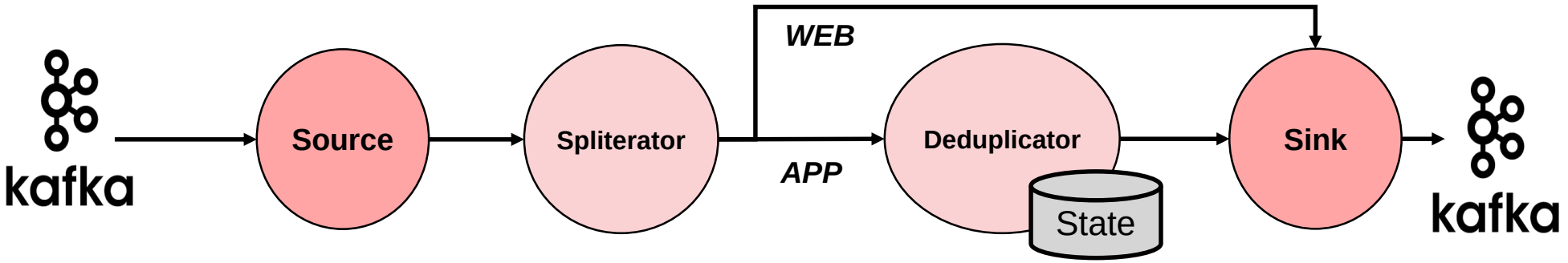
- Общий кластер под разные задания

Задача в UI


Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit New Job
Version: 1.17.0 | Commit: 69ecda0 @ 2023-03-17T10:30:06+01:00 | Message: 0
flink-spring-boot
Cancel Job
Job ID
e8ac6465378c7a2993b2acd53cbc17f3
Job State
RUNNING 6
Actions
Job Manager Log
Start Time
2023-07-29 15:23:53
Duration
39s
Overview
Exceptions
TimeLine
Checkpoints
Configuration
Source: Kafka source 2
Parallelism: 1
Backpressured (max): 0%
Busy (max): 0%
FORWARD
Process
Parallelism: 1
Backpressured (max): 0%
Busy (max): 0%
HASH
deduplicate_app_messages-2
Parallelism: 1
Backpressured (max): 0%
Busy (max): 0%
FORWARD
map-2
Parallelism: 1
Backpressured (max): 0%
Busy (max): 0%
FORWARD
Kafka sink 2: Writer
Parallelism: 1
Backpressured (max): 0%
Busy (max): 0%
FORWARD
Kafka sink 2: Committer
Parallelism: 1
Backpressured (max): 0%
Busy (max): 0%
FORWARD
Name
Status
Bytes Received
Records Received
Bytes Sent
Records Sent
Parallelism
Start Time
Duration
End Time
Tasks
Source: Kafka source 2
RUNNING
0 B
0
0 B
0
1
2023-07-29 15:23:53
39s
–
1
Process
RUNNING
4 B
0
0 B
0
1
2023-07-29 15:23:53
39s
–
1
deduplicate_app_messages-2
RUNNING
4 B
0
0 B
0
1
2023-07-29 15:23:53
39s
–
1
map-2
RUNNING
8 B
0
0 B
0
1
2023-07-29 15:23:53
39s
–
1
Kafka sink 2: Writer
RUNNING
4 B
0
0 B
0
1
2023-07-29 15:23:53
39s
–
1

Не все так просто

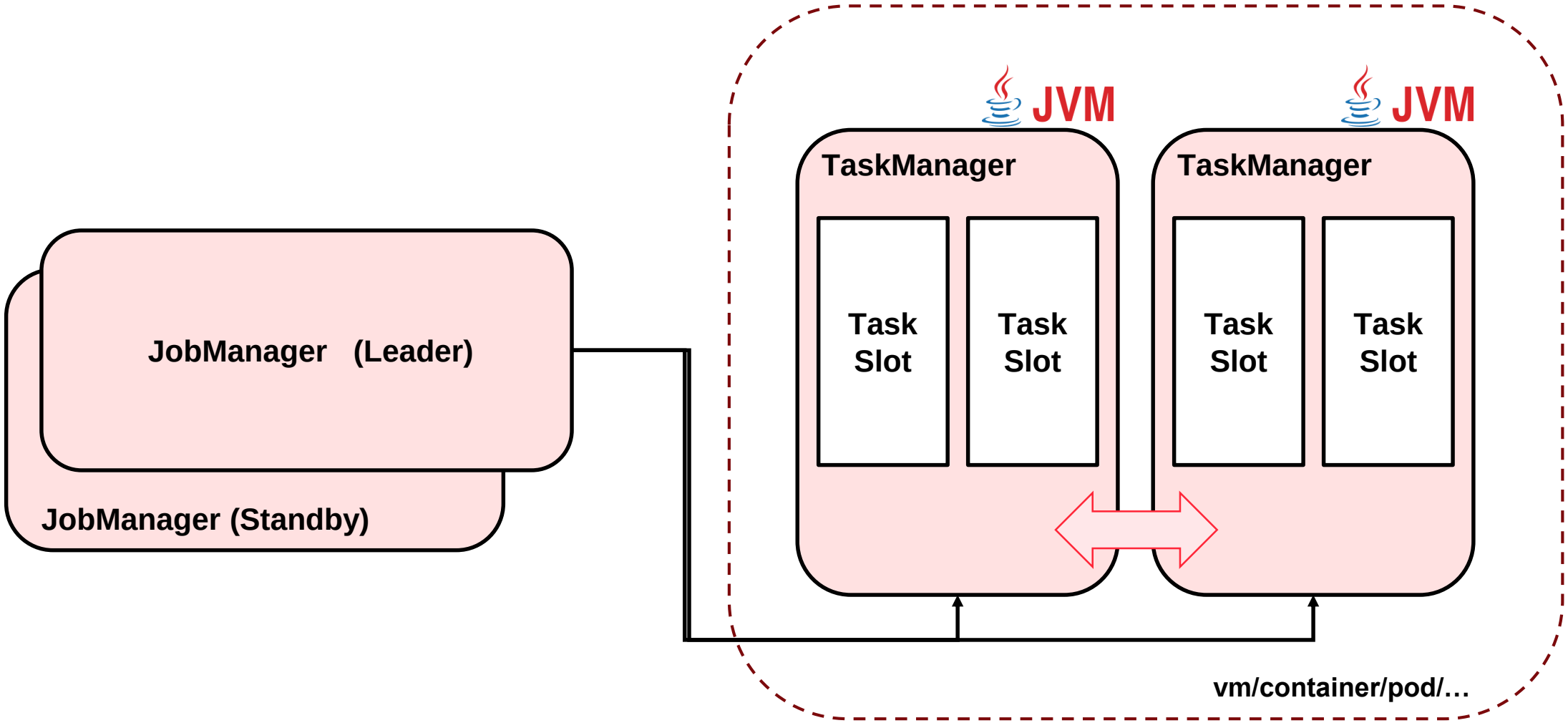
Текущая реализация задания



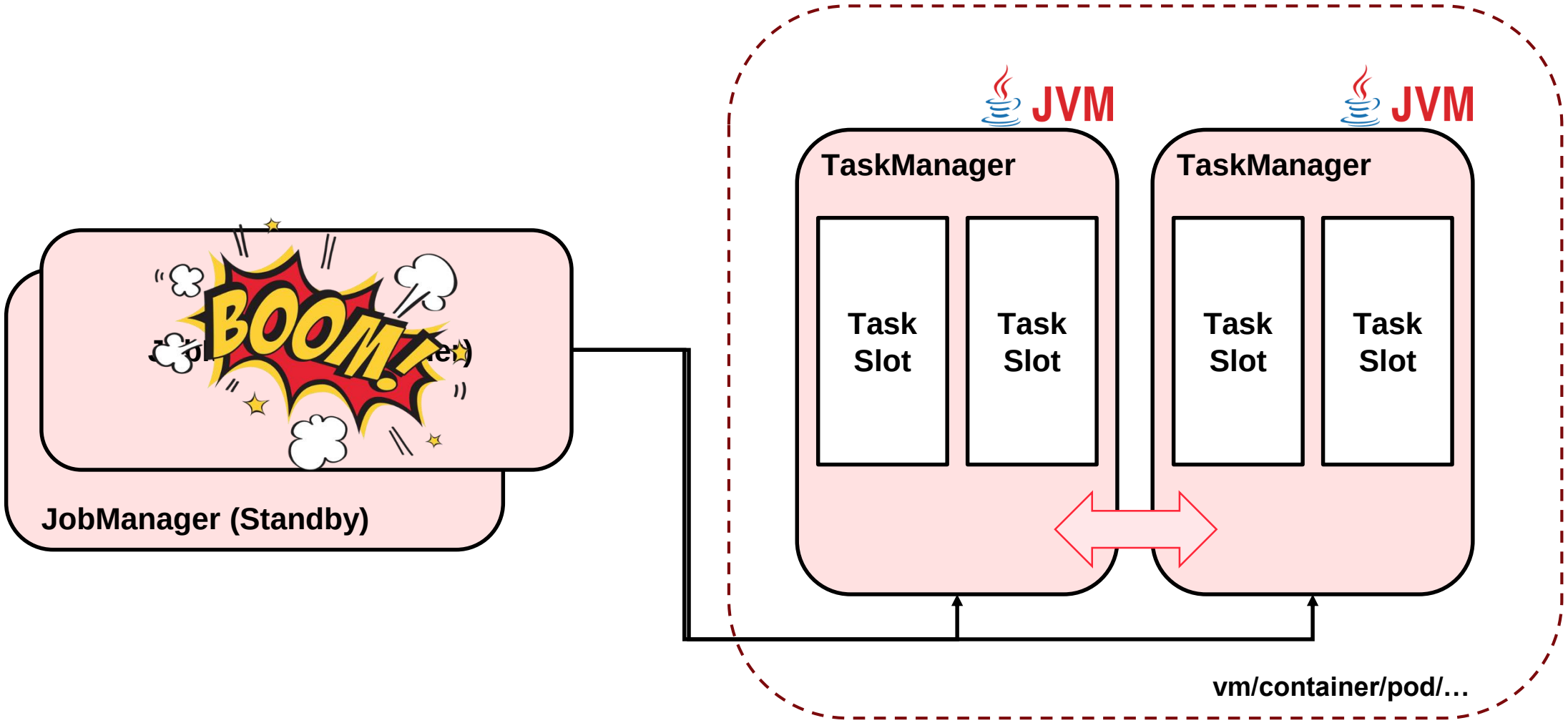
Что может пойти не так?

- **Сбой кластера Flink**
- Настройка exactly once
- Эволюция приложения
- Непрерывная работа Flink-задания
- Увеличение нагрузки. RocksDB
- Работа с большим состоянием
- Ускорение работы Flink-задания

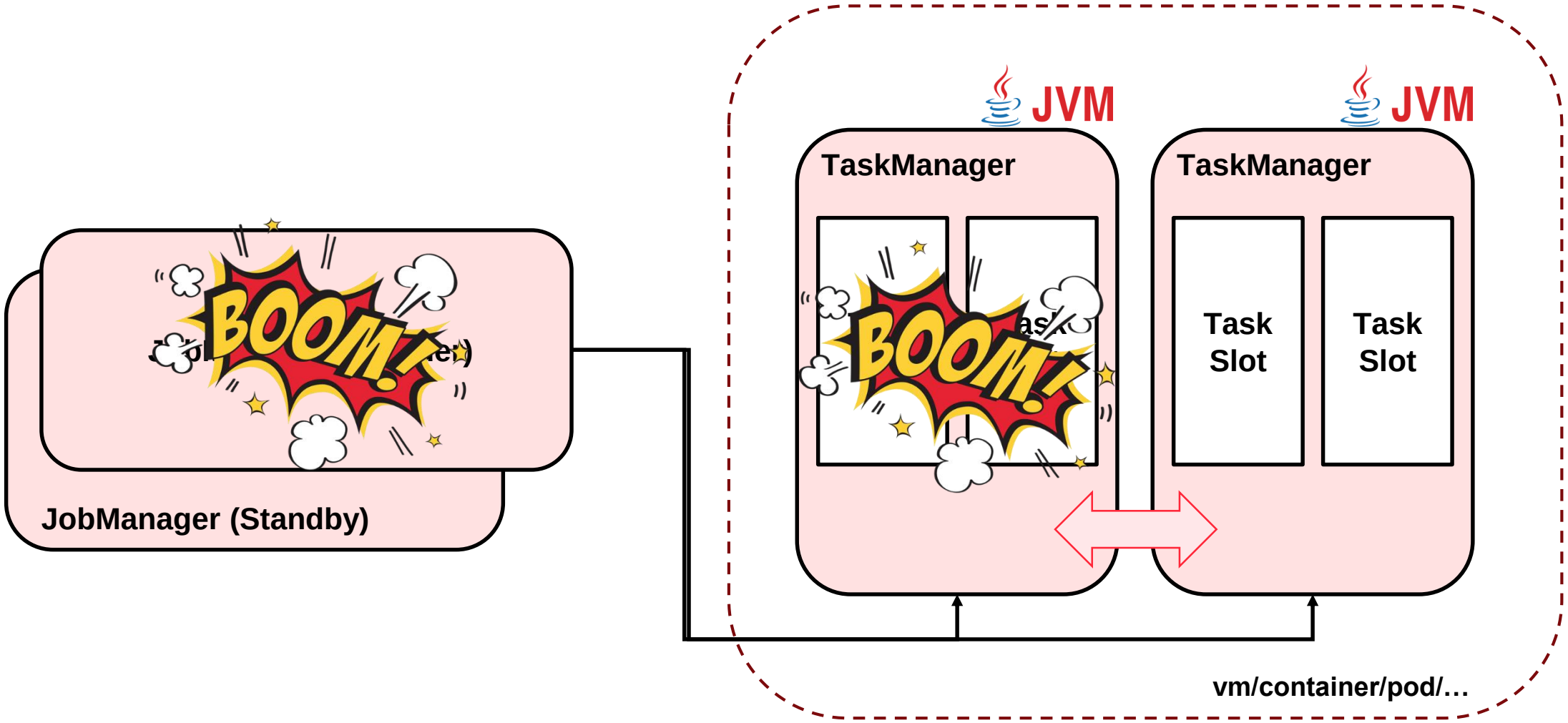
Сбой кластера Flink



Сбой кластера Flink



Сбой кластера Flink



HA JobManager

Time

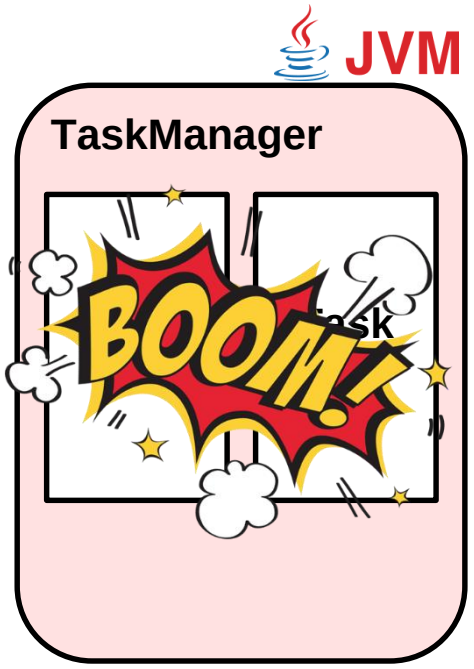
JobManager Leader	**JobManager** Standby	**JobManager** Standby
	**JobManager** Standby	**JobManager** Standby
Recovering	**JobManager** Leader	**JobManager** Standby
JobManager Standby	**JobManager** Leader	**JobManager** Standby

Службы High Availability:

kubernetes

APACHE ZooKeeper™

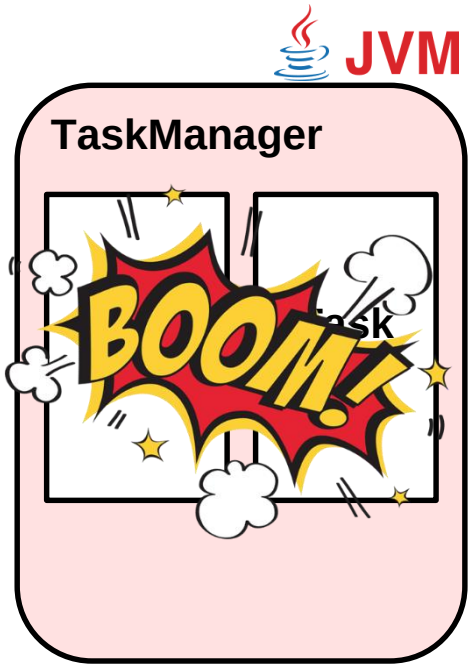
Стратегии перезапуска упавшей задачи




```java
final var env = StreamExecutionEnvironment
        .getExecutionEnvironment();
env.setRestartStrategy(
        RestartStrategies.fixedDelayRestart(
                restartAttempts: 3,
                Time.minutes(1)
        )
);
```

- No Restart Strategy
- Fixed Delay Restart Strategy
- Exponential Delay Restart Strategy
- Failure Rate Restart Strategy
- Fallback Restart Strategy

Стратегии перезапуска упавшей задачи



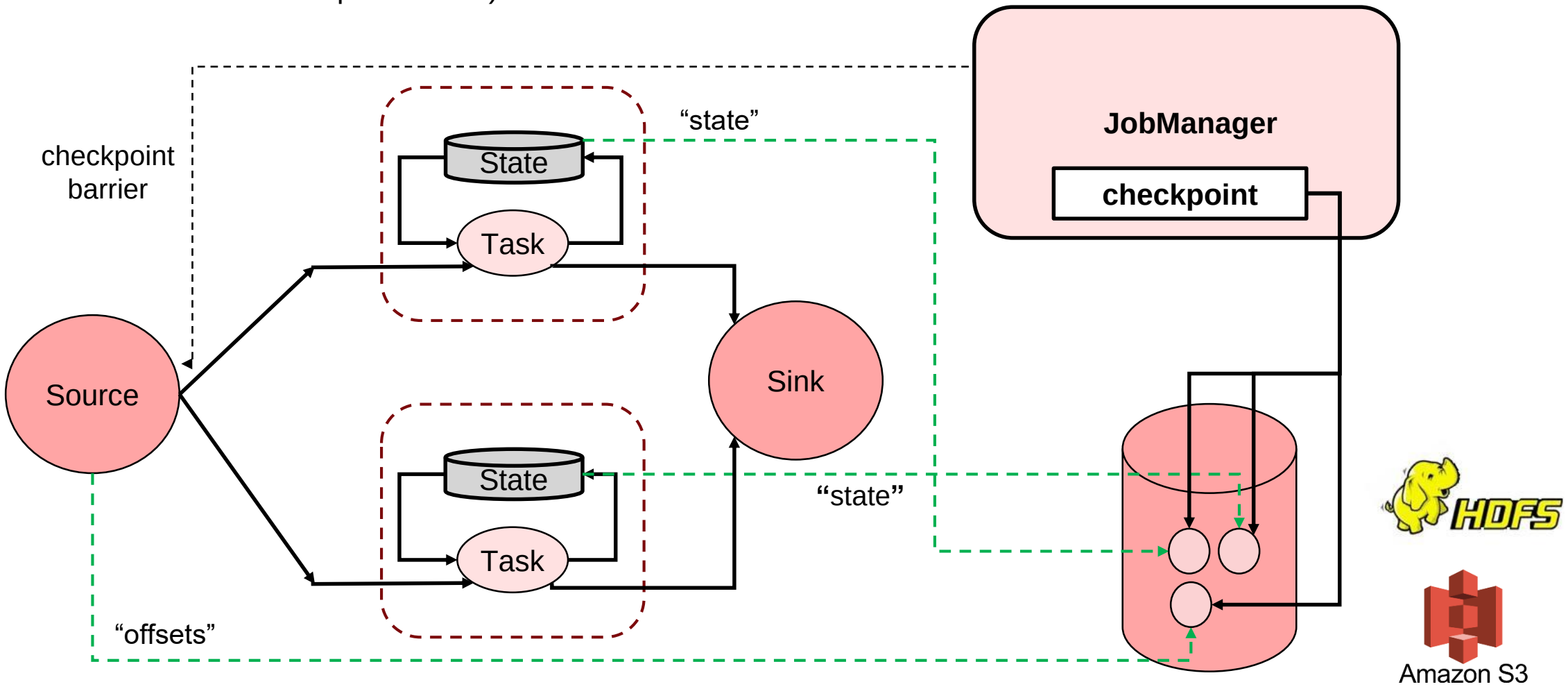

```
final var env = StreamExecutionEnvironment
        .getExecutionEnvironment();
env.setRestartStrategy(
        RestartStrategies.fixedDelayRestart(
                restartAttempts: 3,
                Time.minutes(1)
        )
);
```

- No Restart Strategy
- Fixed Delay Restart Strategy
- Exponential Delay Restart Strategy
- Failure Rate Restart Strategy
- Fallback Restart Strategy

Необходимо перезапустить упавшую задачу, а также все другие задачи, выполняемые в упавшем TaskManager!

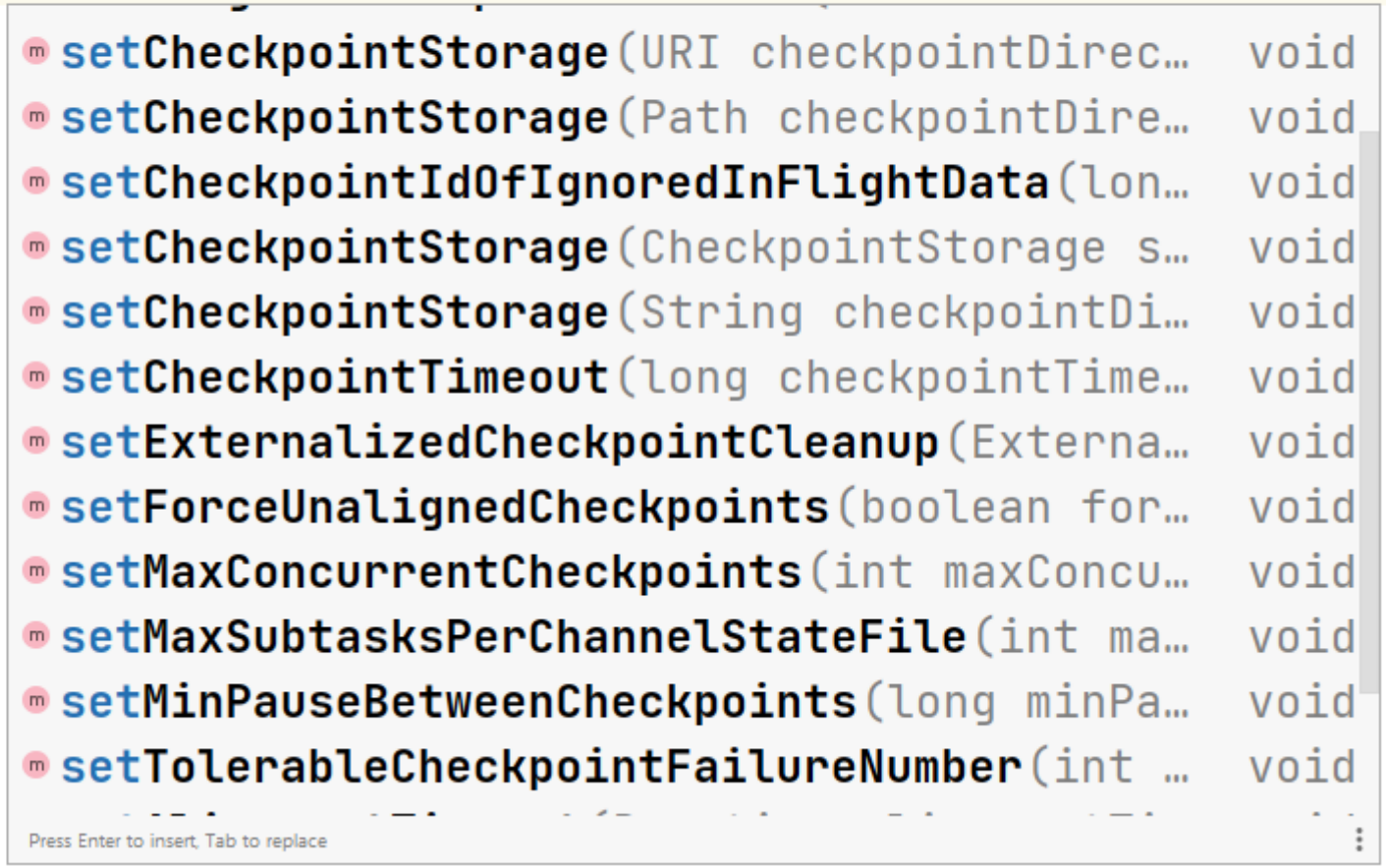
Checkpoint

Checkpoint – согласованная копия состояния **каждой задачи** потокового приложения в момент, когда **все задачи** отработали **один и тот же вход**. Используется для перезапуска приложения в случае **сбоя** (checkpoints удаляются при сознательной остановке приложения).



Checkpoint

```
final var env = StreamExecutionEnvironment.getExecutionEnvironment();
env.enableCheckpointing( interval: 10_000);
env.getCheckpointConfig().set
```



Savepoint

Savepoint – это согласованное изображение состояния выполнения потокового задания, созданное с помощью механизма контрольных точек Flink, но содержит дополнительные метаданные.

НЕ создается и **НЕ** удаляется автоматически. Используется для перезапуска приложения.

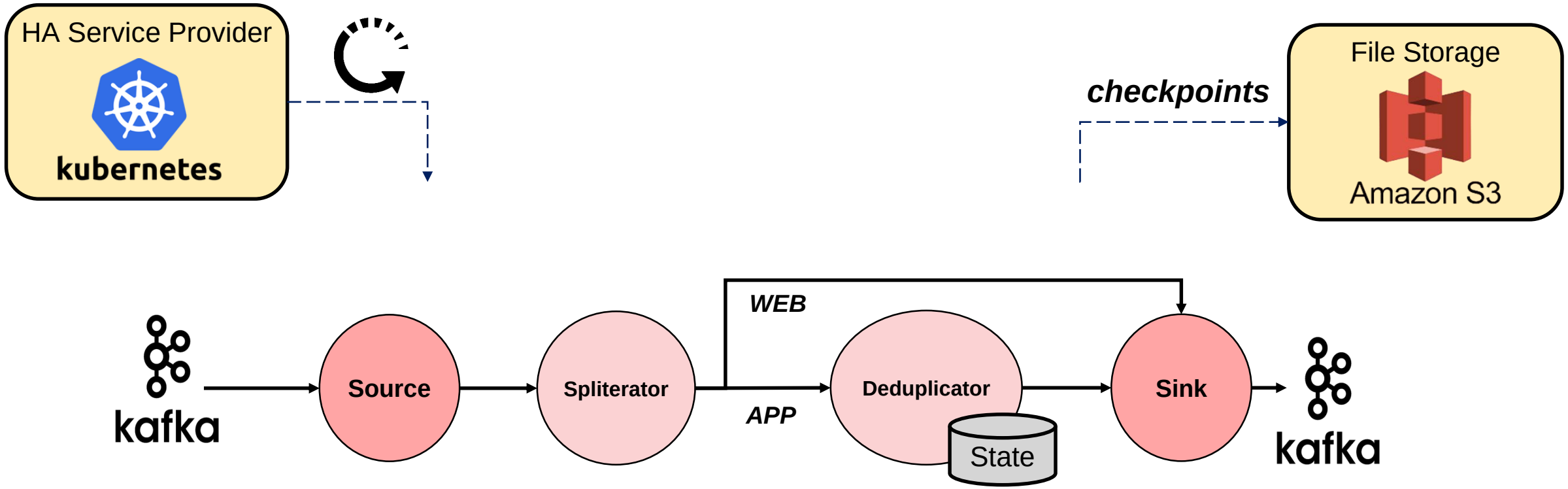
Использование:

- Перезапуск приложение
- Запуск приложения на другом Flink-кластере
- Приостановка/перезапуск в целях временного освобождения ресурсов кластера

Что может пойти не так?

- ☑ **Сбой кластера Flink**
- ❑ Настройка exactly once
- ❑ Эволюция приложения
- ❑ Непрерывная работа Flink-задания
- ❑ Увеличение нагрузки. RocksDB
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

Текущая реализация задания



Что может пойти не так?

- ✅ Сбой кластера Flink
- ❑ **Настройка exactly once**
- ❑ Эволюция приложения
- ❑ Непрерывная работа Flink-задания
- ❑ Увеличение нагрузки. RocksDB
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

Семантики доставки

```
return KafkaSink.<MyEvent>builder()
        .setBootstrapServers(kafkaProperties.getBootstrapServers())
        .setKafkaProducerConfig(kafkaProducerProperties)
        .setRecordSerializer(KafkaRecordSerializationSchema.<MyEvent>builder()
                .setTopic(kafkaProperties.getTopics().getMyTopic())
                .setValueSerializationSchema(myEventSerializationSchema)
                .build())
        .setDeliveryGuarantee(DeliveryGuarantee.)
        .build();
```

```
NONE                                DeliveryGuarantee
valueOf(String name)                DeliveryGuarantee
AT_LEAST_ONCE                       DeliveryGuarantee
EXACTLY_ONCE                        DeliveryGuarantee
Press Enter to insert, Tab to replace
```

Exactly once не бесплатный

Checkpointing Modes

Overview | History | Summary | Configuration | Refresh

Option	Value
Checkpointing Mode	At Least Once
Checkpoint Storage	JobManagerCheckpointStorage
State Backend	HashMapStateBackend
Interval	Periodic checkpoints disabled
Timeout	10m 0s
Minimum Pause Between Checkpoints	0ms
Maximum Concurrent Checkpoints	1

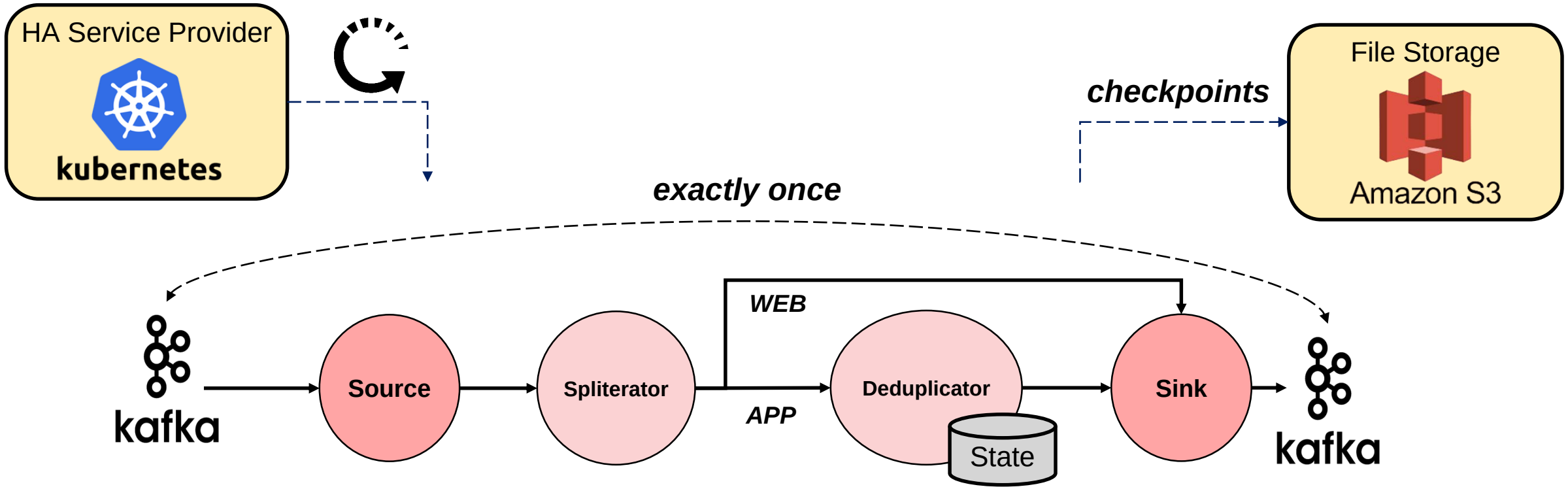
```
final var env = StreamExecutionEnvironment.getExecutionEnvironment();
env.enableCheckpointing( interval: 10_000, CheckpointingMode.)
```

EXACTLY_ONCE CheckpointingMode
AT_LEAST_ONCE CheckpointingMode
Press Enter to insert, Tab to replace

Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ **Настройка exactly once**
- ❑ Эволюция приложения
- ❑ Непрерывная работа Flink-задания
- ❑ Увеличение нагрузки. RocksDB
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

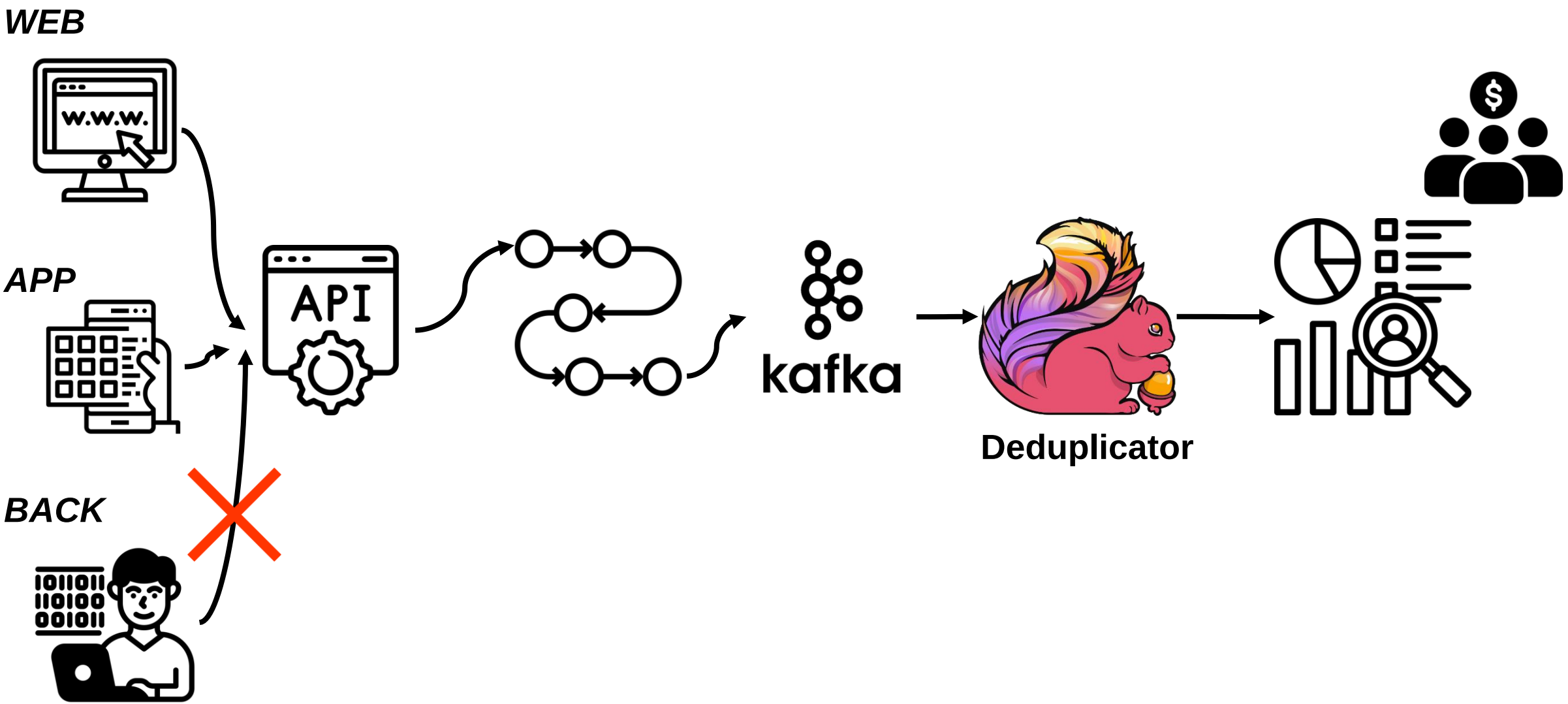
Текущая реализация задания



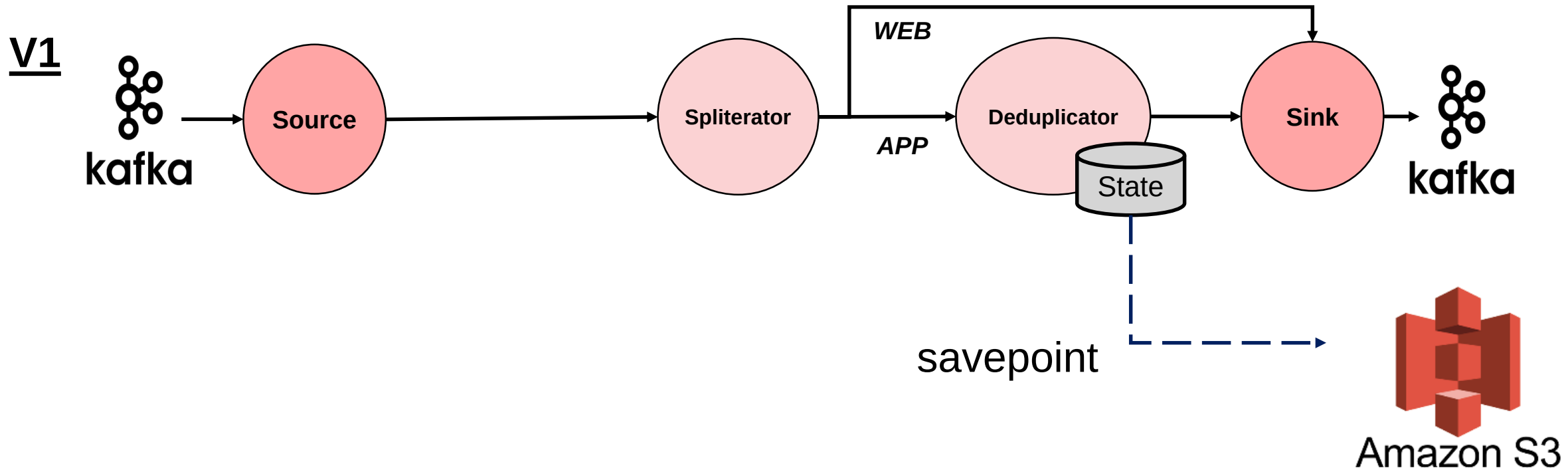
Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ❑ **Эволюция приложения**
- ❑ Непрерывная работа Flink-задания
- ❑ Увеличение нагрузки. RocksDB
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

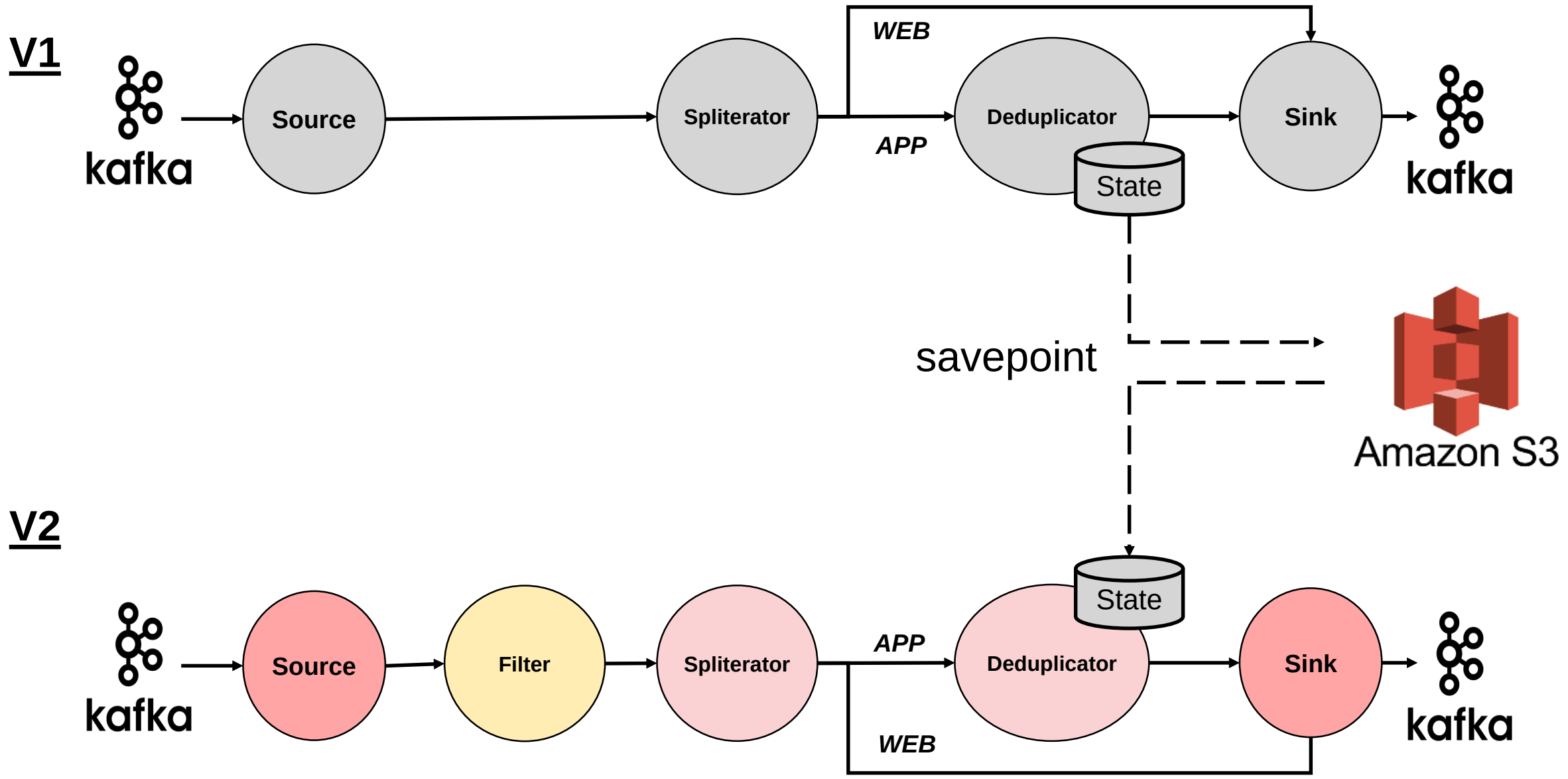
Эволюция графа задания



Эволюция графа задания



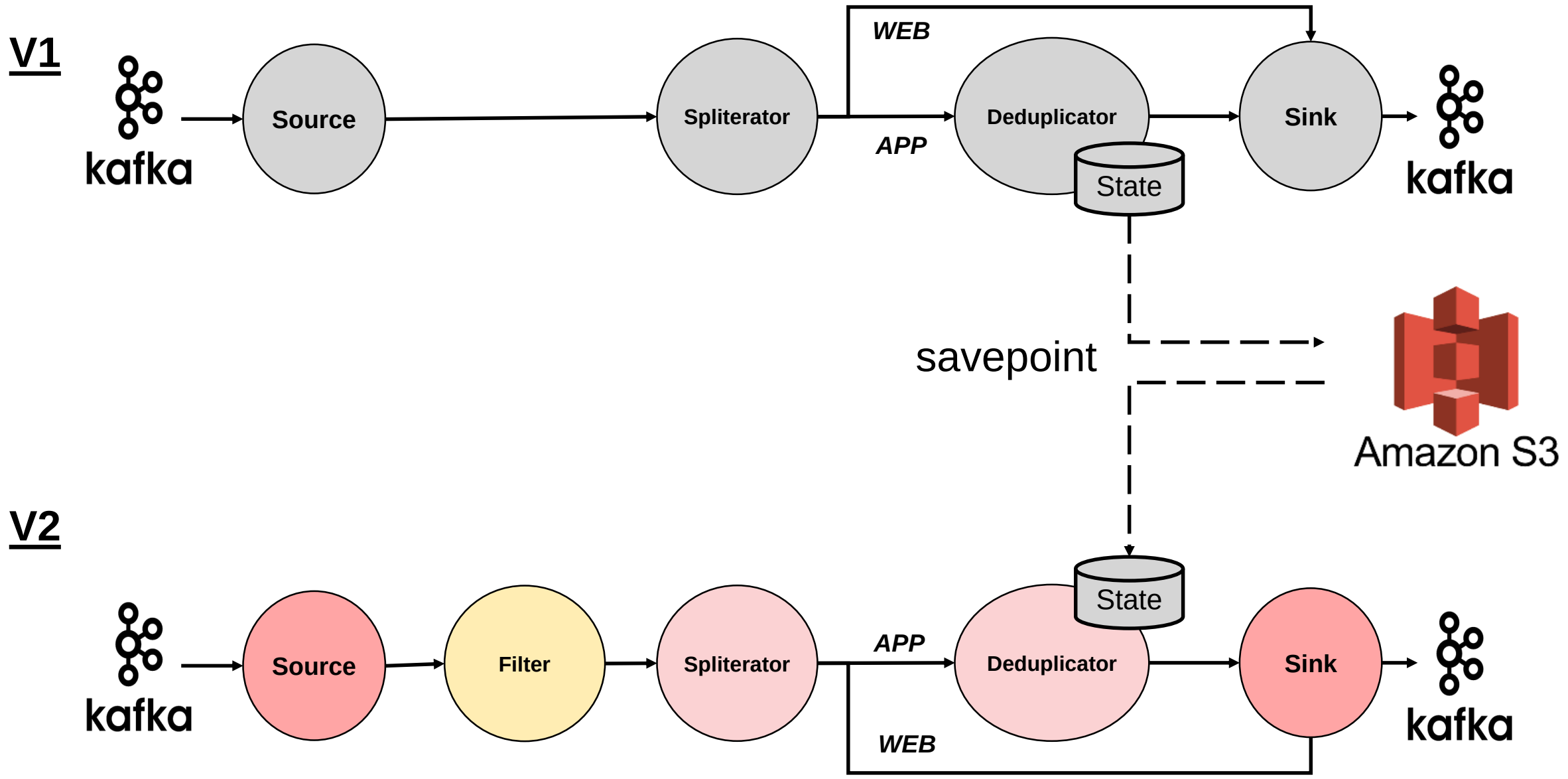
Эволюция графа задания



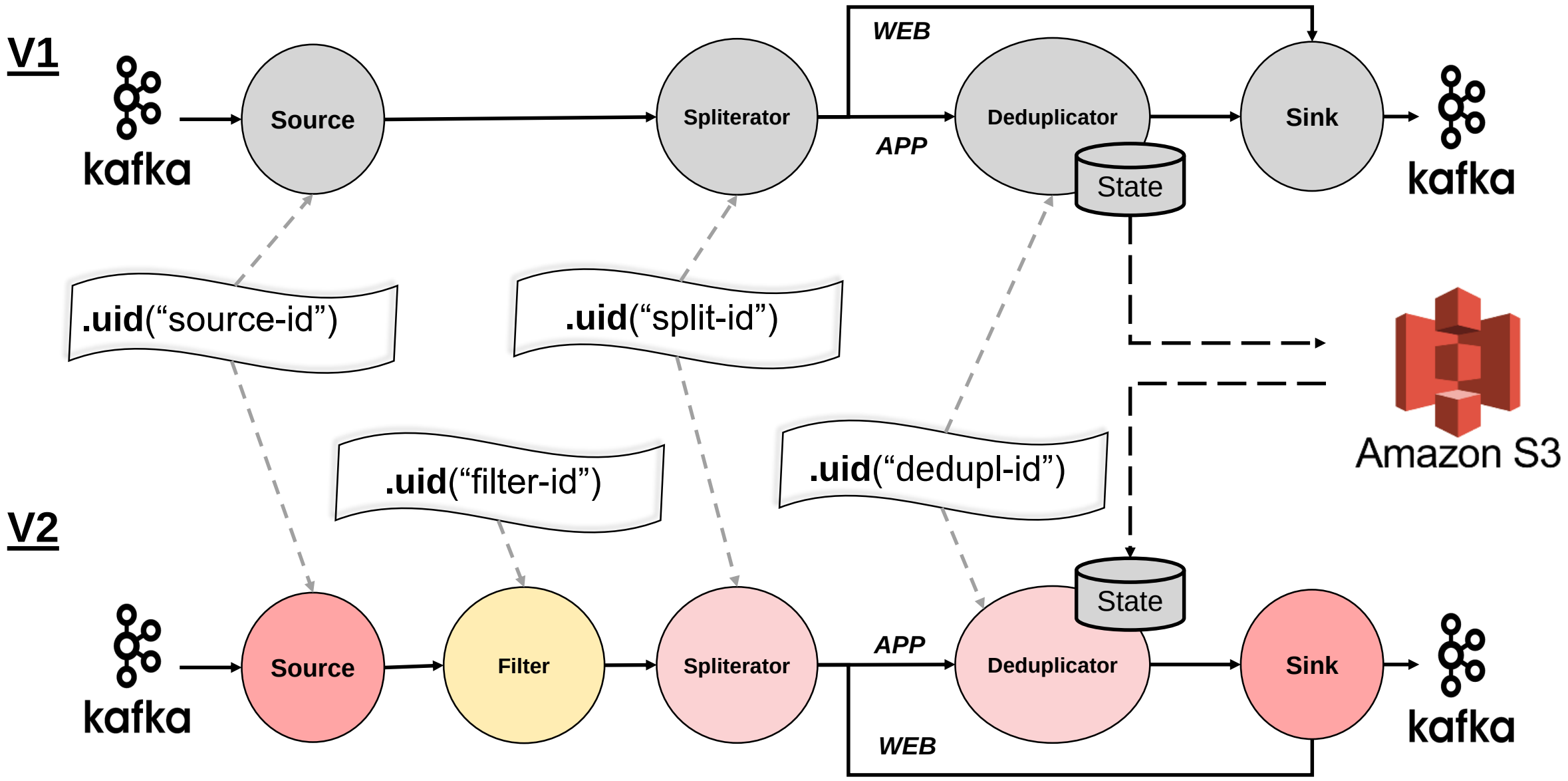
Эволюция графа задания

```
17:54:22.163 [jobmanager-io-thread-1] INFO org.apache.flink.runtime.checkpoint.CheckpointCoordinator
- Starting job 5d2e3e2f39b3edf2ba0c911c6838fba4 from savepoint
file:/var/folders/_y/gd8sxnq91z9glrxjlkrj98tnbsxn37/T/junit11984132124657901599/savepoints/savepoint
-7f452b-fa4b970123df ()
17:54:22.173 [jobmanager-io-thread-1] DEBUG org.apache.flink.runtime.jobmaster
.DefaultJobMasterServiceProcess - Initialization of the JobMasterService for job
5d2e3e2f39b3edf2ba0c911c6838fba4 under leader id fa8ec7f8-d69c-4310-9ccf-e1bd67c7d5b8 failed.
org.apache.flink.runtime.client.JobInitializationException Create breakpoint : Could not start the
JobMaster.
    at org.apache.flink.runtime.jobmaster.DefaultJobMasterServiceProcess.lambda$new$0
    (DefaultJobMasterServiceProcess.java:97) <7 internal lines>
Caused by: java.util.concurrent.CompletionException Create breakpoint : java.lang.IllegalStateException:
Failed to rollback to checkpoint/savepoint
file:/var/folders/_y/gd8sxnq91z9glrxjlkrj98tnbsxn37/T/junit11984132124657901599/savepoints/savepoint
-7f452b-fa4b970123df. Cannot map checkpoint/savepoint state for operator
20ba6b65f97481d5570070de90e4e791 to the new program, because the operator is not available in the
new program. If you want to allow to skip this, you can set the --allowNonRestoredState option on
the CLI.
    at java.base/java.util.concurrent.CompletableFuture.encodeThrowable(CompletableFuture.java:314)
    at java.base/java.util.concurrent.CompletableFuture.completeThrowable(CompletableFuture.java:319)
```

Эволюция графа задания

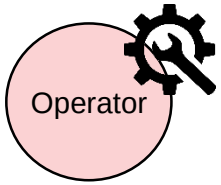


Эволюция графа задания

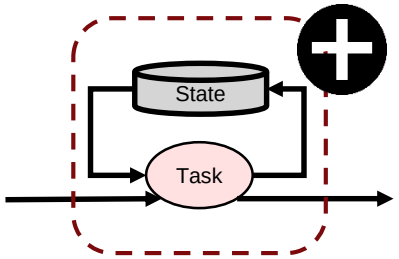


Обновление приложения с сохранением состояния

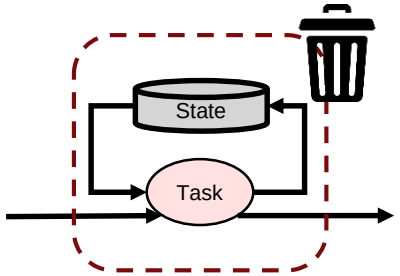
Исходное приложение и новая версия приложения должны быть **совместимы** с точкой сохранения savepoint.
Поэтому приложение нужно развивать, соблюдая правила.



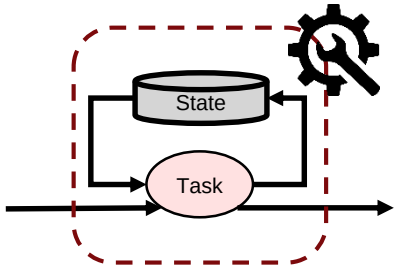
Изменяем/добавляем текущие операторы **без состояния**



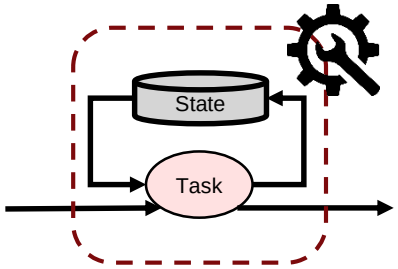
Добавляем новые операторы **с состоянием**



Удаляем состояния операторов (нужно указать флаг отключения проверки безопасности сопоставления состояний)



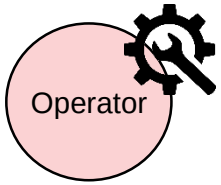
Изменяем **примитив** состояния (тип данных внутри состояния)



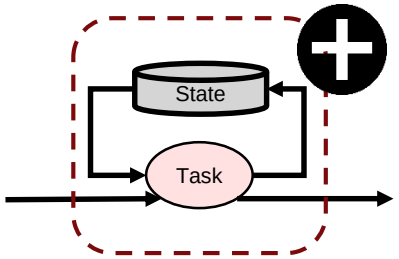
Произвольно изменяем состояния

Обновление приложения с сохранением состояния

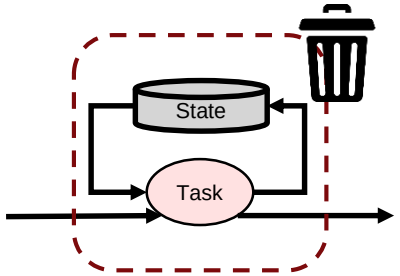
Исходное приложение и новая версия приложения должны быть **совместимы** с точкой сохранения savepoint.
Поэтому приложение нужно развивать, соблюдая правила.



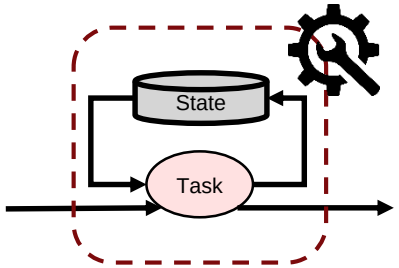
Изменяем/добавляем текущие операторы **без состояния**



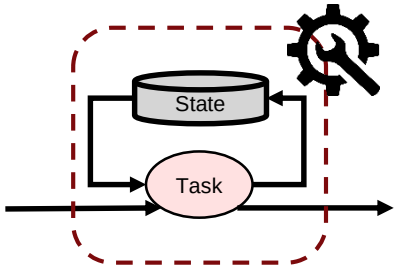
Добавляем новые операторы **с состоянием**



Удаляем состояния операторов (нужно указать флаг отключения проверки безопасности сопоставления состояний)



Изменяем **примитив** состояния (тип данных внутри состояния)

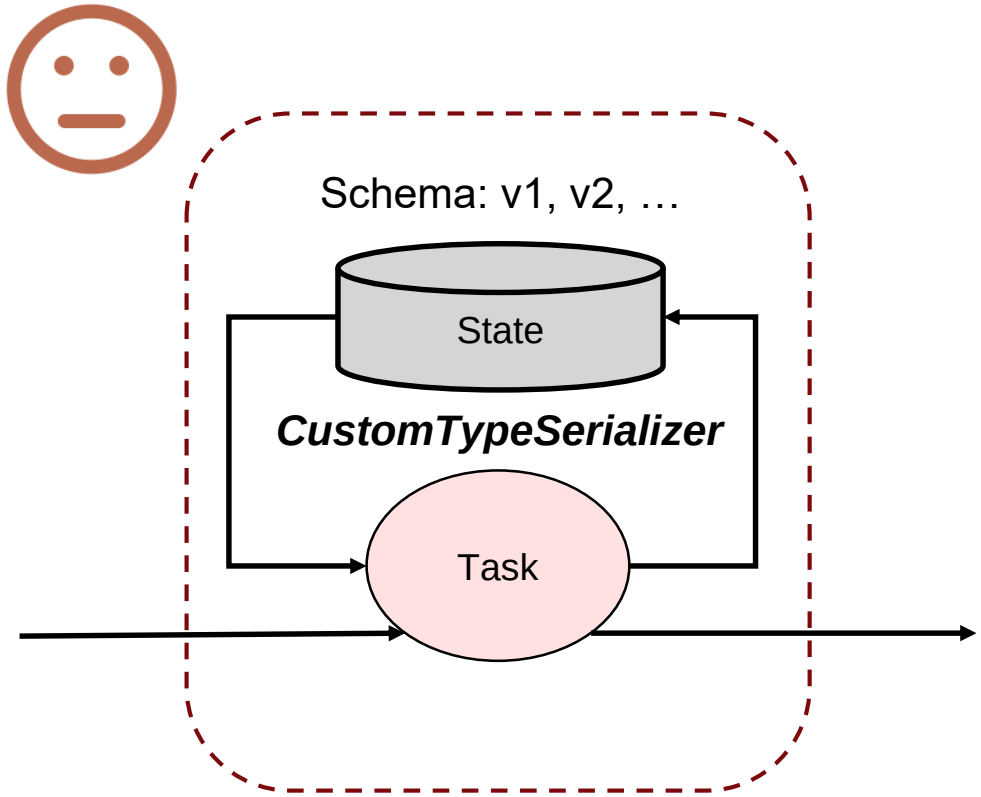


Произвольно изменяем состояния

Произвольное изменение формата состояния

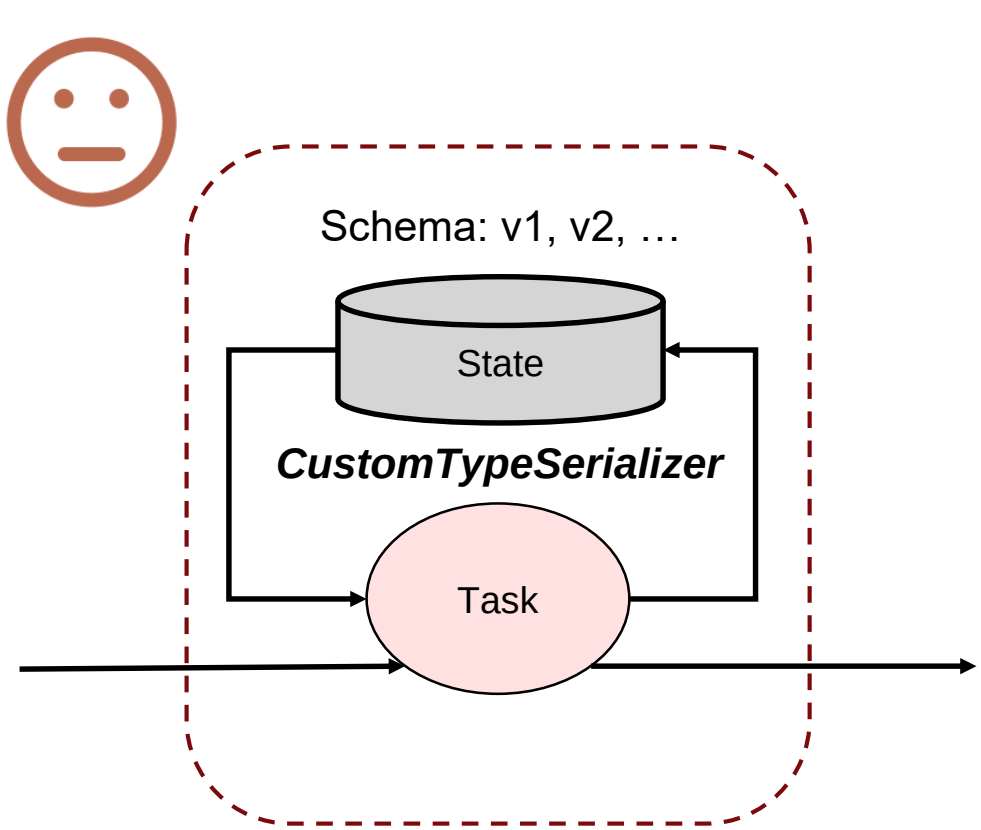
Произвольное изменение формата состояния

1. “Умный” сериализатор

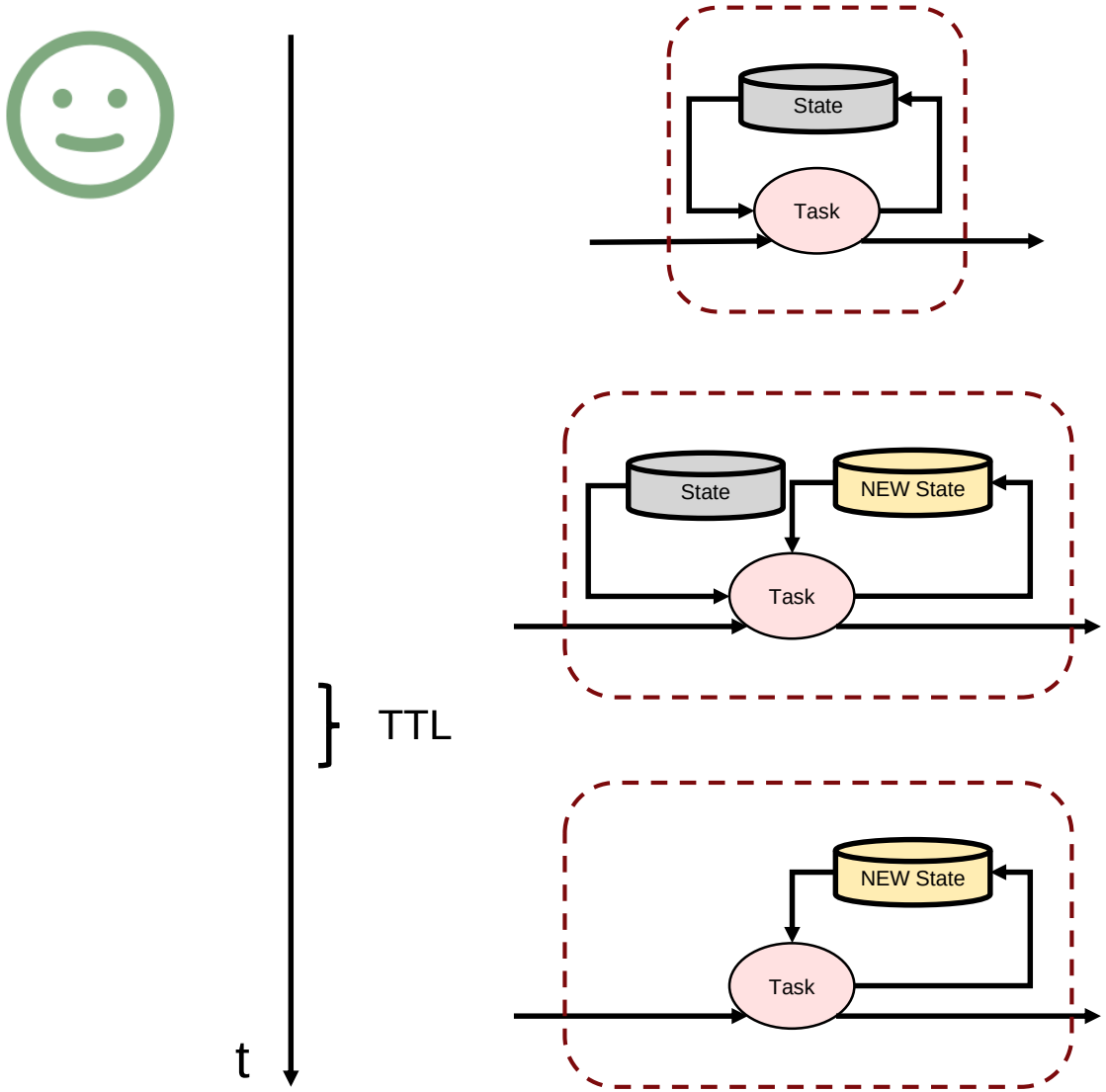


Произвольное изменение формата состояния

1. “Умный” сериализатор



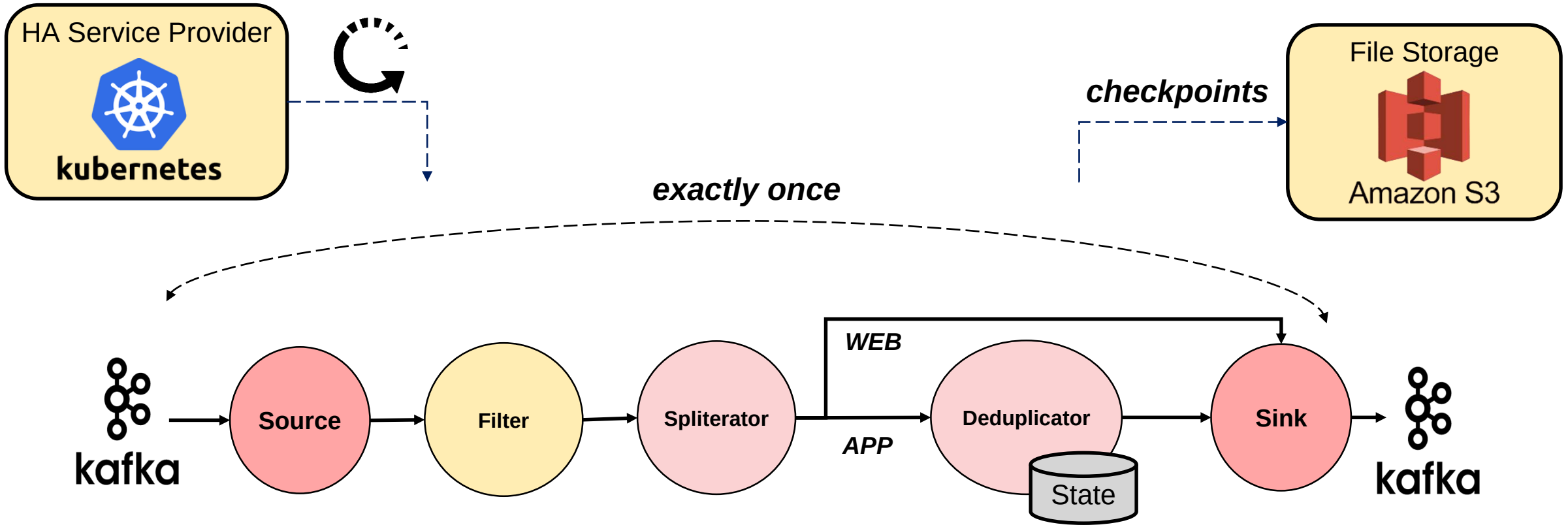
2. Плавный переход на второе “новое” состояние



Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ **Эволюция приложения**
- ❑ Непрерывная работа Flink-задания
- ❑ Увеличение нагрузки. RocksDB
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

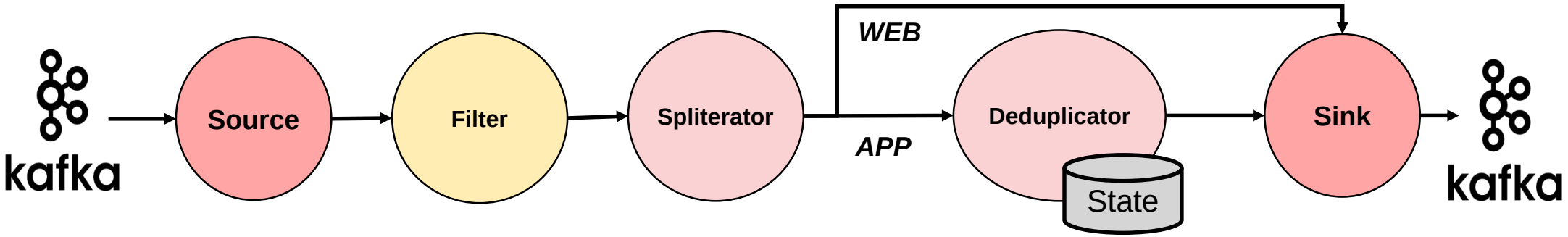
Текущая реализация задания



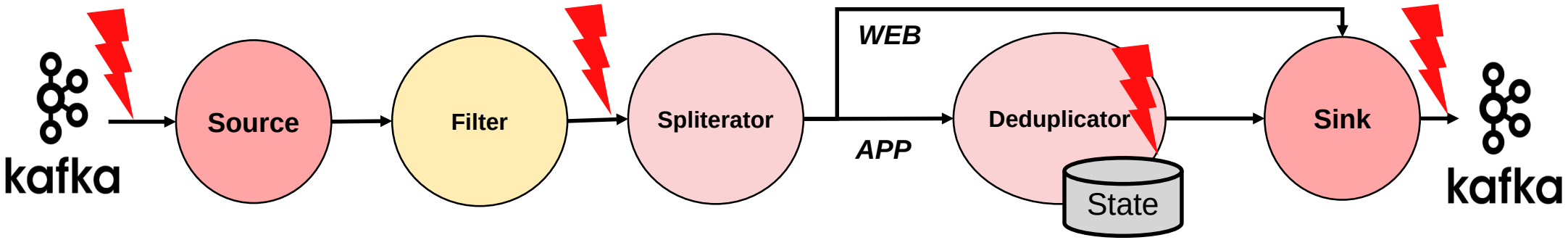
Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ❑ **Непрерывная работа Flink-задания**
- ❑ Увеличение нагрузки. RocksDB
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

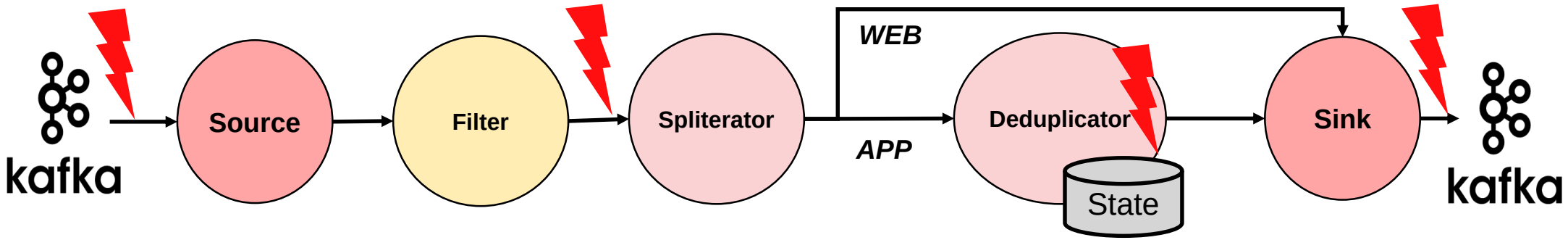
Стратегии обработки ошибок



Стратегии обработки ошибок



Стратегии обработки ошибок



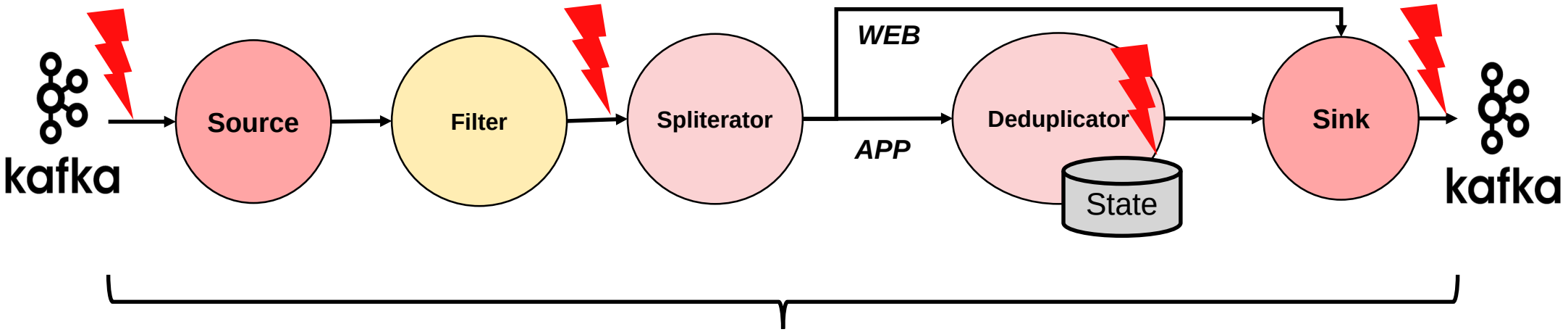
Типы ошибок:

- Постоянные (невалидное событие, ошибка в коде …)
- Временные (инфраструктурные …)

Стратегии обработки:

- Пропуск событий
- Повтор обработки событий
- Исправление и перезапуск Flink-задания
- DLQ (Dead Letter Queue)

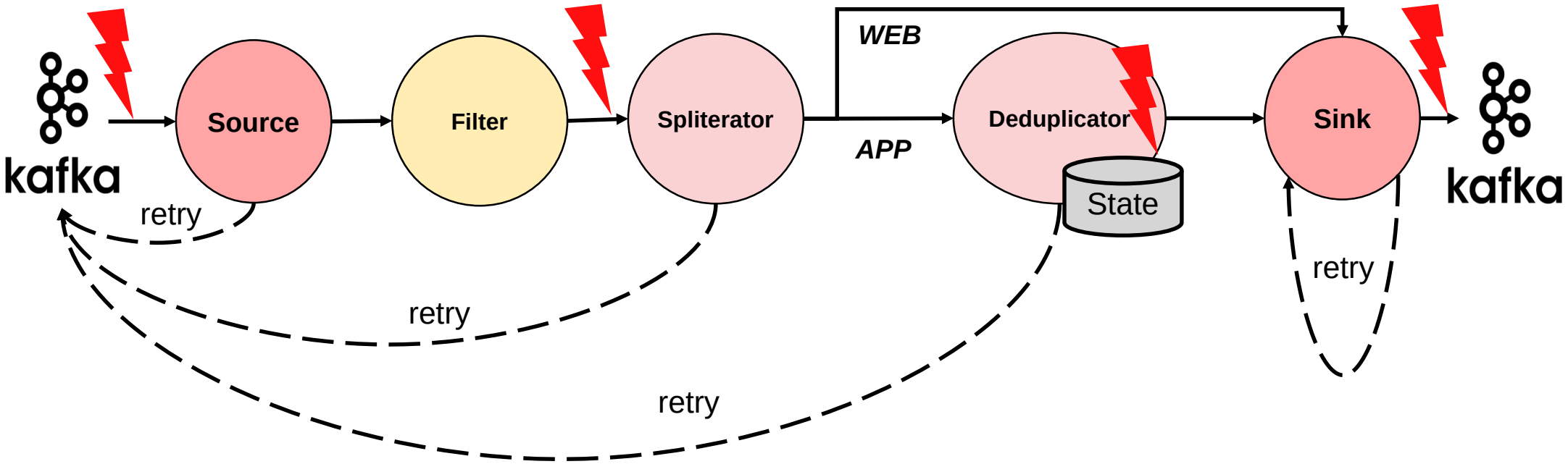
Пропуск событий



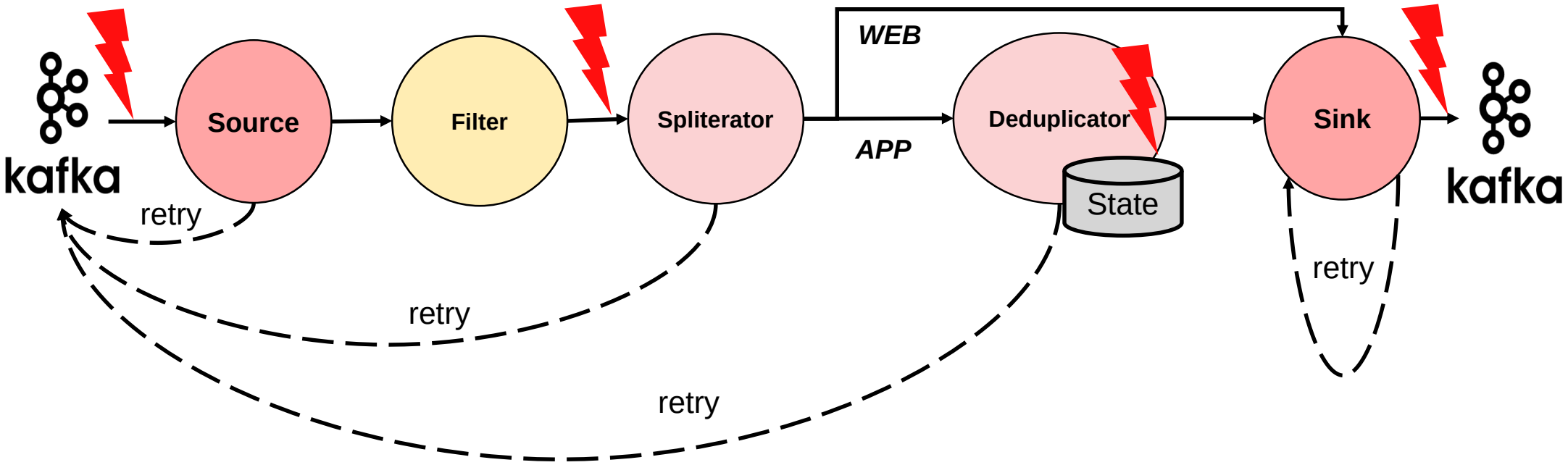
Grafana

Stack

Повтор обработки событий

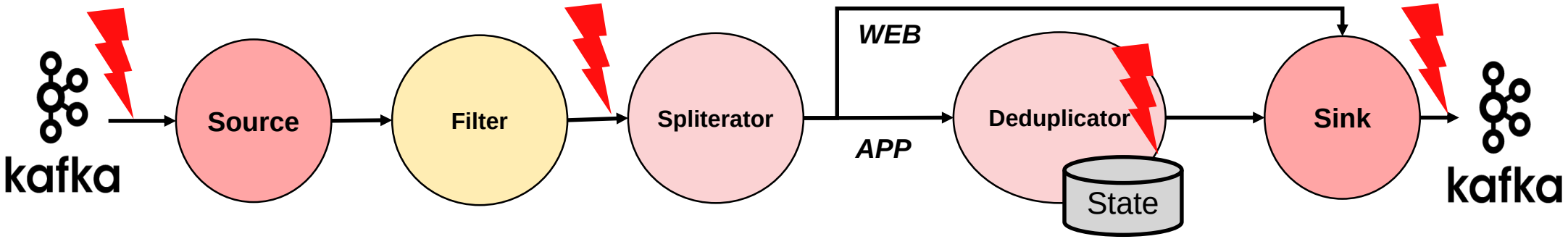


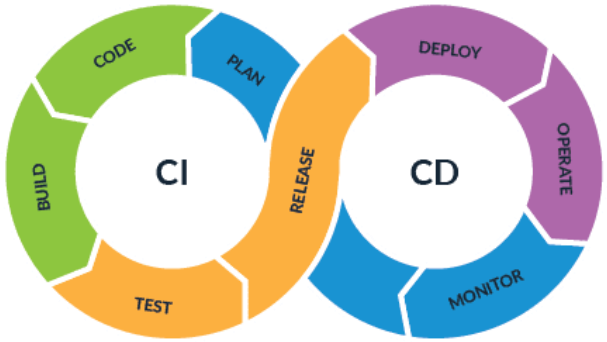
Повтор обработки событий



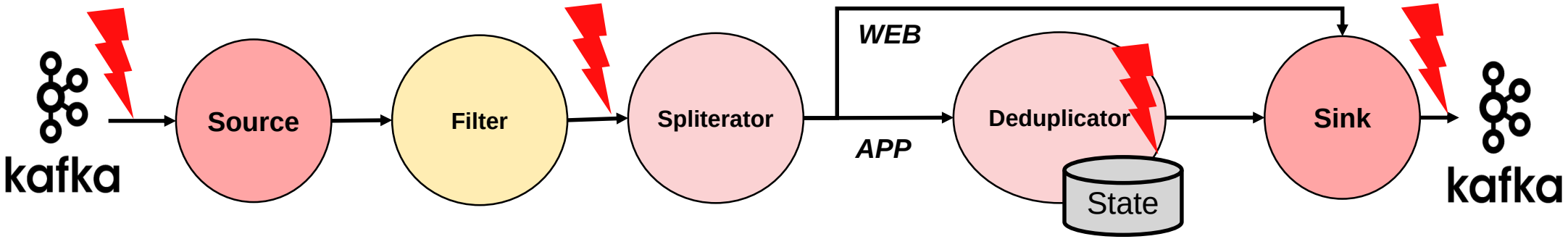
Бессмысленно для постоянных типов ошибок!

Исправление и перезапуск Flink-задания

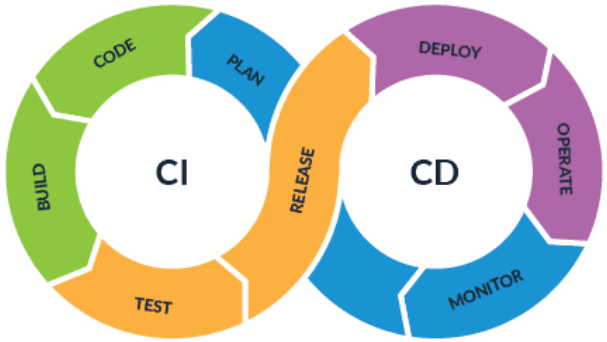




Исправление и перезапуск Flink-задания

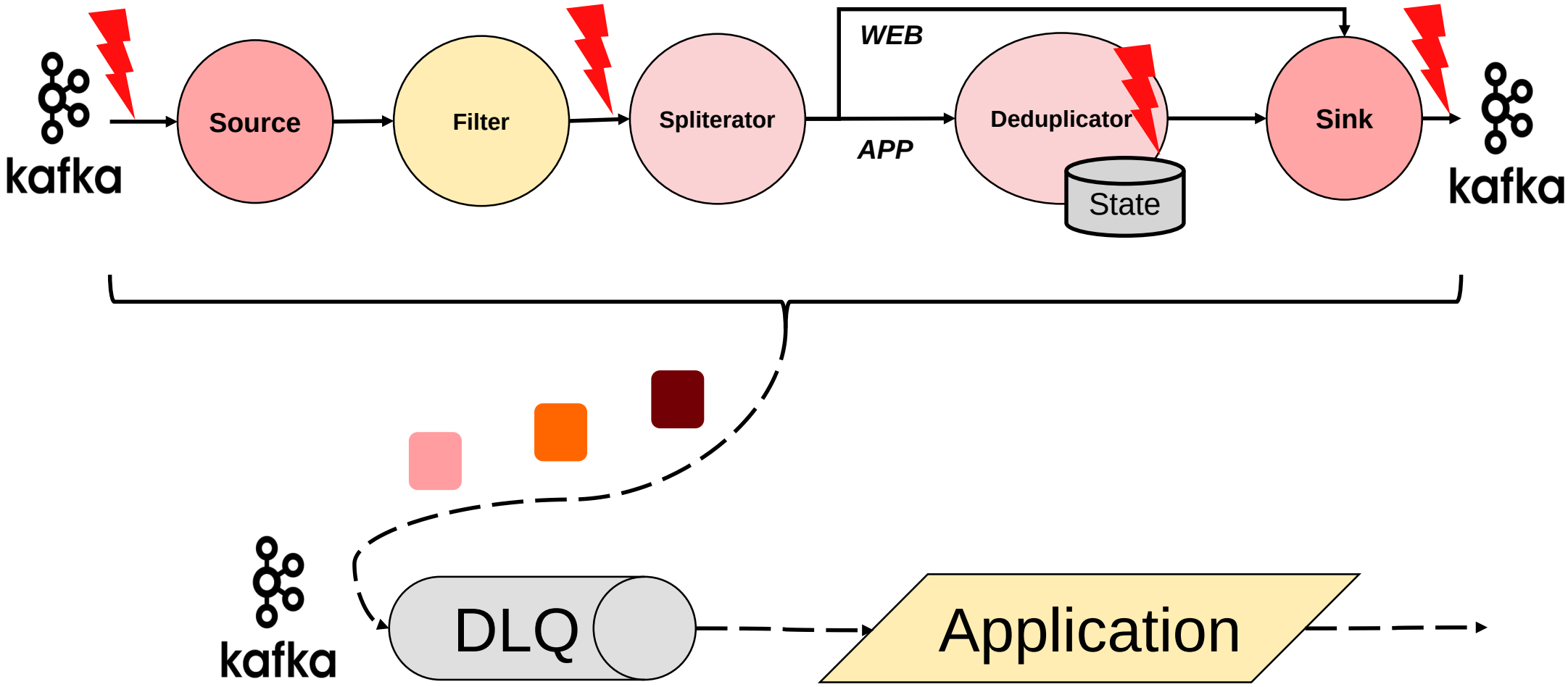


Stack

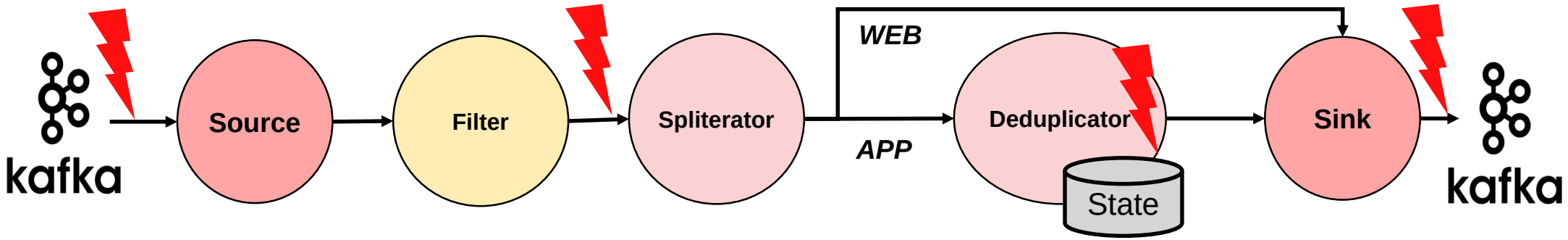


Real time?

Dead Letter Queue (DLQ)



Стратегии обработки ошибок



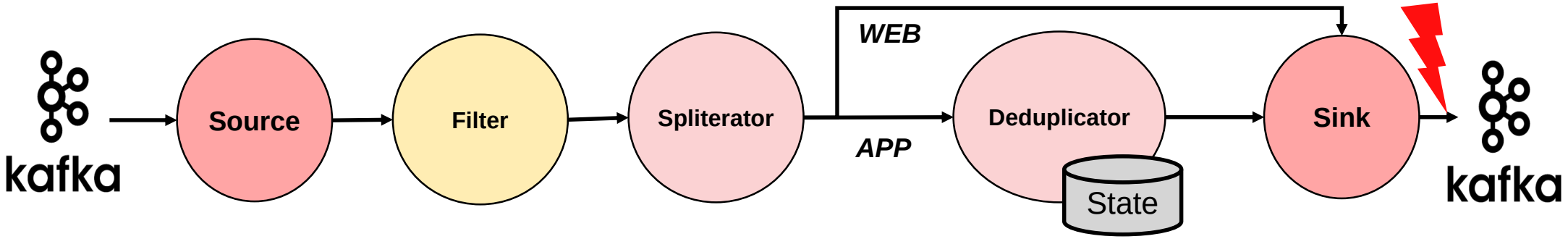
Типы ошибок:

- Постоянные (невалидное событие, ошибка в коде …)
- Временные (инфраструктурные …)

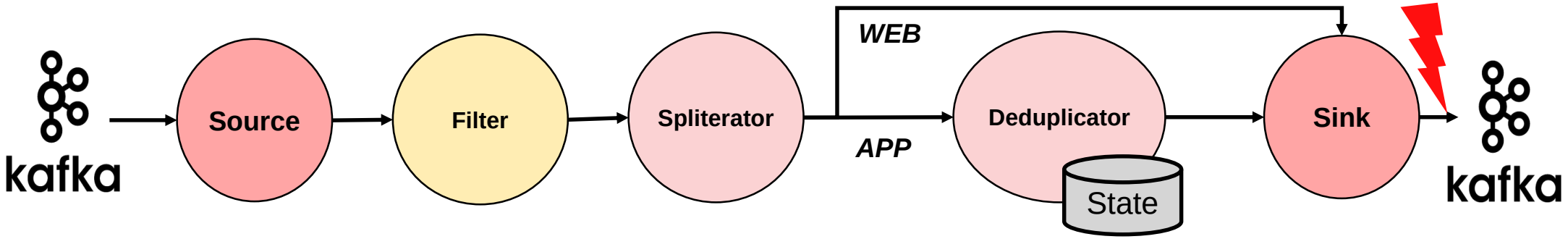
Стратегии обработки:

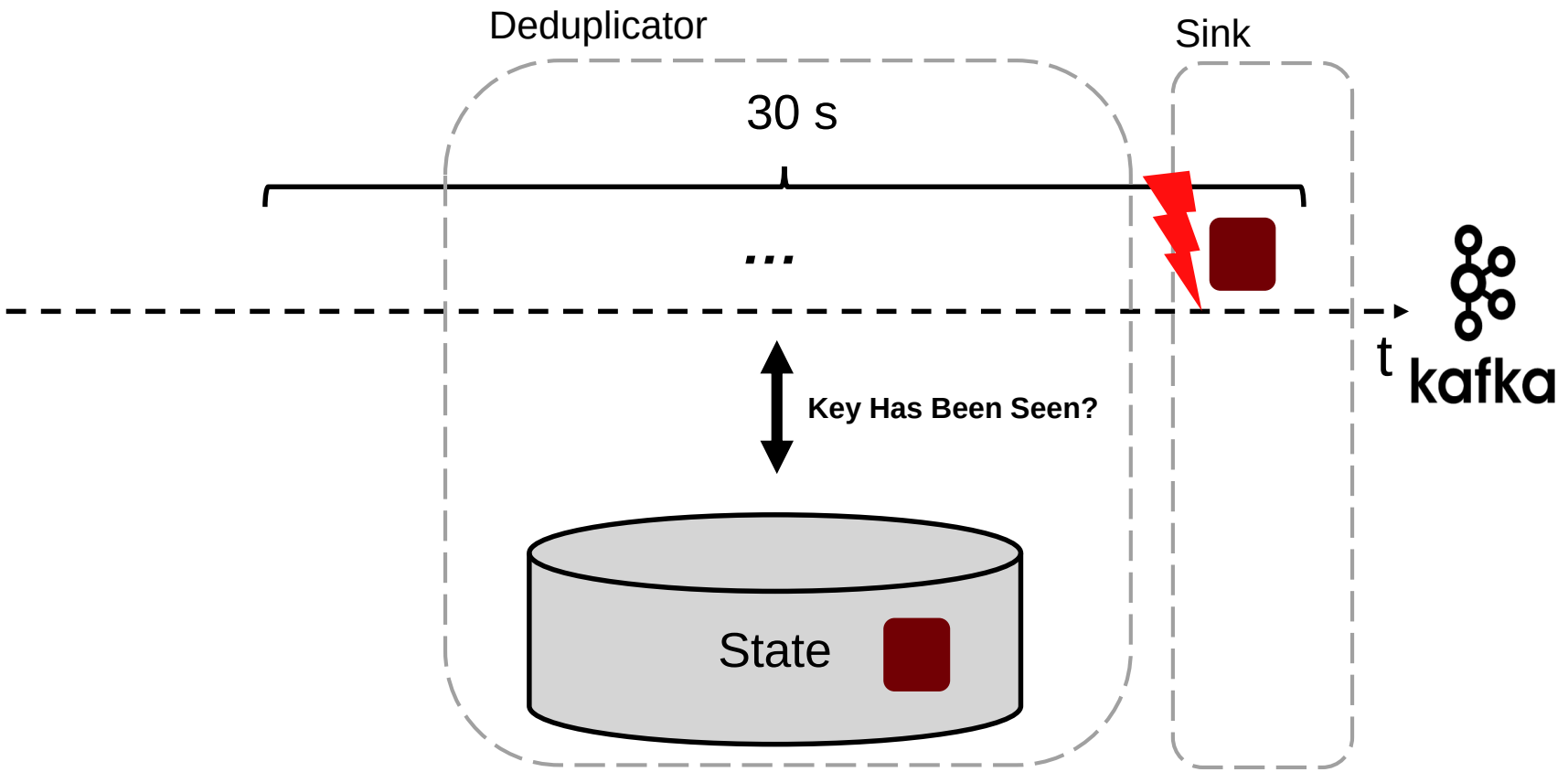
- Пропуск событий
- Повтор обработки событий
- Исправление и перезапуск Flink-задания
- DLQ (Dead Letter Queue)

Сложности для Stateful Operator

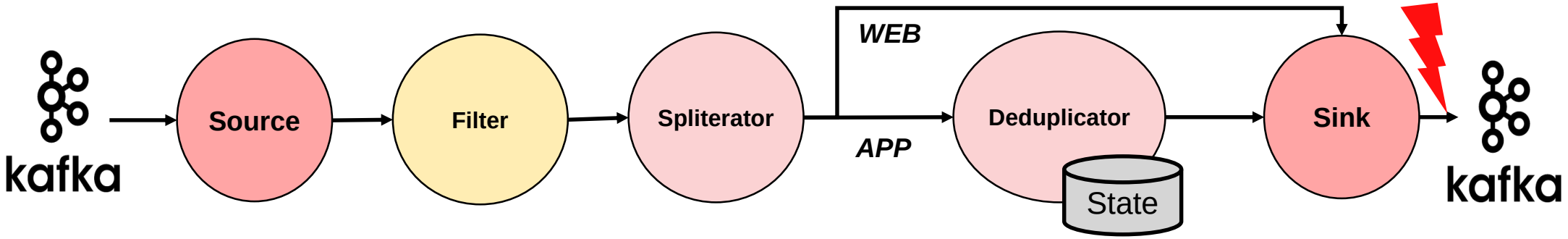


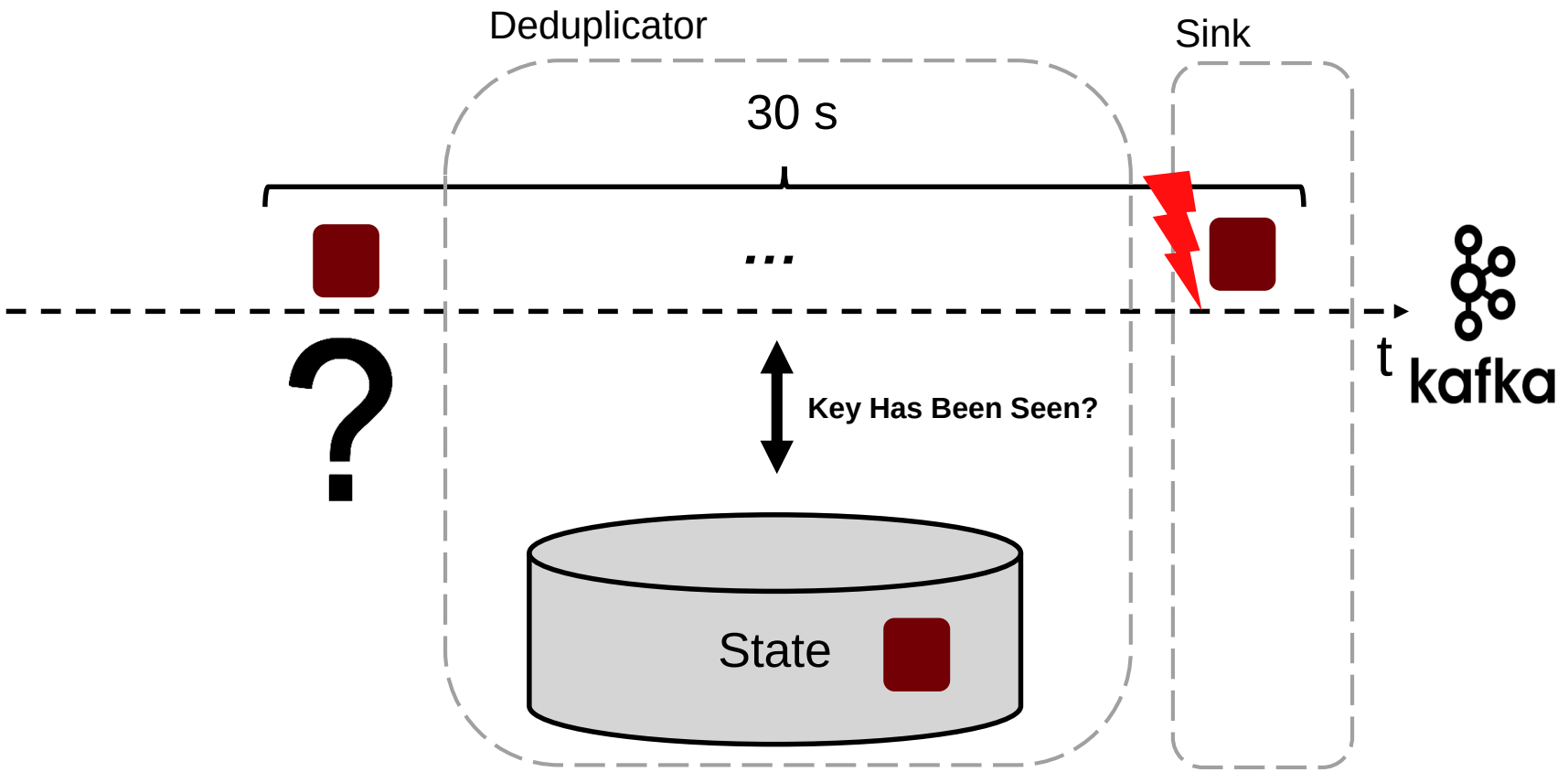
Сложности для Stateful Operator



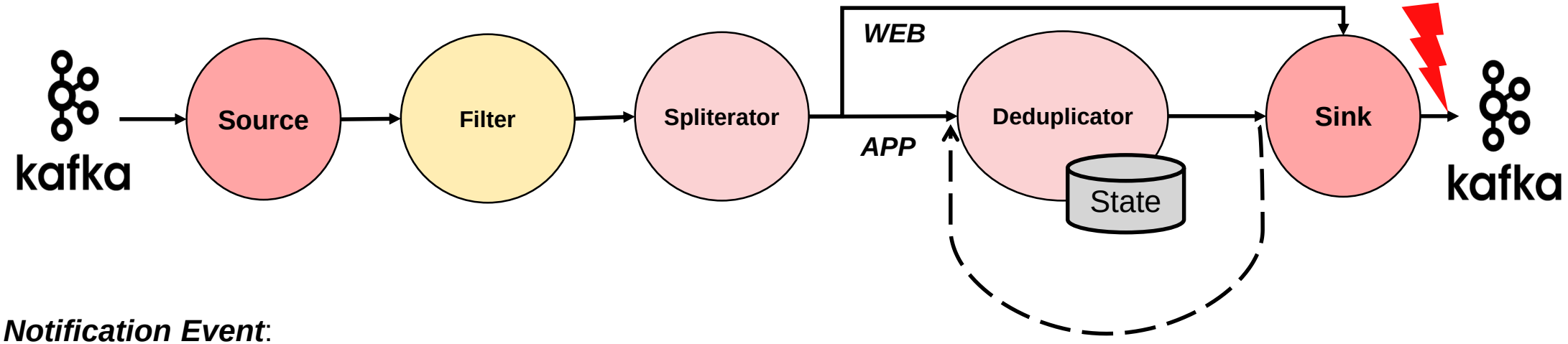


Сложности для Stateful Operator





Сложности для Stateful Operator

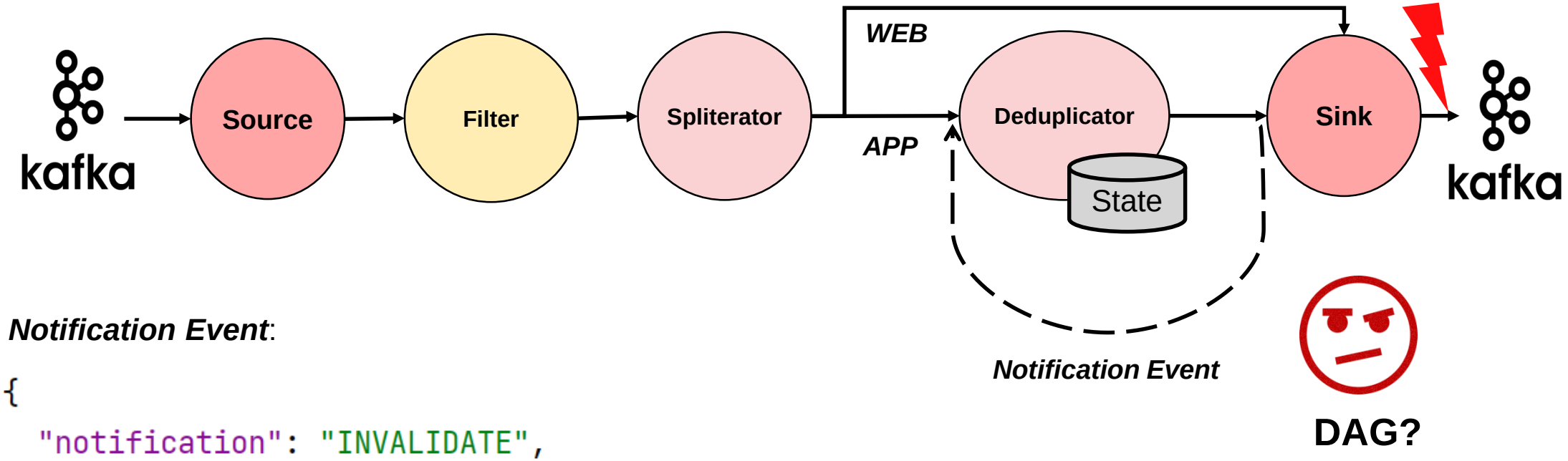


Notification Event:

```
{
  "notification": "INVALIDATE",
  "exception": "...",
  "event": {
    "id": "9ea7f44f-822b-4569-84ea-6dc6ffcaf19f",
    "uniqueValue": "656b4183-282c-1689019596",
    "type": "WEB",
    "timestamp": 1689019717
  }
}
```

Применимо, если TTL >> реакции на Notification Event, но все равно можем потерять событие

Сложности для Stateful Operator



DAG?

Notification Event:

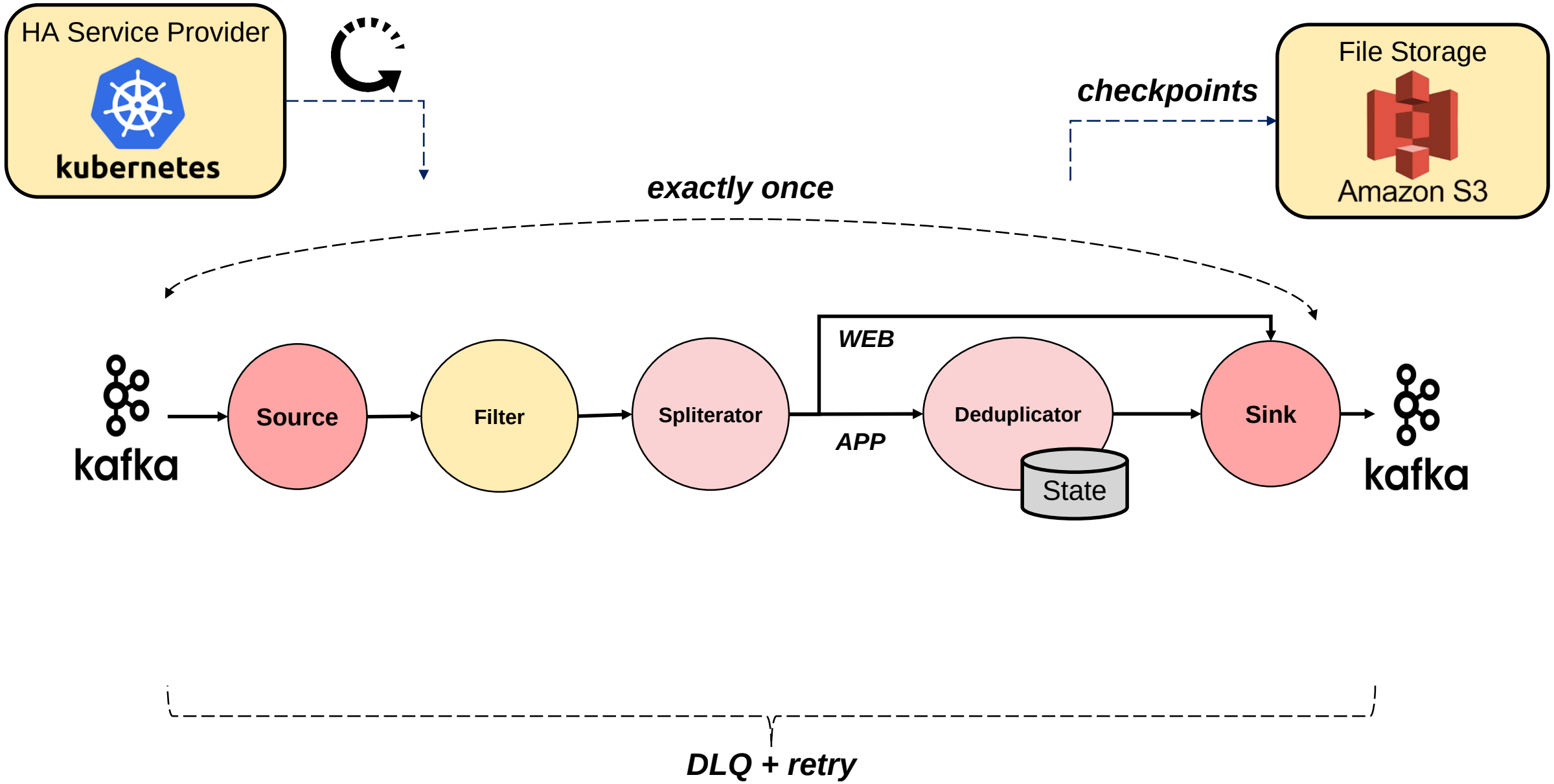
```
{
  "notification": "INVALIDATE",
  "exception": "...",
  "event": {
    "id": "9ea7f44f-822b-4569-84ea-6dc6ffcaf19f",
    "uniqueValue": "656b4183-282c-1689019596",
    "type": "WEB",
    "timestamp": 1689019717
  }
}
```

Применимо, если TTL >> реакции на Notification Event, но все равно можем потерять событие

Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ **Непрерывная работа Flink-задания**
- ☐ Увеличение нагрузки. RocksDB
- ☐ Работа с большим состоянием
- ☐ Ускорение работы Flink-задания

Текущая реализация задания



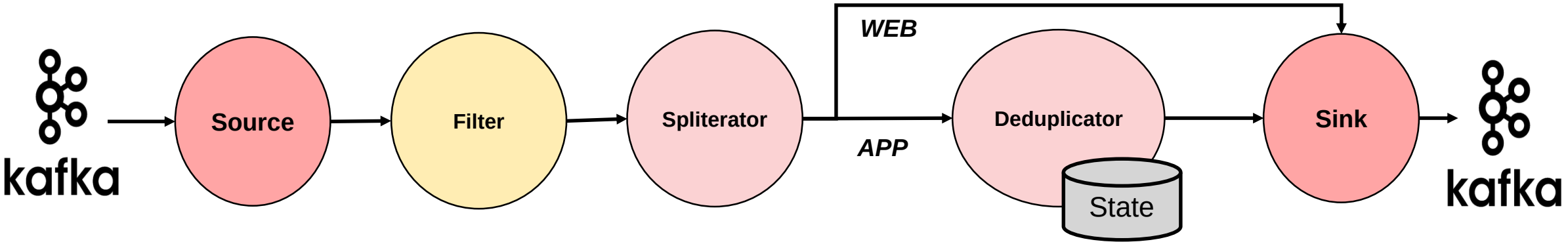
Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ Непрерывная работа Flink-задания
- ❑ **Увеличение нагрузки. RocksDB**
- ❑ Работа с большим состоянием
- ❑ Ускорение работы Flink-задания

Увеличение нагрузки

Исходные данные:

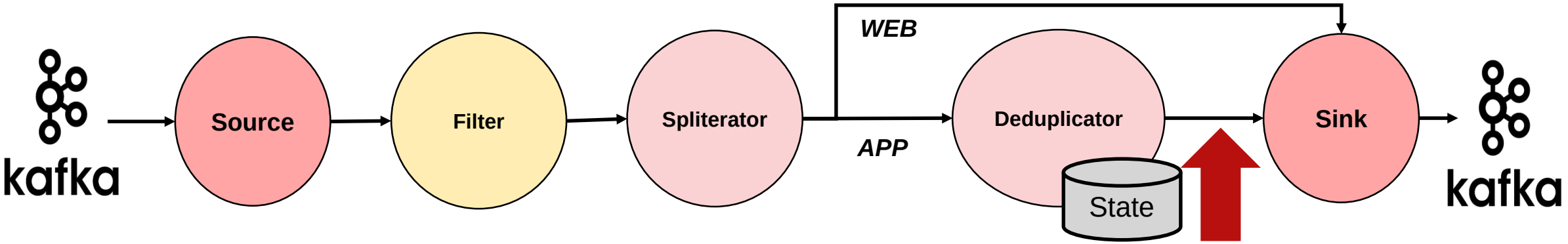
- Дедупликация WEB в окне 30 s → **10 min**
- Нагрузка ~100 rps → **100 000 rps**



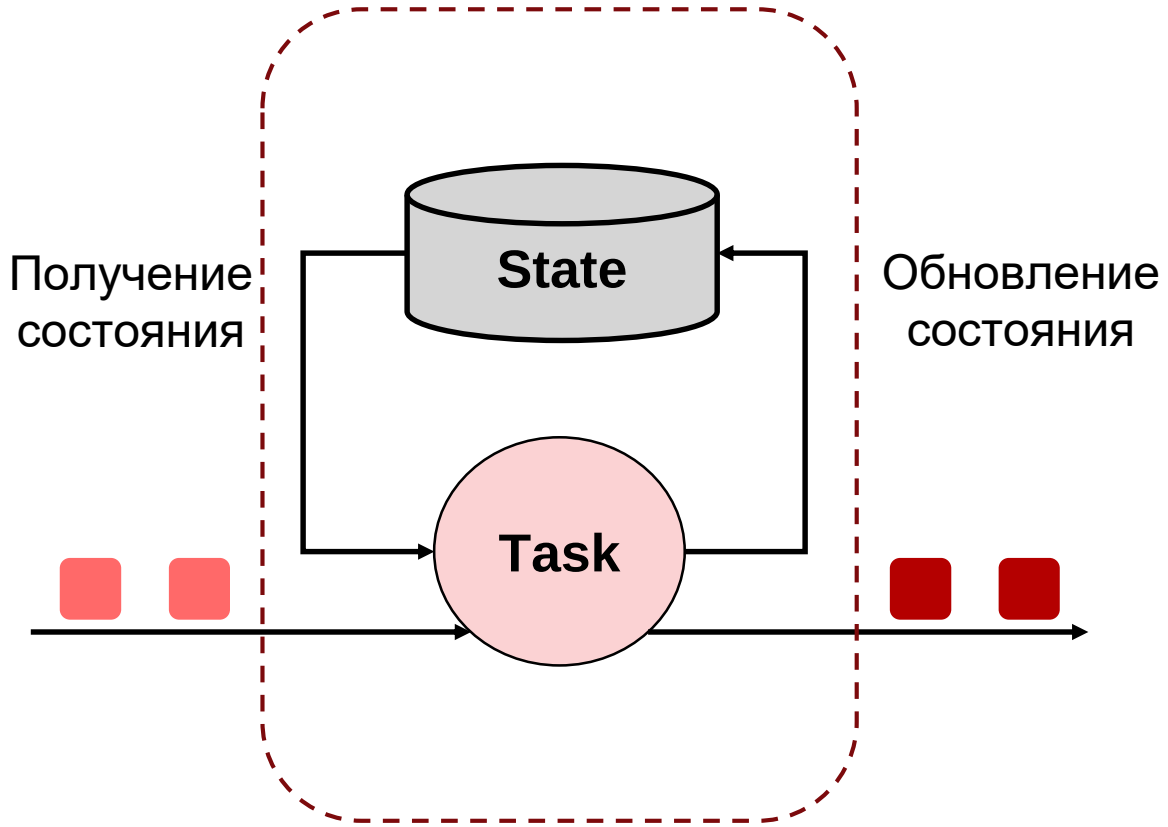
Увеличение нагрузки

Исходные данные:

- Дедупликация WEB в окне 30 s → **10 min**
- Нагрузка ~100 rps → **100 000 rps**



Выбираем реализацию состояния



Java Heap

- Локальная разработка, отладка

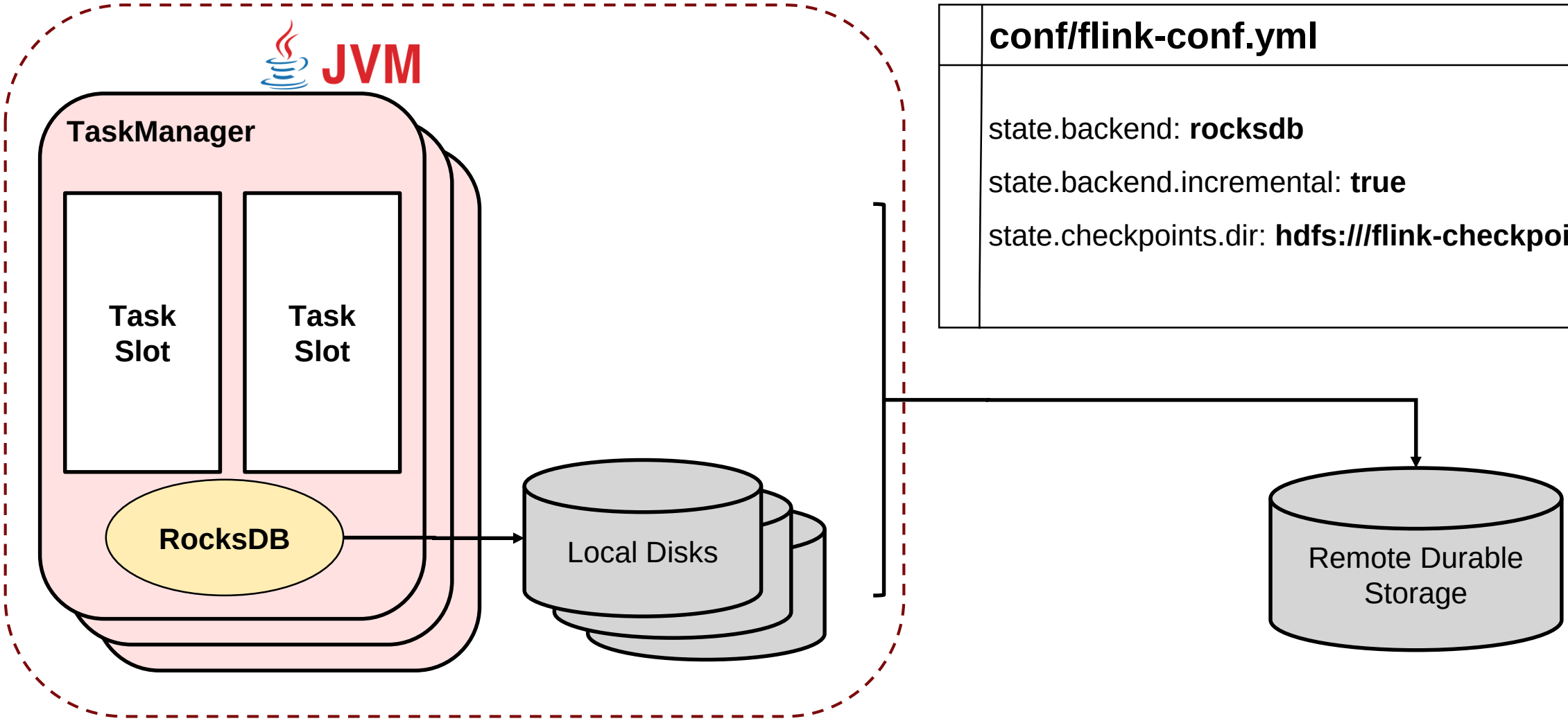
RocksDB

- Большой размер состояния
- Предсказуемая задержка (нет влияния JVM GC)
- Ускоряет время checkpoint (асинхронные, инкрементальные)
- де/сериализация при чтении/записи

File System State

- Deprecated

RocksDB



conf/flink-conf.yml

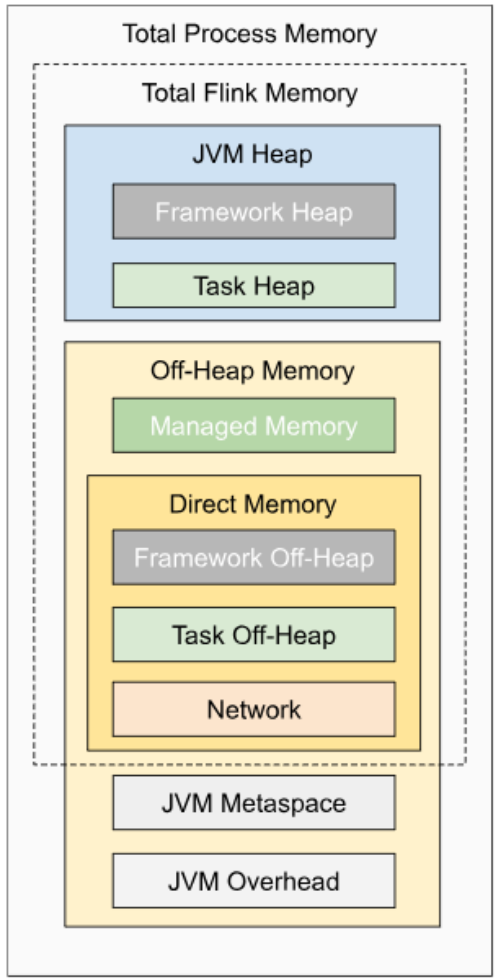
state.backend: **rocksdb**

state.backend.incremental: **true**

state.checkpoints.dir: **hdfs:///flink-checkpoints**

Настройка памяти RocksDB

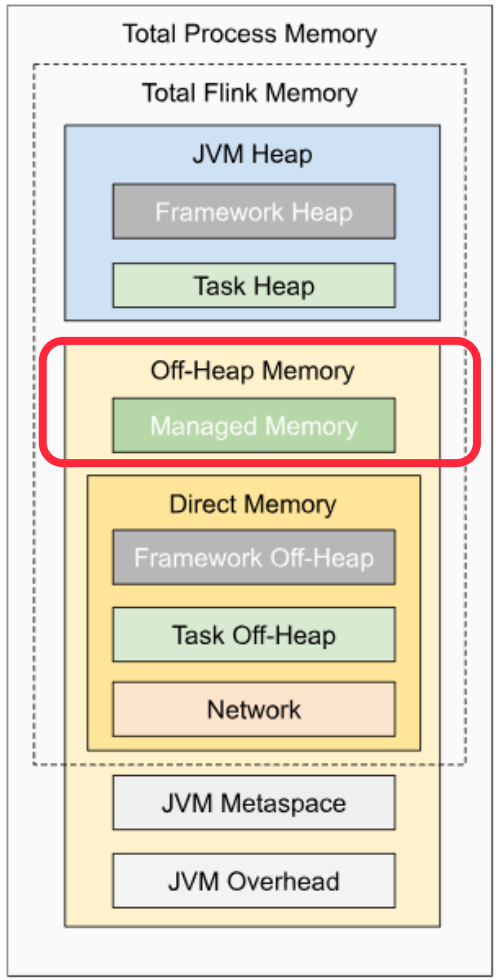
Memory



Flink Memory Model	Effective Configuration ⓘ	Metric
Framework Heap	128 MB	7.18% 36.7 MB / 512 MB ⓘ
Task Heap	384 MB	
Managed Memory	512 MB	0% 0 B / 512 MB
Framework Off-Heap	128 MB	ⓘ
Task Off-Heap	0 B	
Network	128 MB	0% 0 B / 128 MB
JVM Metaspace	256 MB	14.53% 37.2 MB / 256 MB
JVM Overhead	192 MB	ⓘ

Настройка памяти RocksDB

Memory



Flink Memory Model	Effective Configuration ⓘ	Metric
Framework Heap	128 MB	7.18% 36.7 MB / 512 MB ⓘ
Task Heap	384 MB	
Managed Memory	512 MB	0% 0 B / 512 MB
Framework Off-Heap	128 MB	ⓘ
Task Off-Heap	0 B	
Network	128 MB	0% 0 B / 128 MB
JVM Metaspace	256 MB	14.53% 37.2 MB / 256 MB
JVM Overhead	192 MB	ⓘ

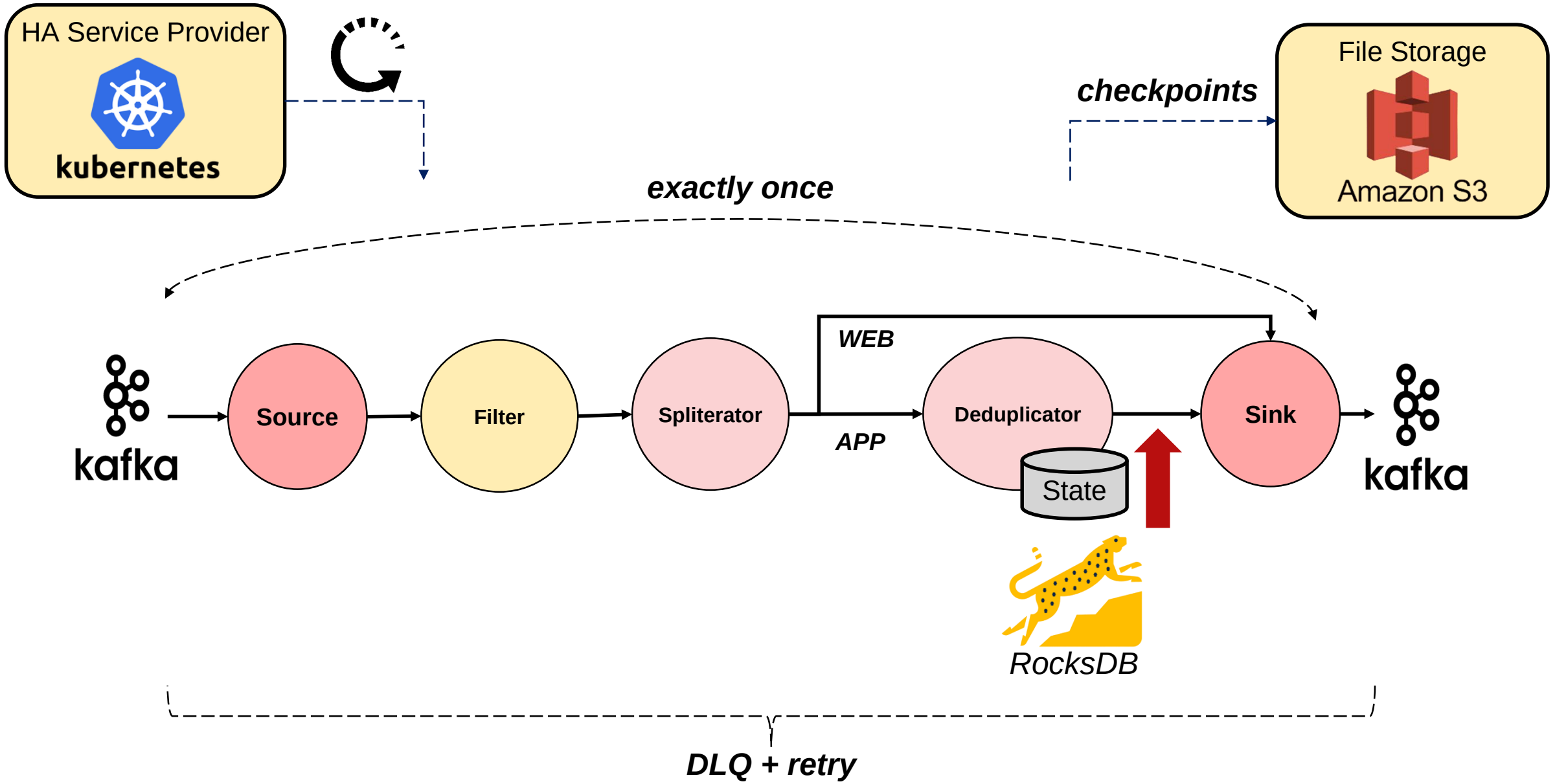
Устранение проблем производительности RocksDB

- Увеличение объема **Managed Memory**
- Уменьшение количества различных состояний в Flink-задании

Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ Непрерывная работа Flink-задания
- ☑ **Увеличение нагрузки. RocksDB**
- ☐ Работа с большим состоянием
- ☐ Ускорение работы Flink-задания

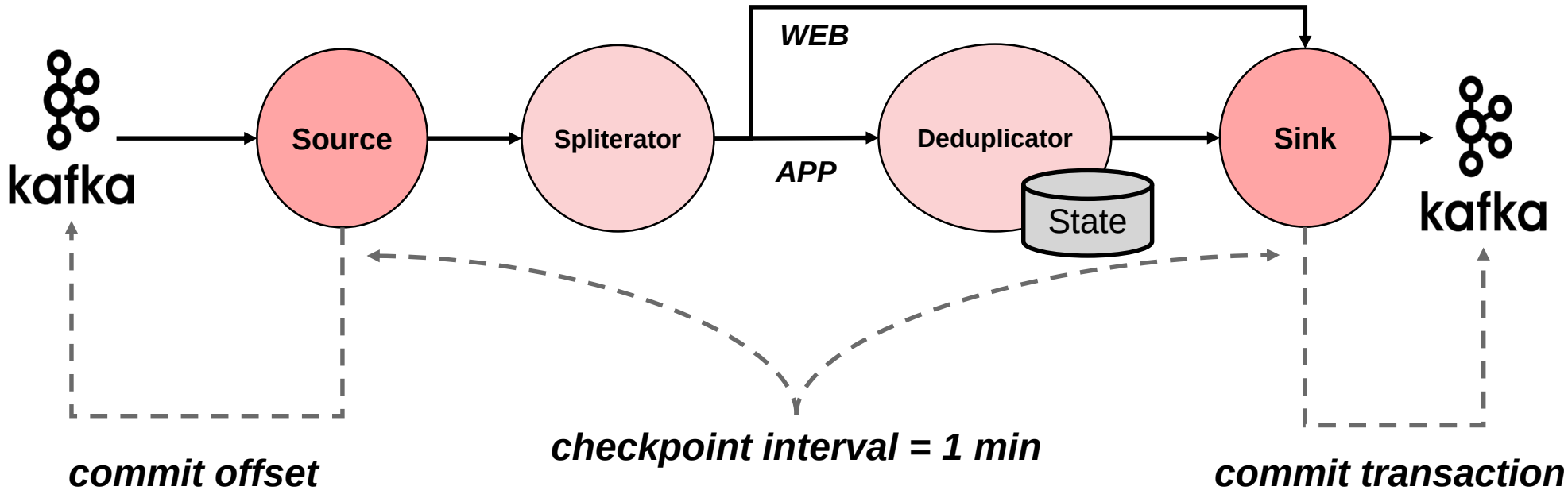
Текущая реализация задания



Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ Непрерывная работа Flink-задания
- ☑ Увеличение нагрузки. RocksDB
- ❑ **Работа с большим состоянием**
- ❑ Ускорение работы Flink-задания

Выбор размера checkpoint



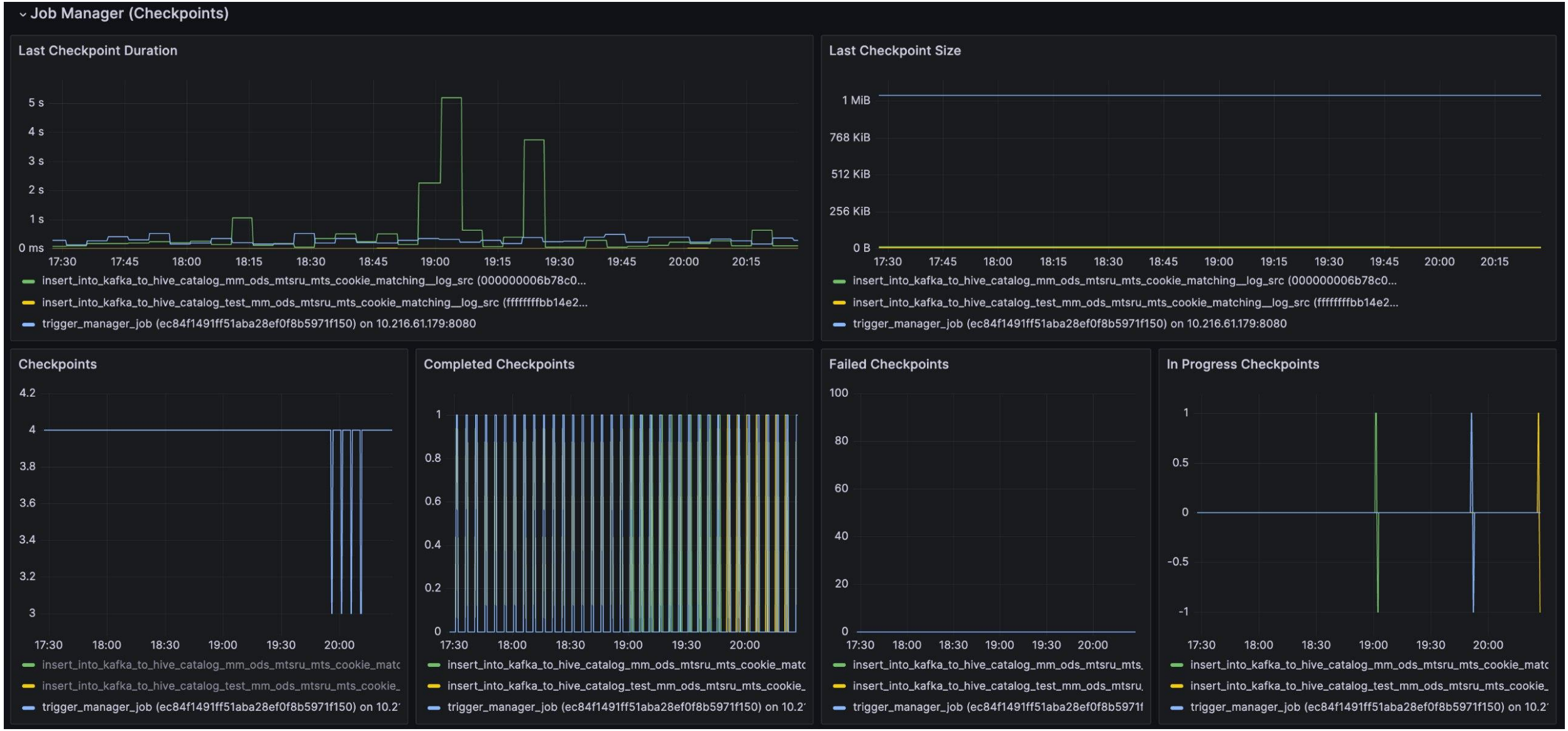
checkpoint interval = 10 min

- Большой размер checkpoint → длительное сохранение
- Редкие фиксации данных (задержки у потребителей)
- Повторная обработка многих записей в случае сбоя

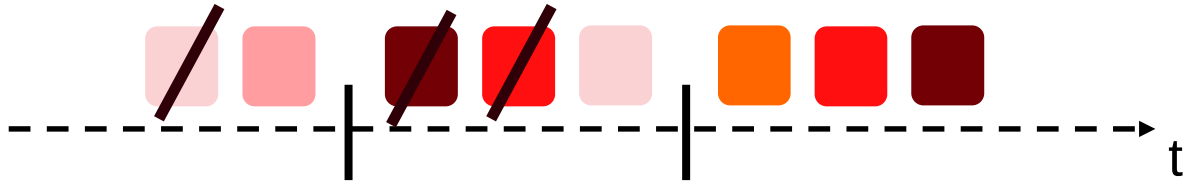
checkpoint interval = 5 s

- Нагрузка на CPU, I/O

Мониторинг checkpoint



Инкрементальные checkpoints



Full checkpoint

checkpoint_1

checkpoint_2

checkpoint_3

Incremental checkpoint

checkpoint_1

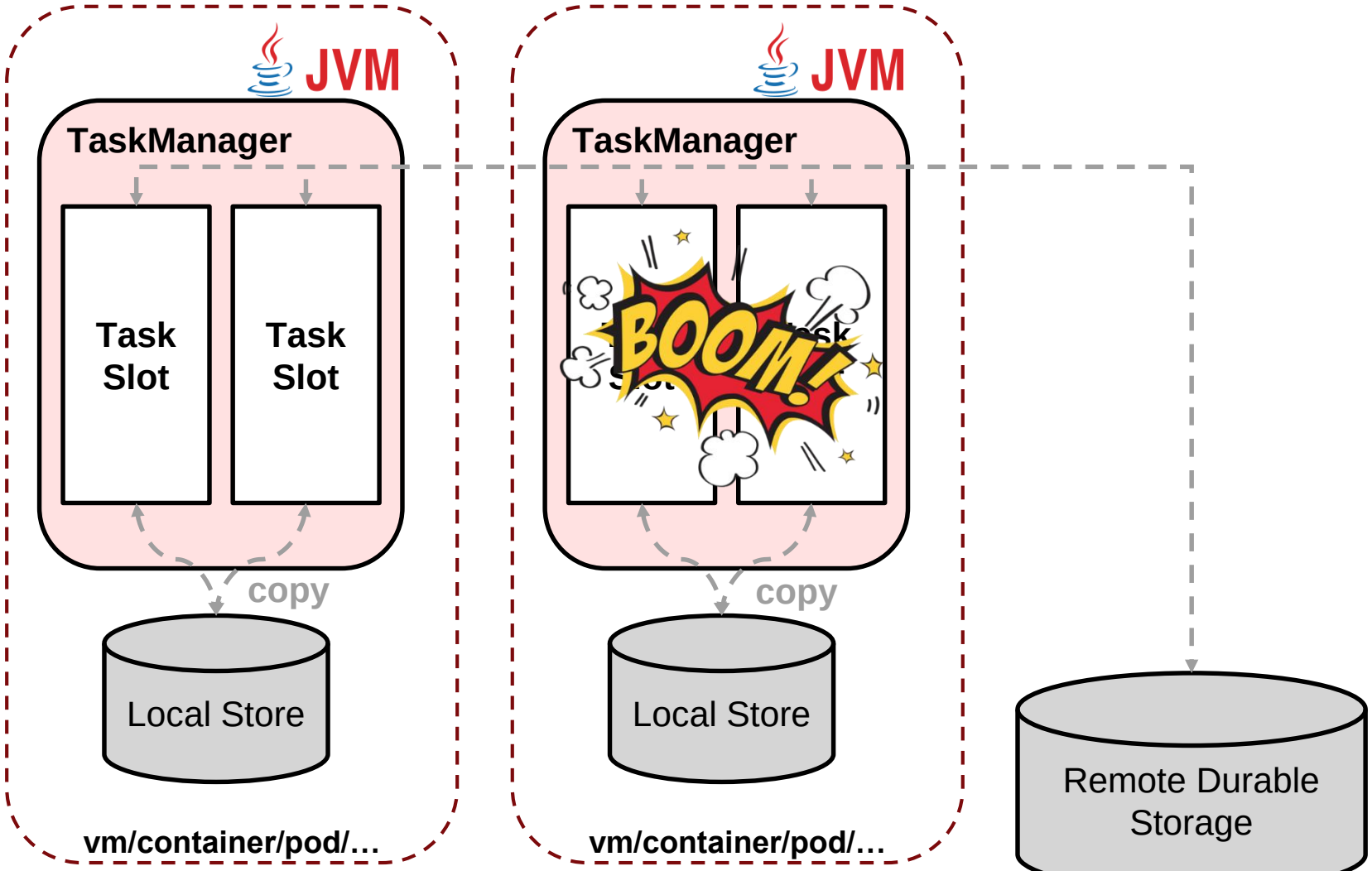
checkpoint_2

checkpoint_3

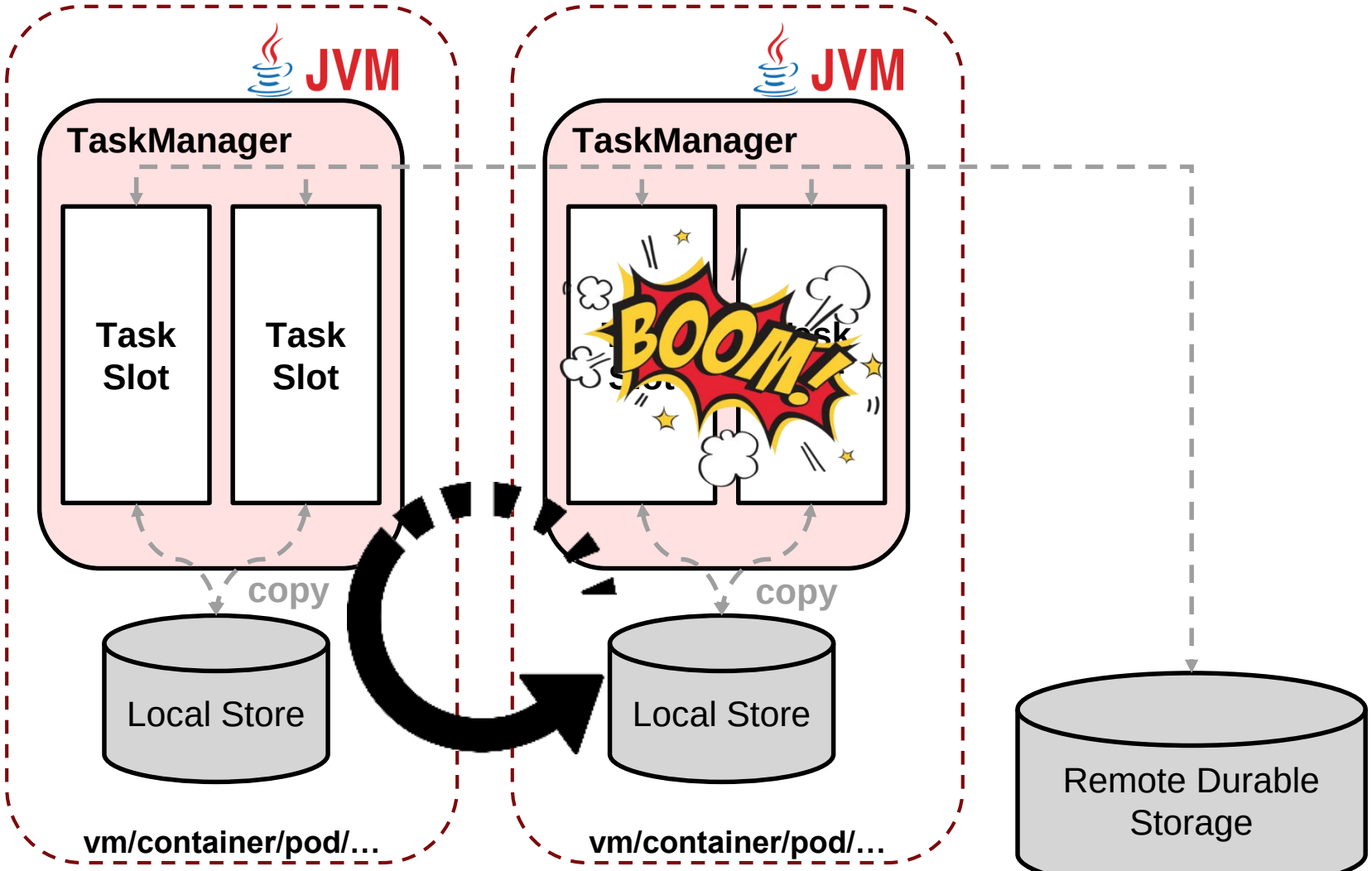
Инкрементальные checkpoints

https://youtu.be/Tn_uk5EDiv8?si=jmvxRRvKmpzUX6pC

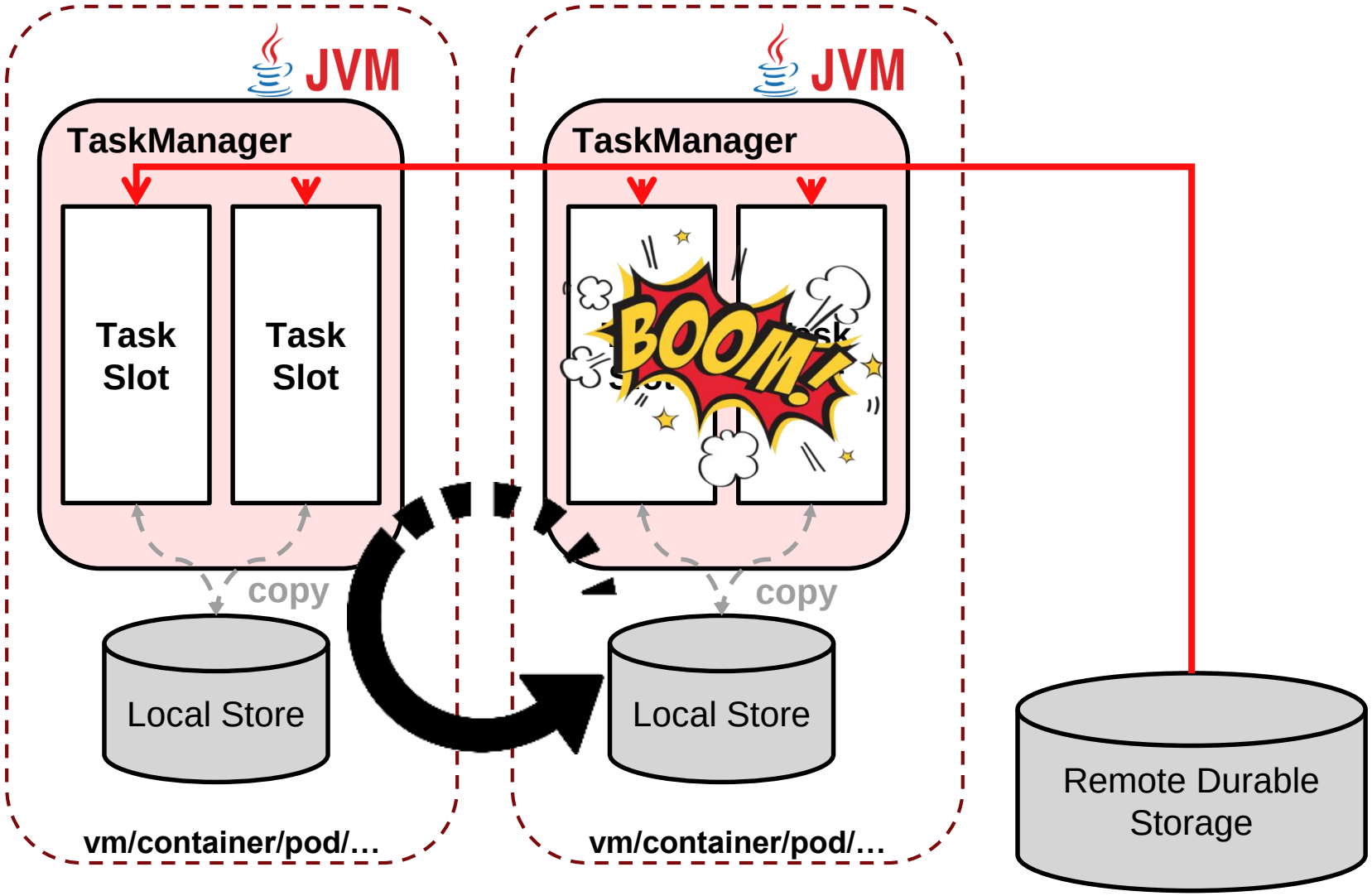
Локальное восстановление задачи



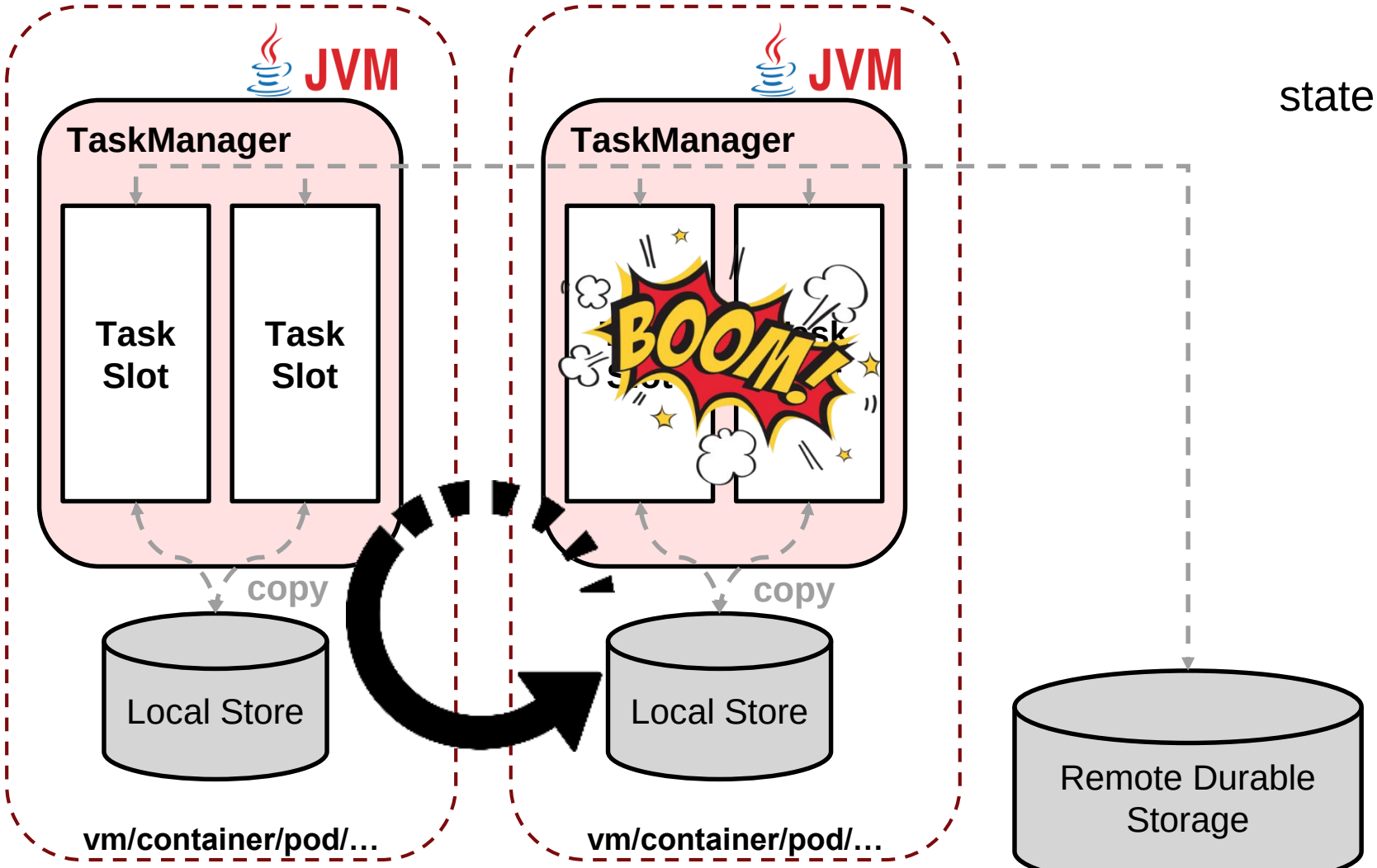
Локальное восстановление задачи



Локальное восстановление задачи

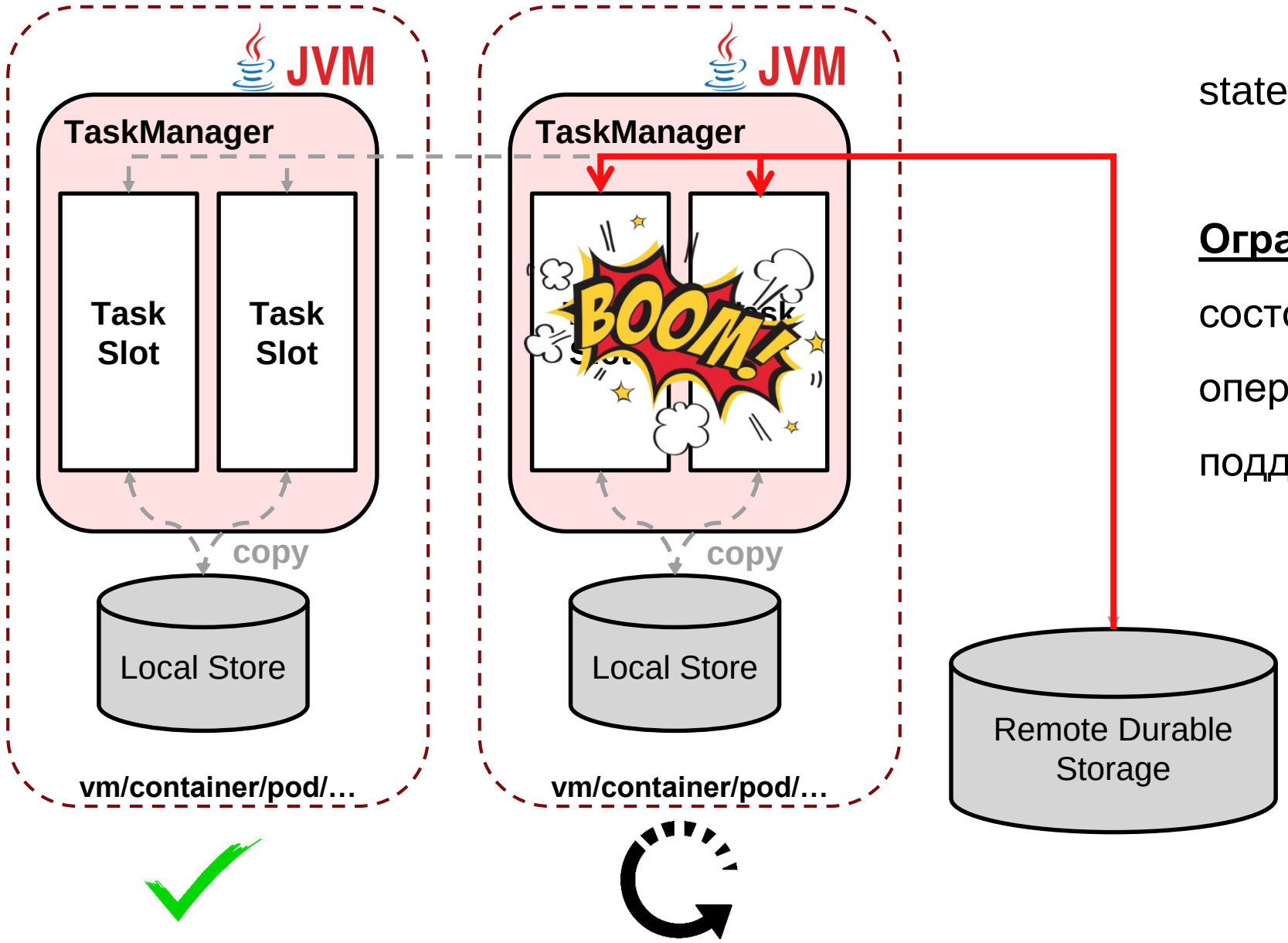


Локальное восстановление задачи



state.backend.local-recovery: **true**

Локальное восстановление задачи



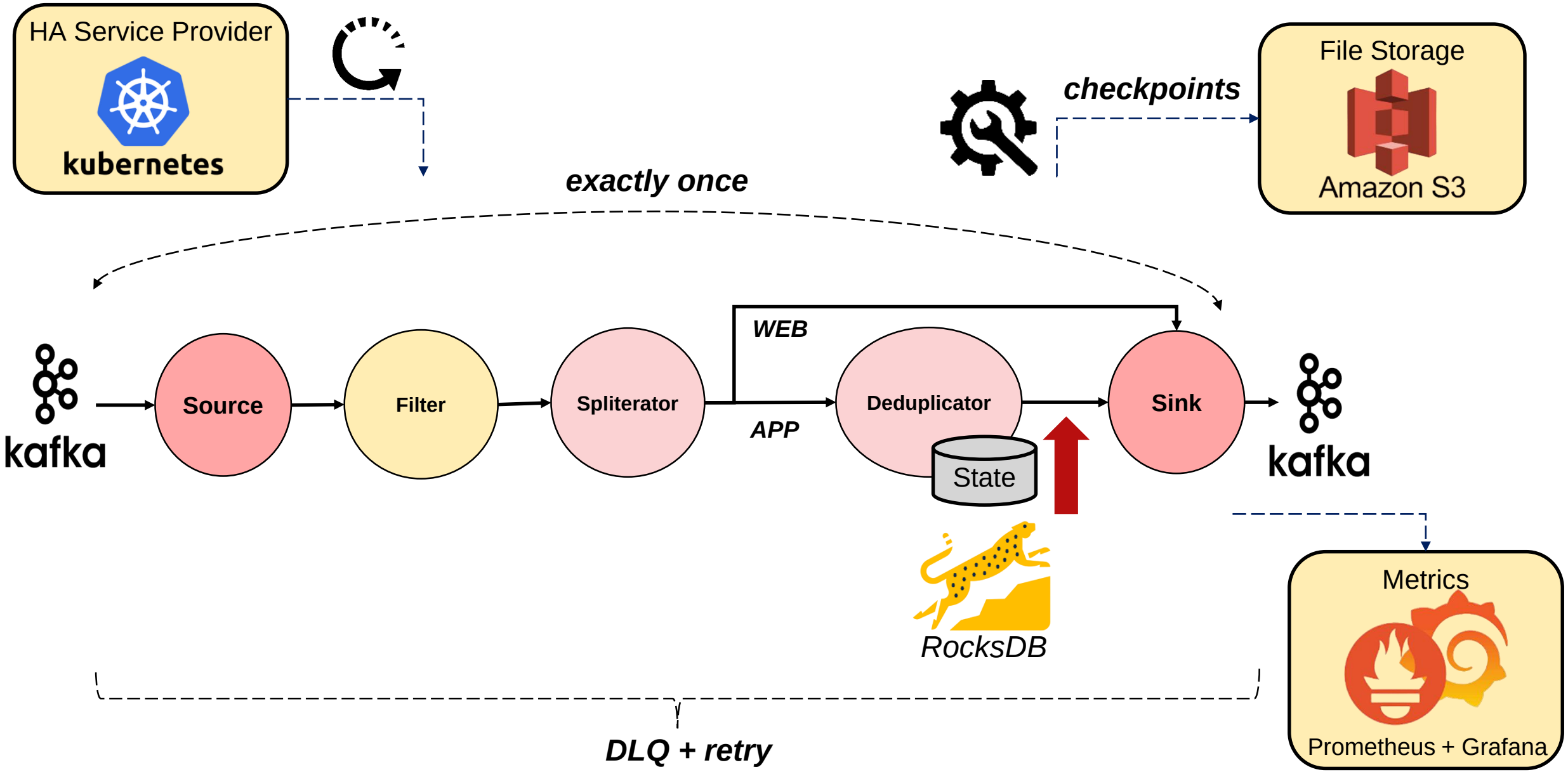
state.backend.local-recovery: **true**

Ограничение: поддержка только состояний с ключом. Состояния операторов и таймеры не поддерживаются

Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ Непрерывная работа Flink-задания
- ☑ Увеличение нагрузки. RocksDB
- ☑ **Работа с большим состоянием**
- ☐ Ускорение работы Flink-задания

Текущая реализация задания



Что может пойти не так?

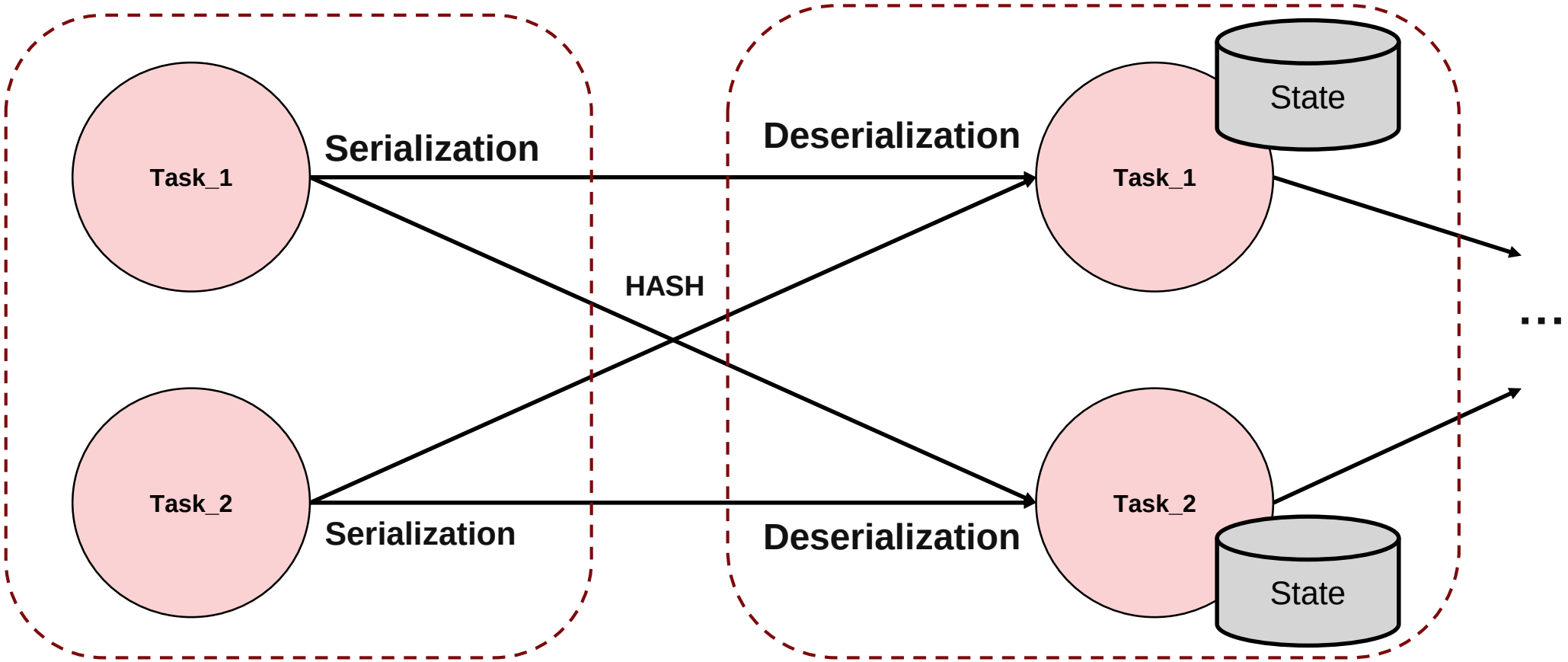
- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ Непрерывная работа Flink-задания
- ☑ Увеличение нагрузки. RocksDB
- ☑ Работа с большим состоянием
- ☐ **Ускорение работы Flink-задания**

Определение bottleneck. Изменение параллельности

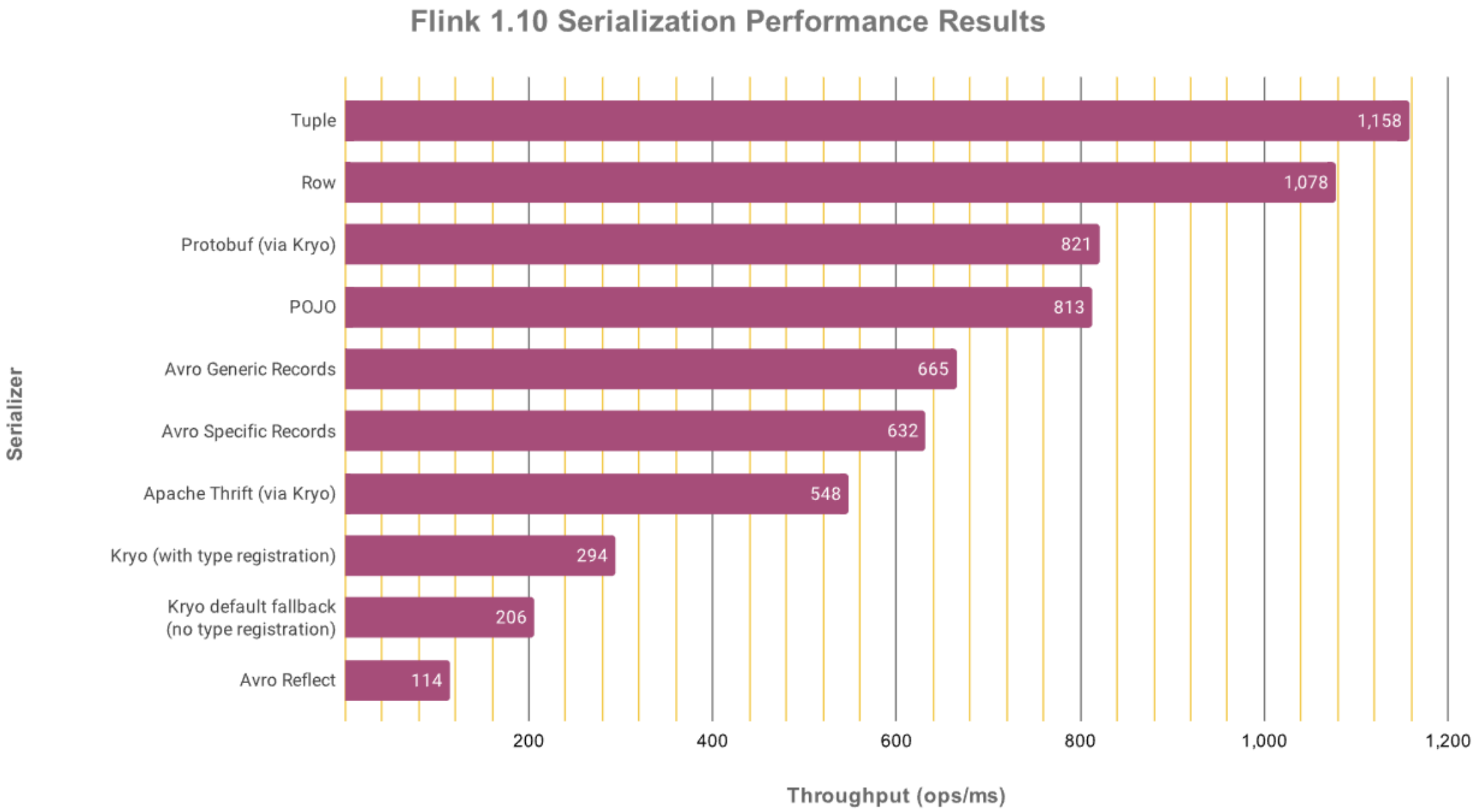
Overview | Exceptions | TimeLine | Checkpoints | Configuration

Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Duration	Tasks
Source: Custom Source	RUNNING	0 B	0	2.13 GB	48,873,775	1	2023-08-01 17:57:08	7m 1s	1
Filter	RUNNING	2.13 GB	48,866,741	2.13 GB	48,866,748	2	2023-08-01 17:57:08	7m 1s	2
metric_flat_map_id	RUNNING	2.13 GB	48,860,090	2.13 GB	48,860,090	2	2023-08-01 17:57:08	7m 1s	2
Sink: Writer	RUNNING	2.13 GB	48,853,175	0 B	0	2	2023-08-01 17:57:08	7m 1s	2
Sink: Committer	RUNNING	8 B	0	0 B	0	2	2023-08-01 17:57:08	7m 1s	2

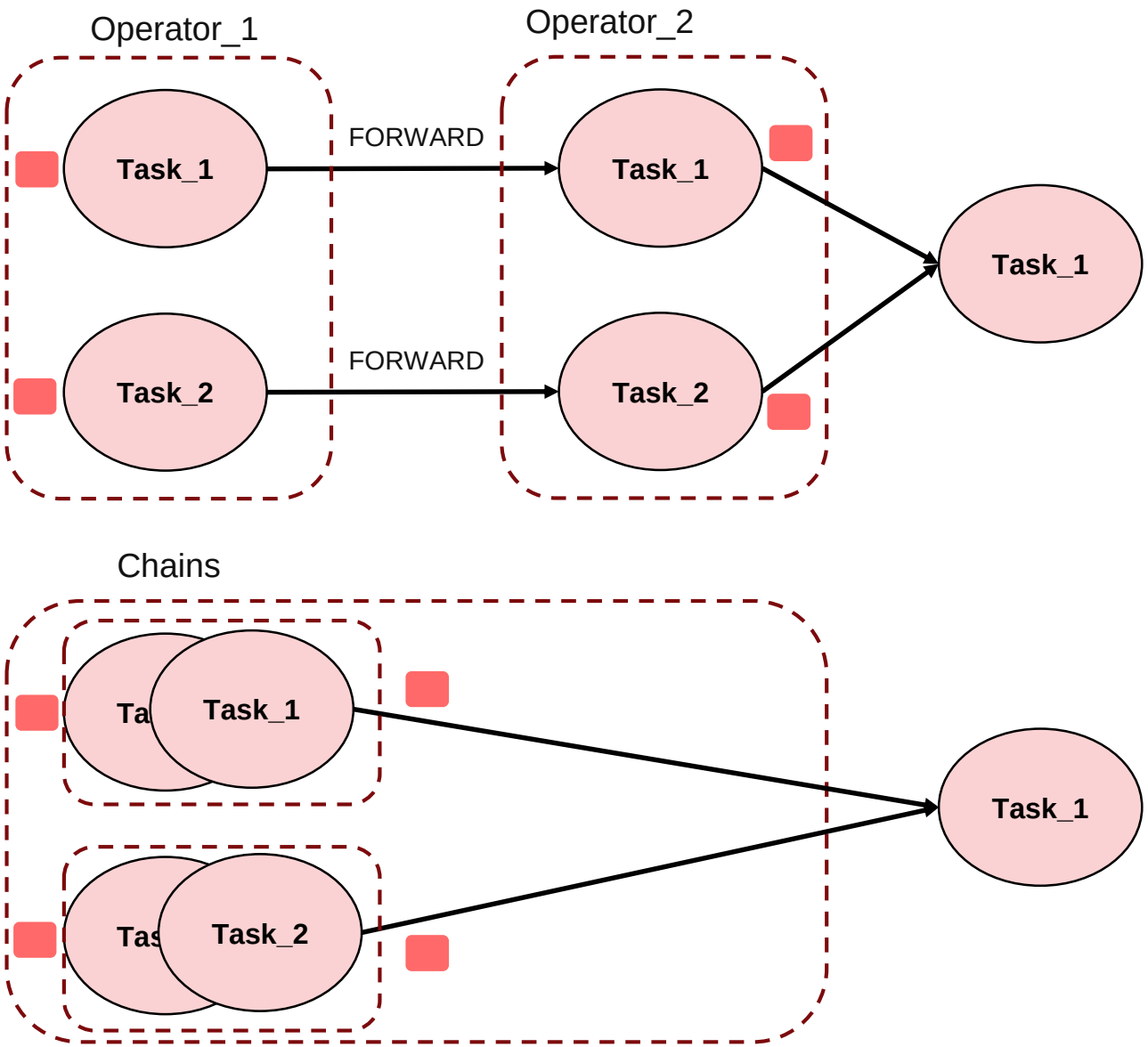
Ускорение сериализации



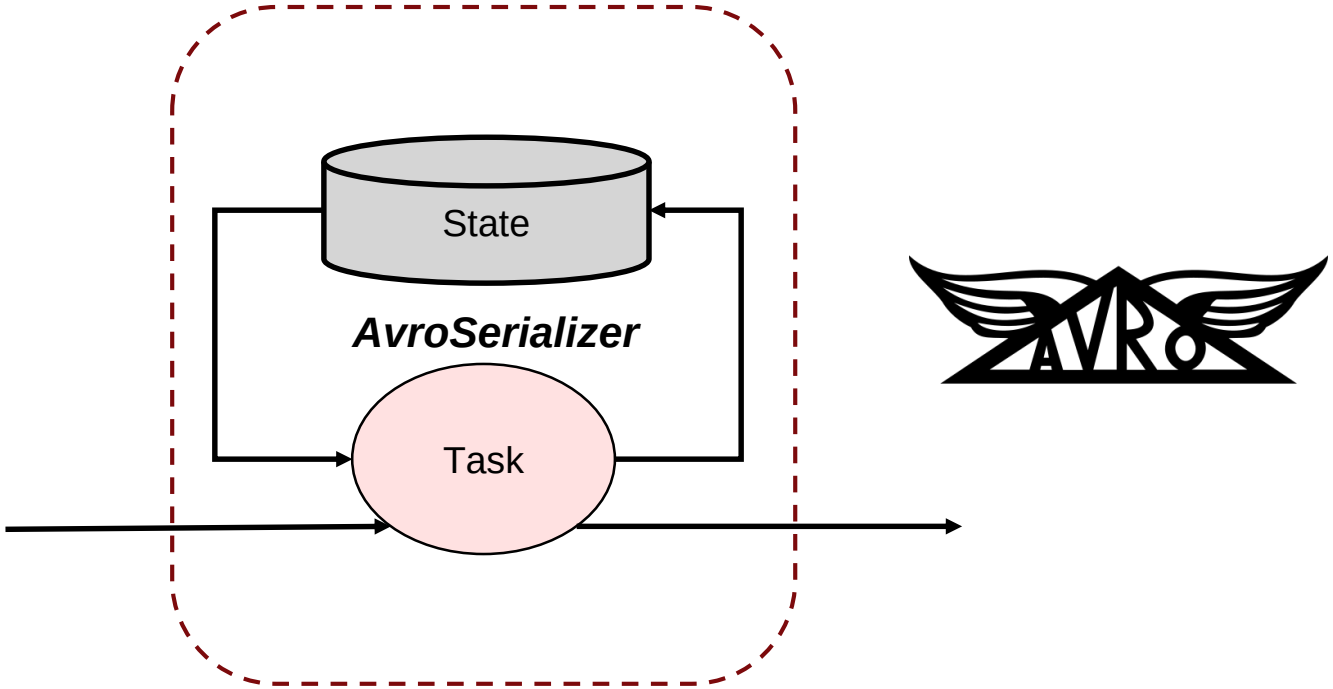
Ускорение сериализации



Оптимизация «Цепочки операторов»



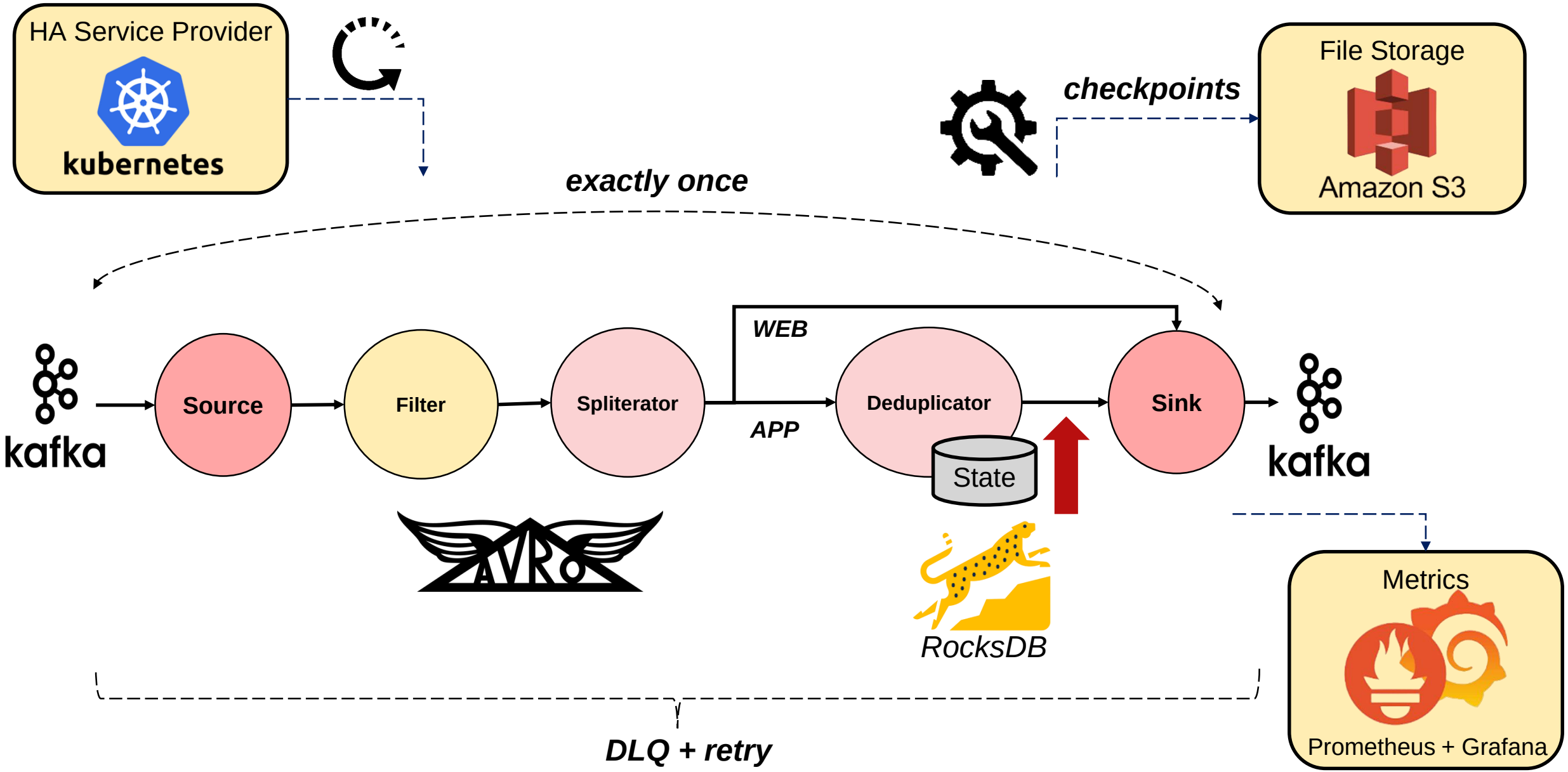
Сериализация объектов в состоянии



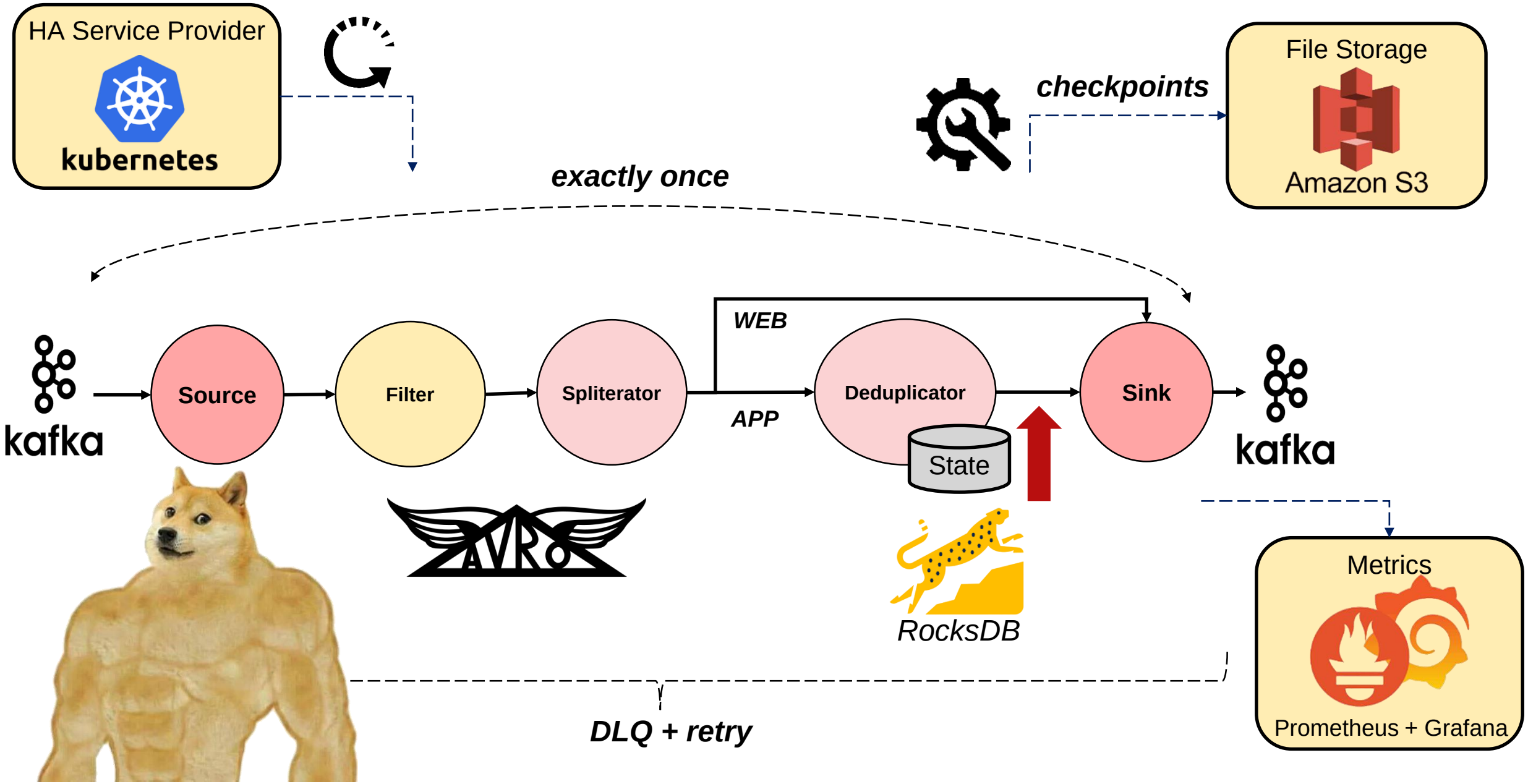
Что может пойти не так?

- ☑ Сбой кластера Flink
- ☑ Настройка exactly once
- ☑ Эволюция приложения
- ☑ Непрерывная работа Flink-задания
- ☑ Увеличение нагрузки. RocksDB
- ☑ Работа с большим состоянием
- ☑ **Ускорение работы Flink-задания**

Текущая реализация задания



Текущая реализация задания



Итоги

- Покрывайте приложение всеми доступными метриками
- Настройте HA для кластера
- Версии приложения должны быть совместимы с savepoint
- Настройте и оптимизируйте RocksDB под большое состояние

Спасибо!

Бобряков Александр

@appp_master