Drools ParDo and SCIO Dataflow: A Goodbye Microservices Tale

Alberto López

BEAM SUMMIT

September 4-5, 2024

Sunnyvale, CA. USA

Agenda

- Introduction
- The Old Tale: Cloudera and Openshift
- The Modern Tale: Dataflow, GKE, Memorystore/BigTable
- The New Tale: Dataflow, Dataflow, Dataflow
- Implementation: DroolsIO
- Conclusions and Future Work

About me

- From Madrid, Spain.
- Lived in Ireland and England.
- Working in Deutsche Bank; Technology, Data and Innovation (TDI) as Technical Leader in Madrid.
- Electronics and Telecommunications Engineer.
    - Started coding in C, C++, Java and Android, +14 years ago.
    - Ended up doing loads of Scala the last 6 years (Kafka, Spark) and Beam (last 2 years).
- Music lover!

Introduction

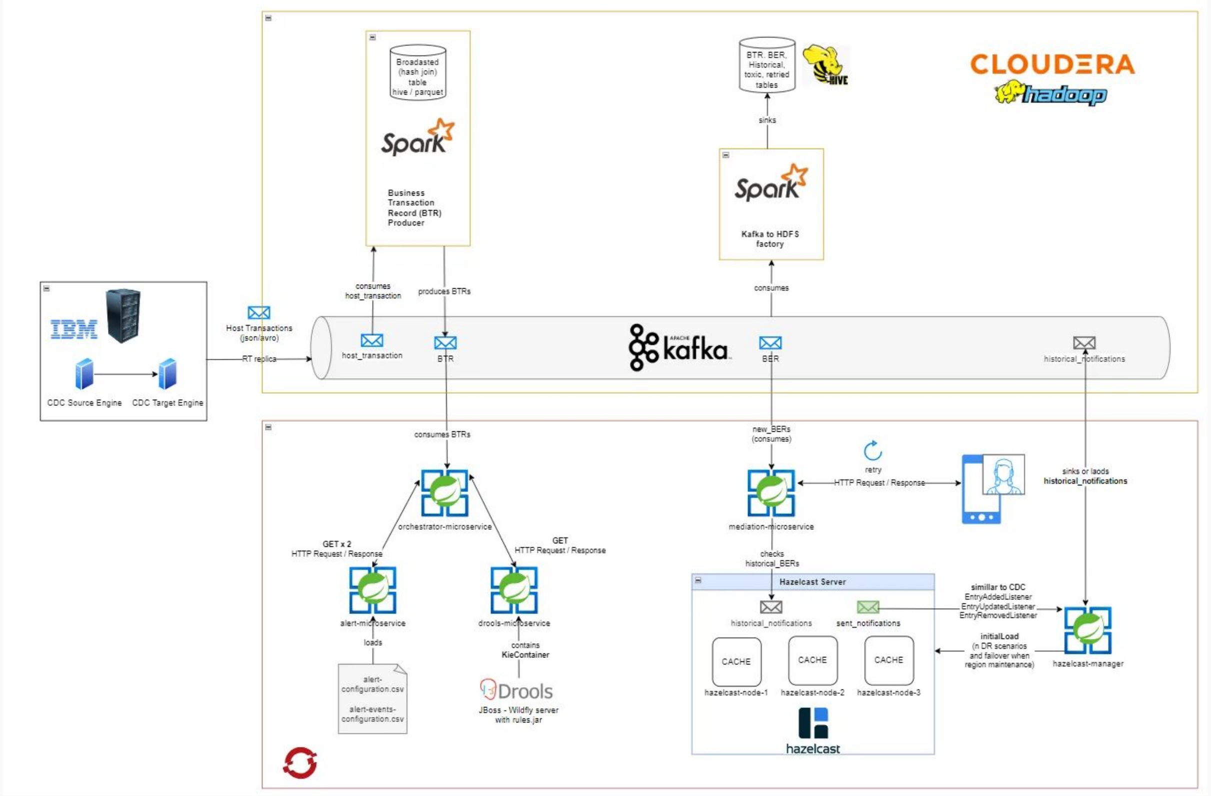
Introduction

Hello Dataflow! Sayonara microservices. Bye Spring with **Drools**. Ciao costly Hazelcast/Memorystore/BigTable…This is a success story about how “low level” engineering and architecture can beat high level architecture approaches:

- **Reducing costs massively.**
- **Time to market.**
- **Improving performance, efficiency and scalability.**
- **Simplifying flows and eliminating technical debt.**

The Old Tale: Cloudera and Openshift

The Old Tale: Cloudera and Openshift

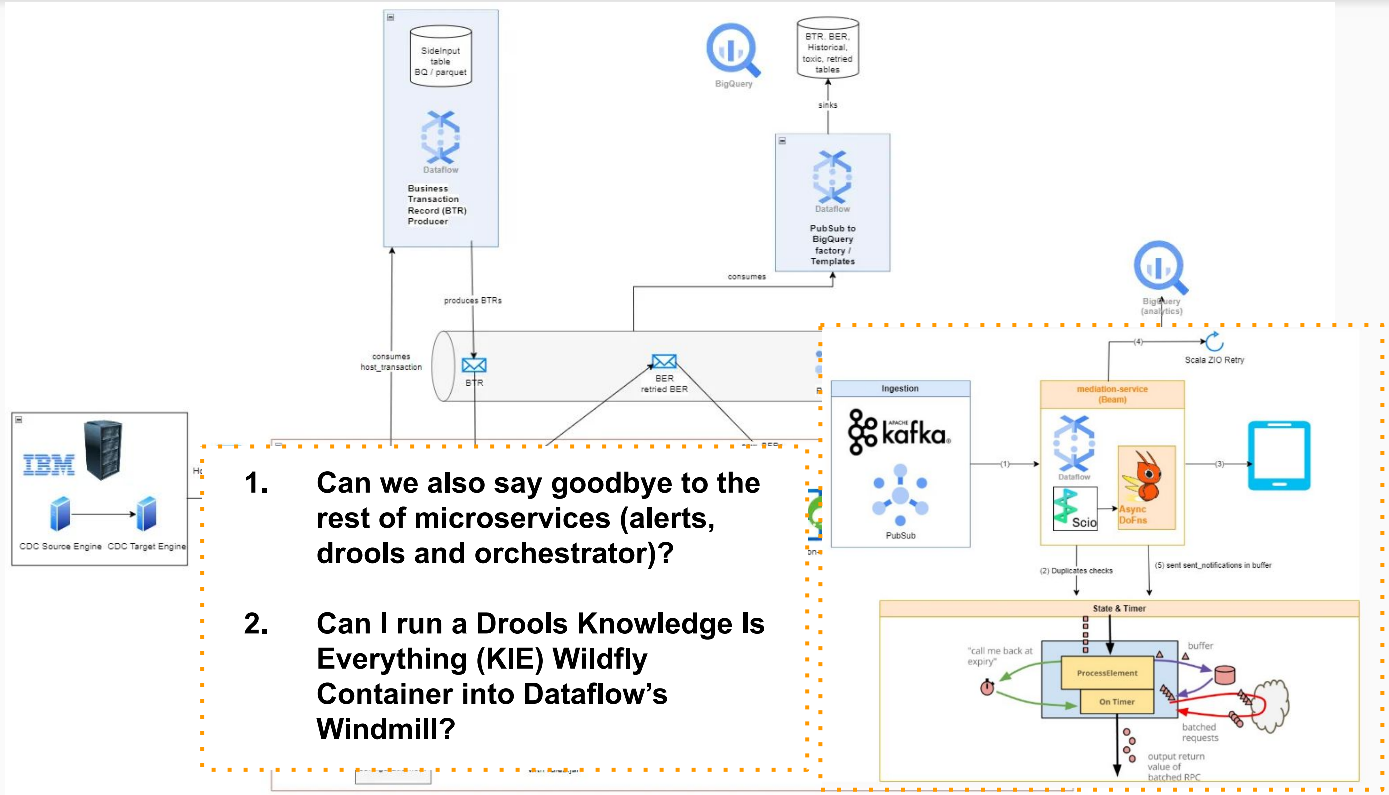


You have some containerised microservices (e.g: Spring on Openshift) that are being migrated into the cloud: “**Lift and Shift** them on GKE”.

But, you also have to migrate an In Memory Data Grid (IMDG) running on Openshift: “OK, pick Memorystore/BigTable, adapt your app and…**Lift and Shift** the rest on GKE”.

The Modern Tale: Dataflow, GKE, Memorystore/BigTable

The Modern Tale: Dataflow, GKE, Memorystore/BigTable



1. **Can we also say goodbye to the rest of microservices (alerts, drools and orchestrator)?**

2. **Can I run a Drools Knowledge Is Everything (KIE) Wildfly Container into Dataflow's Windmill?**

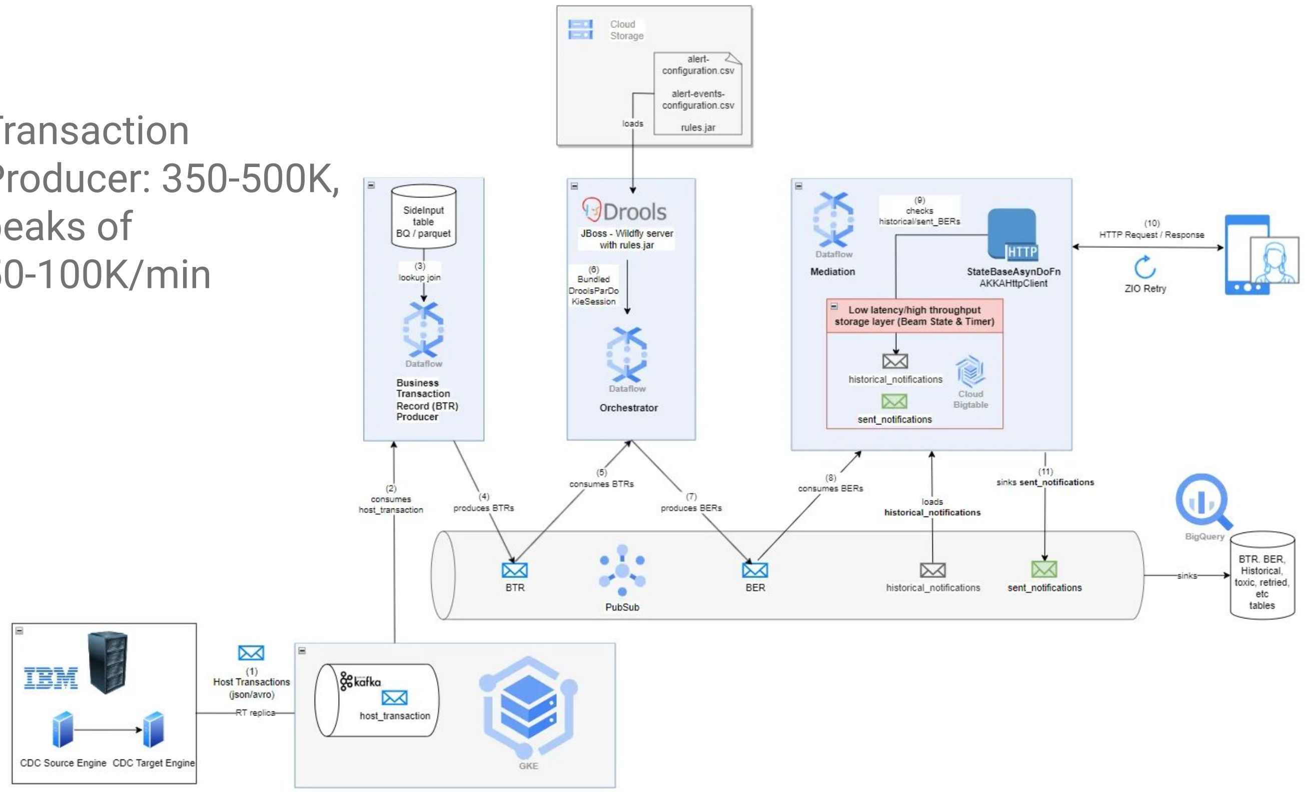
Check yesterday's talk on: https://medium.com/@serna.alberto.eng/avoid-http-requests-duplicates-in-apache-beam-with-scio-a-custom-baseasyncdofn-and-state-and-2c7d63059ab3

The New Tale: Dataflow, Dataflow, Dataflow

BEAM SUMMIT

The New Tale: Dataflow, Dataflow, Dataflow

Transaction Producer: 350-500K, peaks of 50-100K/min



Cloud Storage
alert-configuration.csv
alert-events-configuration.csv
rules.jar
loads
SideInput table BQ / parquet
(3) lookup join
Dataflow
Business Transaction Record (BTR) Producer
Drools
JBoss - Wildfly server with rules.jar
(6) Bundled DroolsParDo KieSession
Dataflow
Orchestrator
Dataflow
Mediation
(9) checks historical/sent_BERs
HTTP
StateBaseAsynDoFn AKKAHttpClient
Low latency/high throughput storage layer (Beam State & Timer)
historical_notifications
sent_notifications
Cloud Bigtable
(10) HTTP Request / Response
ZIO Retry
(2) consumes host_transaction
(4) produces BTRs
(5) consumes BTRs
(7) produces BERs
(8) consumes BERs
loads historical_notifications
(11) sinks sent_notifications
BTR
PubSub
BER
historical_notifications
sent_notifications
BigQuery
sinks
BTR, BER, Historical, toxic, retried, etc tables
IBM
CDC Source Engine
CDC Target Engine
(1) Host Transactions (json/avro)
RT replica
kafka
host_transaction
GKE

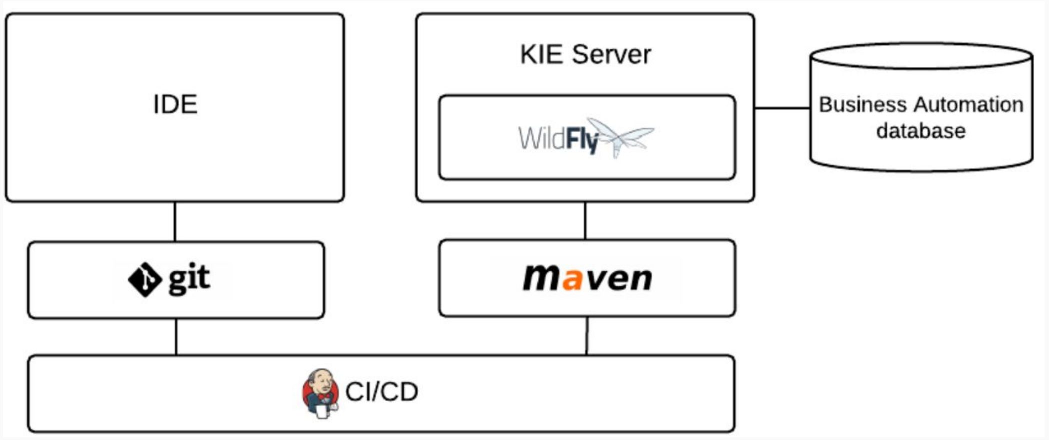
LATENCIES:

- Cdc to Kafka ~**1.5 - 2s**
- Dataflow (BTR) to Dataflow (Orchestrator) to Dataflow (mediation) **~1s**
- Notification HUB **~0.6 - 1 s**

BEAM SUMMIT

Implementation: DroolsIO

Implementation: DroolsIO - what's Drools?



"**Drools** is a business-rule management system with a forward-chaining and backward-chaining inference-based rules engine, allowing fast and reliable evaluation of business rules and complex event processing.

A rules engine is also a fundamental building block to create an expert system which, in **artificial intelligence**, is a computer system that emulates the decision-making ability of a human expert."

https://docs.drools.org/7.58.0.Final/drools-docs/html_single/#decision-engine-con_decision-engine

Implementation: DroolsIO

https://www.baeldung.com/drools

*“**Facts*** – represents data that serves as input for rules

Working Memory – a storage with *Facts,* where they are used for pattern matching and can be modified, inserted and removed

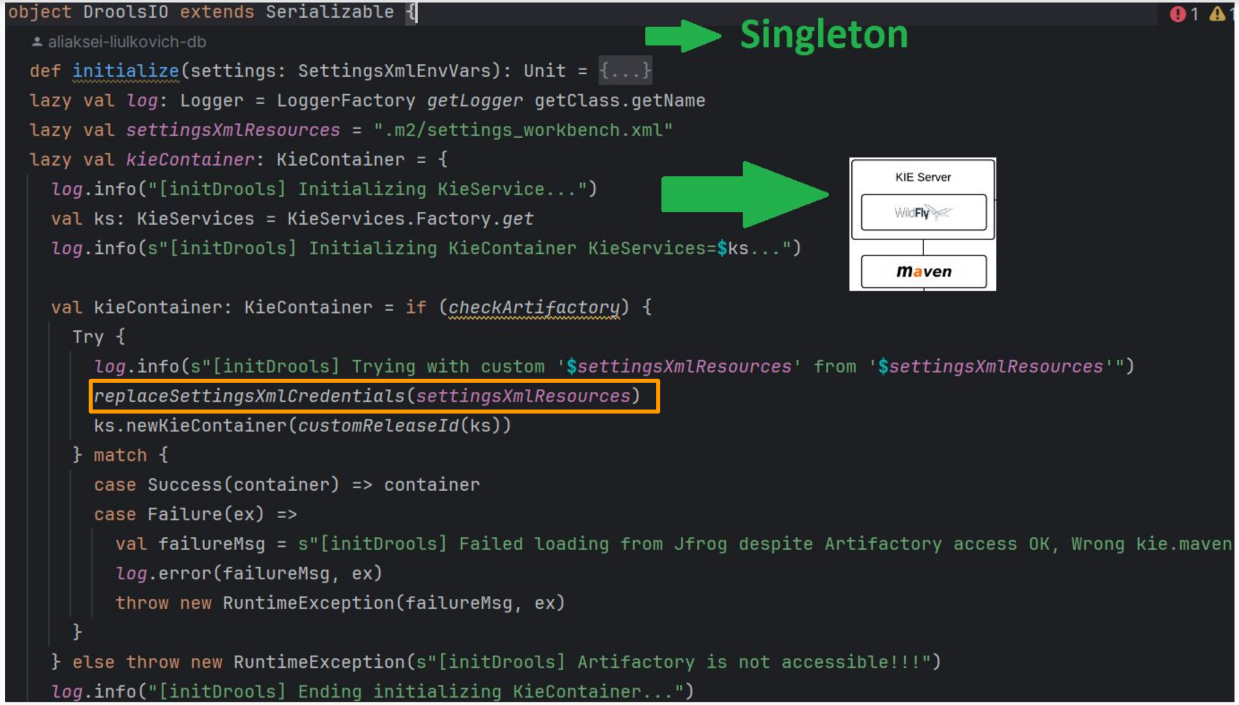
Rule – represents a single rule which associates *Facts* with matching actions. It can be written in Drools Rule Language in the ***.drl* files** or as *Decision Table* in an excel spreadsheet

Knowledge Session – it holds all the resources required for firing rules; all *Facts* are inserted into session, and then matching rules are fired

Knowledge Base – represents the knowledge in the Drools ecosystem, it has the information about the resources where *Rules* are found, and also it creates the *Knowledge Session*

Module – A module holds multiple Knowledge Bases which can hold different sessions”

Implementation: DroolsIO - KieContainer



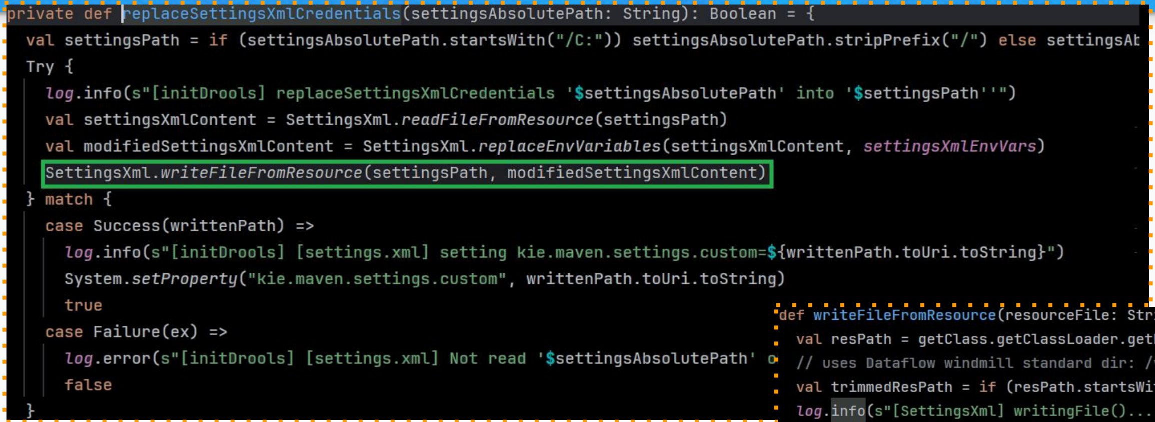
```
object DroolsIO extends Serializable {
  def initialize(settings: SettingsXmlEnvVars): Unit = {...}
  lazy val log: Logger = LoggerFactory getLogger getClass.getName
  lazy val settingsXmlResources = ".m2/settings_workbench.xml"
  lazy val kieContainer: KieContainer = {
    log.info("[initDrools] Initializing KieService...")
    val ks: KieServices = KieServices.Factory.get
    log.info(s"[initDrools] Initializing KieContainer KieServices=$ks...")

    val kieContainer: KieContainer = if (checkArtifactory) {
      Try {
        log.info(s"[initDrools] Trying with custom '$settingsXmlResources' from '$settingsXmlResources'")
        replaceSettingsXmlCredentials(settingsXmlResources)
        ks.newKieContainer(customReleaseId(ks))
      } match {
        case Success(container) => container
        case Failure(ex) =>
          val failureMsg = s"[initDrools] Failed loading from Jfrog despite Artifactory access OK, Wrong kie.maven
          log.error(failureMsg, ex)
          throw new RuntimeException(failureMsg, ex)
      }
    } else throw new RuntimeException(s"[initDrools] Artifactory is not accessible!!!")
    log.info("[initDrools] Ending initializing KieContainer...")
```

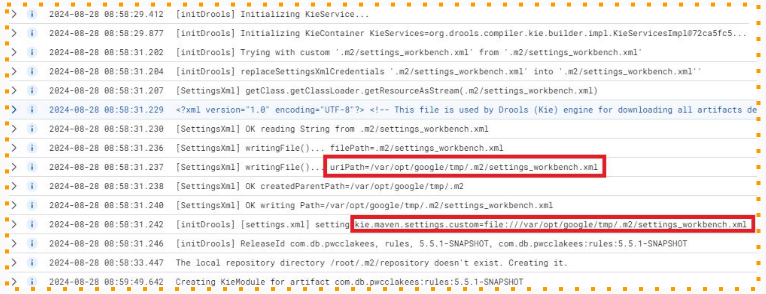
KIE Server
WildFly
maven

- Singleton instance per Worker.
- New KieContainer per Worker.
- Creation Time ~30s (depending on downloading .jars for rules.jar dependencies).

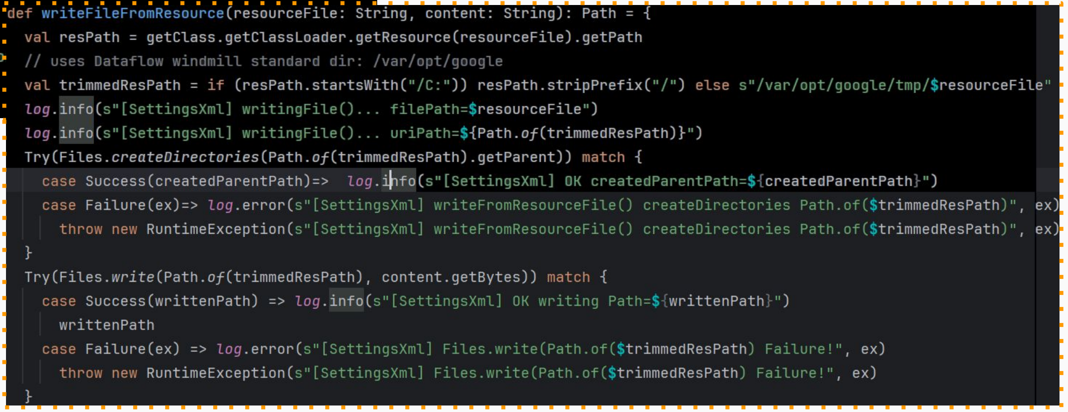
Implementation: DroolsIO



```
private def replaceSettingsXmlCredentials(settingsAbsolutePath: String): Boolean = {
  val settingsPath = if (settingsAbsolutePath.startsWith("/C:")) settingsAbsolutePath.stripPrefix("/") else settingsAb
  Try {
    log.info(s"[initDrools] replaceSettingsXmlCredentials '$settingsAbsolutePath' into '$settingsPath''")
    val settingsXmlContent = SettingsXml.readFileFromResource(settingsPath)
    val modifiedSettingsXmlContent = SettingsXml.replaceEnvVariables(settingsXmlContent, settingsXmlEnvVars)
    SettingsXml.writeFileFromResource(settingsPath, modifiedSettingsXmlContent)
  } match {
    case Success(writtenPath) =>
      log.info(s"[initDrools] [settings.xml] setting kie.maven.settings.custom=${writtenPath.toUri.toString}")
      System.setProperty("kie.maven.settings.custom", writtenPath.toUri.toString)
      true
    case Failure(ex) =>
      log.error(s"[initDrools] [settings.xml] Not read '$settingsAbsolutePath' o
      false
  }
```

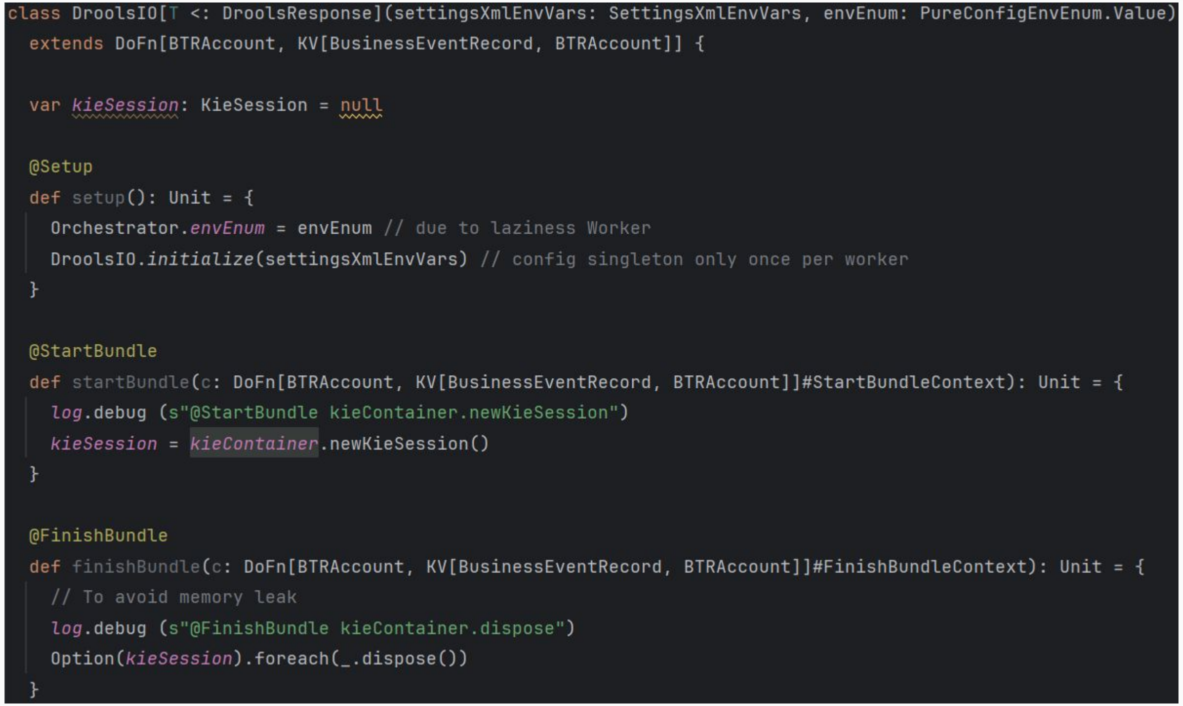


```
2024-08-28 08:58:29.412  [initDrools] Initializing KieService...
2024-08-28 08:58:29.877  [initDrools] Initializing KieContainer KieServices=org.drools.compiler.kie.builder.impl.KieServicesImpl@72ca5fc5...
2024-08-28 08:58:31.202  [initDrools] Trying with custom '.m2/settings_workbench.xml' from '.m2/settings_workbench.xml'
2024-08-28 08:58:31.204  [initDrools] replaceSettingsXmlCredentials '.m2/settings_workbench.xml' into '.m2/settings_workbench.xml''
2024-08-28 08:58:31.207  [SettingsXml] getClass.getClassLoader.getResourceAsStream(.m2/settings_workbench.xml)
2024-08-28 08:58:31.229  <?xml version="1.0" encoding="UTF-8"?> <!-- This file is used by Drools (Kie) engine for downloading all artifacts de
2024-08-28 08:58:31.230  [SettingsXml] OK reading String from .m2/settings_workbench.xml
2024-08-28 08:58:31.236  [SettingsXml] writingFile()... filePath=.m2/settings_workbench.xml
2024-08-28 08:58:31.237  [SettingsXml] writingFile()... uriPath=/var/opt/google/tmp/.m2/settings_workbench.xml
2024-08-28 08:58:31.238  [SettingsXml] OK createdParentPath=/var/opt/google/tmp/.m2
2024-08-28 08:58:31.240  [SettingsXml] OK writing Path=/var/opt/google/tmp/.m2/settings_workbench.xml
2024-08-28 08:58:31.242  [initDrools] [settings.xml] setting kie.maven.settings.custom=file:///var/opt/google/tmp/.m2/settings_workbench.xml
2024-08-28 08:58:31.246  [initDrools] ReleaseId com.db.pwcclakees, rules, 5.5.1-SNAPSHOT, com.db.pwcclakees:rules:5.5.1-SNAPSHOT
2024-08-28 08:58:33.447  The local repository directory /root/.m2/repository doesn't exist. Creating it.
2024-08-28 08:59:49.642  Creating KieModule for artifact com.db.pwcclakees:rules:5.5.1-SNAPSHOT
```



```
def writeFileFromResource(resourceFile: String, content: String): Path = {
  val resPath = getClass.getClassLoader.getResource(resourceFile).getPath
  // uses Dataflow windmill standard dir: /var/opt/google
  val trimmedResPath = if (resPath.startsWith("/C:")) resPath.stripPrefix("/") else s"/var/opt/google/tmp/$resourceFile"
  log.info(s"[SettingsXml] writingFile()... filePath=$resourceFile")
  log.info(s"[SettingsXml] writingFile()... uriPath=${Path.of(trimmedResPath)}")
  Try(Files.createDirectories(Path.of(trimmedResPath).getParent)) match {
    case Success(createdParentPath)=>  log.info(s"[SettingsXml] OK createdParentPath=${createdParentPath}")
    case Failure(ex)=> log.error(s"[SettingsXml] writeFromResourceFile() createDirectories Path.of($trimmedResPath)", ex)
      throw new RuntimeException(s"[SettingsXml] writeFromResourceFile() createDirectories Path.of($trimmedResPath)", ex)
  }
  Try(Files.write(Path.of(trimmedResPath), content.getBytes)) match {
    case Success(writtenPath) => log.info(s"[SettingsXml] OK writing Path=${writtenPath}")
      writtenPath
    case Failure(ex) => log.error(s"[SettingsXml] Files.write(Path.of($trimmedResPath) Failure!", ex)
      throw new RuntimeException(s"[SettingsXml] Files.write(Path.of($trimmedResPath) Failure!", ex)
  }
```

Implementation: DroolsIO - ParDo



```
class DroolsIO[T <: DroolsResponse](settingsXmlEnvVars: SettingsXmlEnvVars, envEnum: PureConfigEnvEnum.Value)
  extends DoFn[BTRAccount, KV[BusinessEventRecord, BTRAccount]] {

  var kieSession: KieSession = null

  @Setup
  def setup(): Unit = {
    Orchestrator.envEnum = envEnum // due to laziness Worker
    DroolsIO.initialize(settingsXmlEnvVars) // config singleton only once per worker
  }

  @StartBundle
  def startBundle(c: DoFn[BTRAccount, KV[BusinessEventRecord, BTRAccount]]#StartBundleContext): Unit = {
    log.debug (s"@StartBundle kieContainer.newKieSession")
    kieSession = kieContainer.newKieSession()
  }

  @FinishBundle
  def finishBundle(c: DoFn[BTRAccount, KV[BusinessEventRecord, BTRAccount]]#FinishBundleContext): Unit = {
    // To avoid memory leak
    log.debug (s"@FinishBundle kieContainer.dispose")
    Option(kieSession).foreach(_.dispose())
  }
```

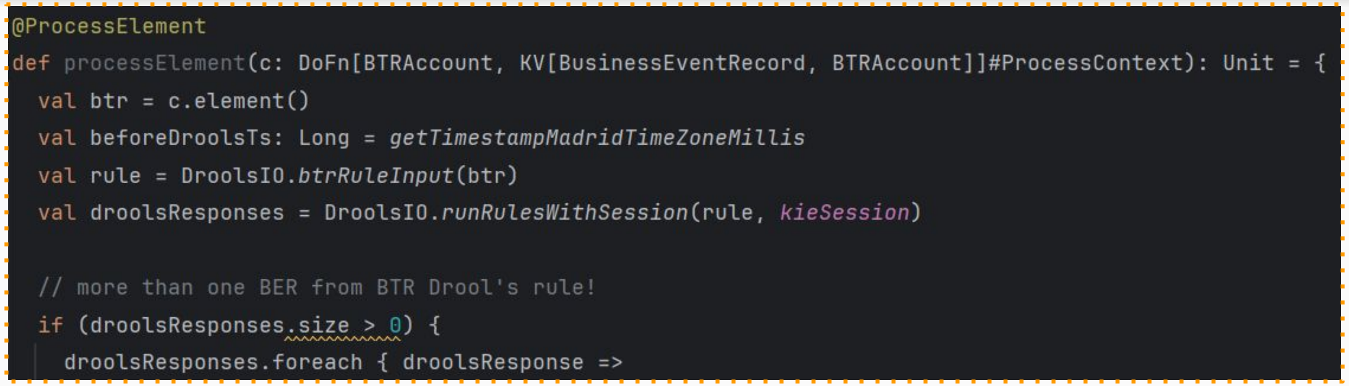
Performance Tip, keep your kieSessions per bundle!

@Setup

@StartBundle

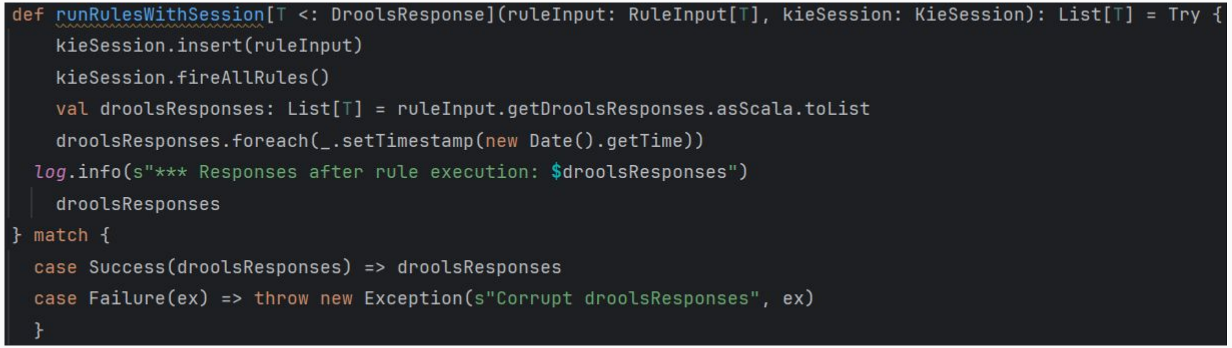
@FinishBundle

Implementation: DroolsIO - ParDo



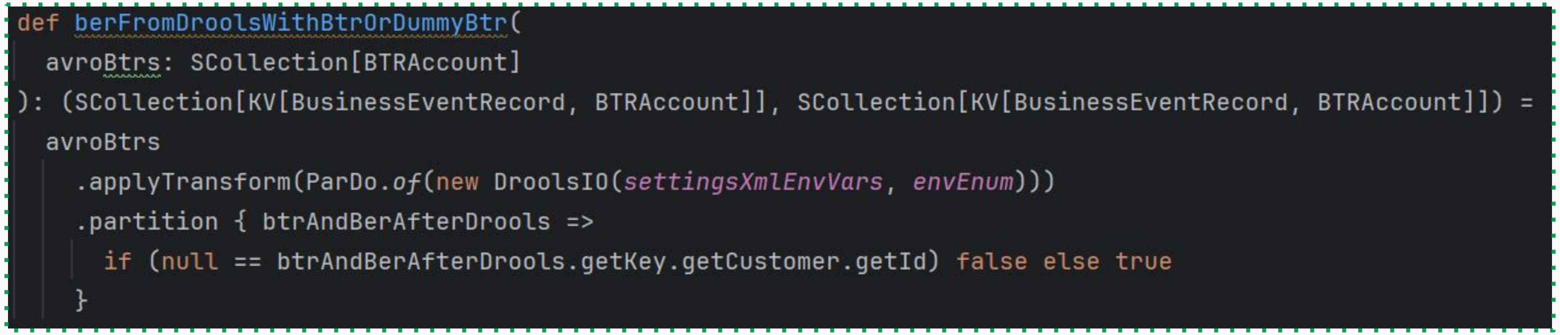
```
@ProcessElement
def processElement(c: DoFn[BTRAccount, KV[BusinessEventRecord, BTRAccount]]#ProcessContext): Unit = {
  val btr = c.element()
  val beforeDroolsTs: Long = getTimestampMadridTimeZoneMillis
  val rule = DroolsIO.btrRuleInput(btr)
  val droolsResponses = DroolsIO.runRulesWithSession(rule, kieSession)

  // more than one BER from BTR Drool's rule!
  if (droolsResponses.size > 0) {
    droolsResponses.foreach { droolsResponse =>
```



```
def runRulesWithSession[T <: DroolsResponse](ruleInput: RuleInput[T], kieSession: KieSession): List[T] = Try {
    kieSession.insert(ruleInput)
    kieSession.fireAllRules()
    val droolsResponses: List[T] = ruleInput.getDroolsResponses.asScala.toList
    droolsResponses.foreach(_.setTimestamp(new Date().getTime))
  log.info(s"*** Responses after rule execution: $droolsResponses")
    droolsResponses
} match {
  case Success(droolsResponses) => droolsResponses
  case Failure(ex) => throw new Exception(s"Corrupt droolsResponses", ex)
  }
```

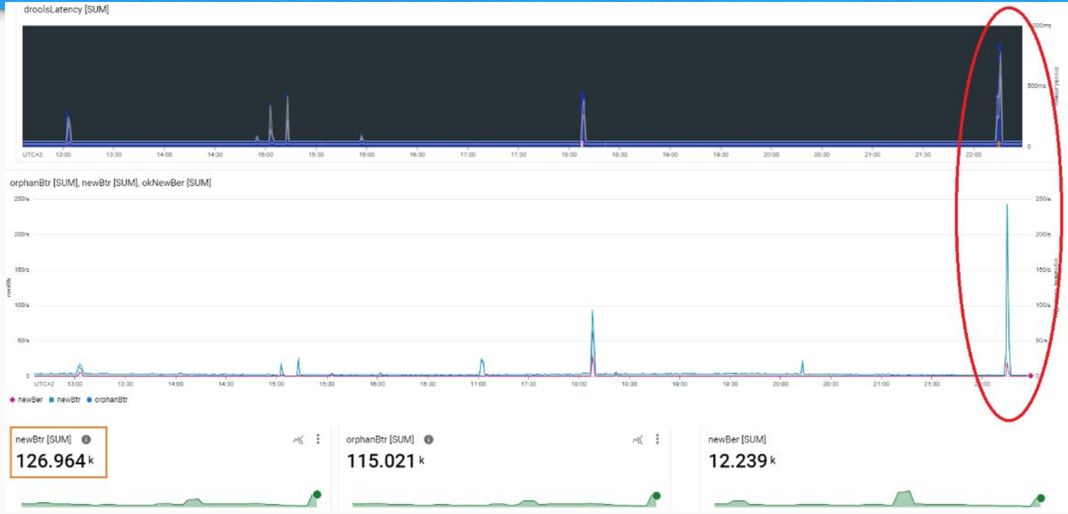
Implementation: DroolsIO - apply ParDo



```
def berFromDroolsWithBtrOrDummyBtr(
  avroBtrs: SCollection[BTRAccount]
): (SCollection[KV[BusinessEventRecord, BTRAccount]], SCollection[KV[BusinessEventRecord, BTRAccount]]) =
  avroBtrs
    .applyTransform(ParDo.of(new DroolsIO(settingsXmlEnvVars, envEnum)))
    .partition { btrAndBerAfterDrools =>
      if (null == btrAndBerAfterDrools.getKey.getCustomer.getId) false else true
    }
```

Conclusions and Future Work

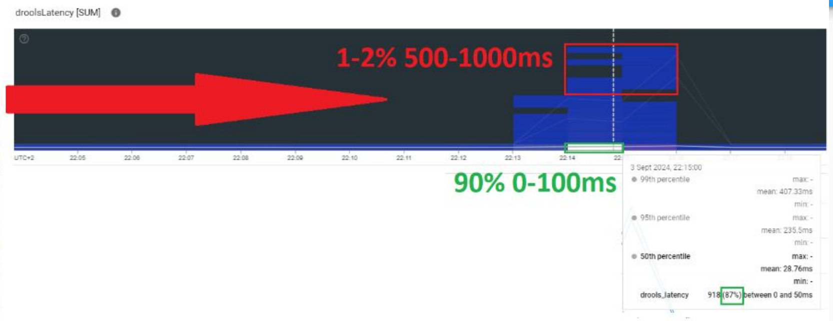
Conclusions



newBtr [SUM] 126.964 k

orphanBtr [SUM] 115.021 k

newBer [SUM] 12.239 k



3 Sept 2024, 22:15:00

99th percentile	max: -
	mean: 407.33ms
	min: -
95th percentile	max: -
	mean: 235.5ms
	min: -
50th percentile	max: -
	mean: 28.76ms
	min: -
drools_latency	918 (87%) between 0 and 50ms

Latency issues, play around with:
- numberOfWorkerHarnessThreads
- machine types
- profiling

???

Conclusions

1. Was it the quickest? **Yes**, time to market was totally won by the Re-engineering approach, it actually was +75% faster than Lifting and Shifting and adapting the Re-Architecting.

2. Was it the cheapest? **Yes**, there's not even a battle here, as we are getting rid off expensive infra on GCP, such as: GKE and BigTable/Memorystore (saving dozens of K€ / year).

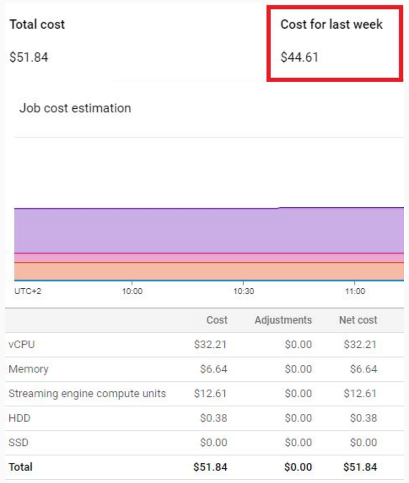
3. Is it the most maintainable? **Yes**, operational and development costs ($$) were dramatically reduced by saying goodbye to: GKE operations, CICD pipelines Releases, application complexity, (orchestrator, alerts, events, mediation, hazelcast-manager).

4. Was it the most efficient/performant? **Yes**!
   a. **lower latencies** with the embedded KIEContainer in the Orchestrator and the S & T pattern in the Mediation with new *StateBaseAsyncDoFn*).
   b. **Improved scalability**.
   c. Goodbye REST API calls and JSON everywhere! Hello **AVRO**!

5. Was it the best way to expose the notifications to analytics? Yes, easy integration with Pub/Sub and BQ!

BEAM SUMMIT

Conclusions

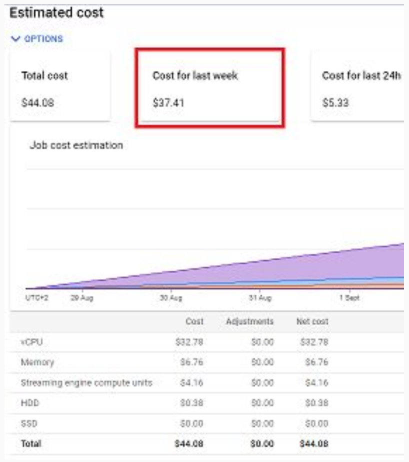


Total cost	Cost for last week
$51.84	$44.61

Job cost estimation

UTC+2 10:00 10:30 11:00

	Cost	Adjustments	Net cost
vCPU	$32.21	$0.00	$32.21
Memory	$6.64	$0.00	$6.64
Streaming engine compute units	$12.61	$0.00	$12.61
HDD	$0.38	$0.00	$0.38
SSD	$0.00	$0.00	$0.00
Total	**$51.84**	**$0.00**	**$51.84**



Estimated cost

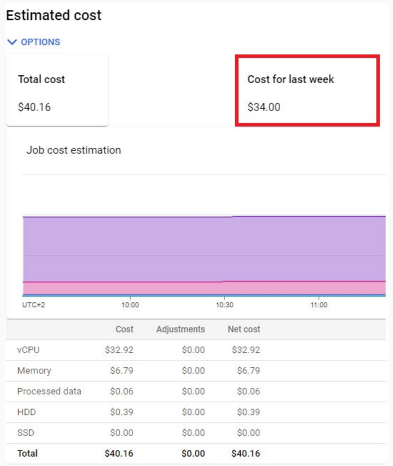
OPTIONS

Total cost	Cost for last week	Cost for last 24h
$44.08	$37.41	$5.33

Job cost estimation

UTC+2 29 Aug 30 Aug 31 Aug 1 Sept

	Cost	Adjustments	Net cost
vCPU	$32.78	$0.00	$32.78
Memory	$6.76	$0.00	$6.76
Streaming engine compute units	$4.16	$0.00	$4.16
HDD	$0.38	$0.00	$0.38
SSD	$0.00	$0.00	$0.00
Total	**$44.08**	**$0.00**	**$44.08**



Estimated cost

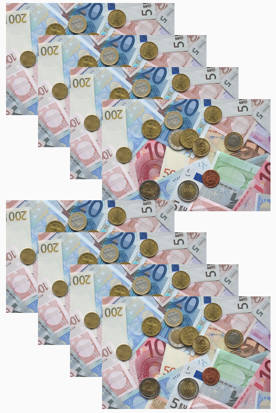
OPTIONS

Total cost	Cost for last week
$40.16	$34.00

Job cost estimation

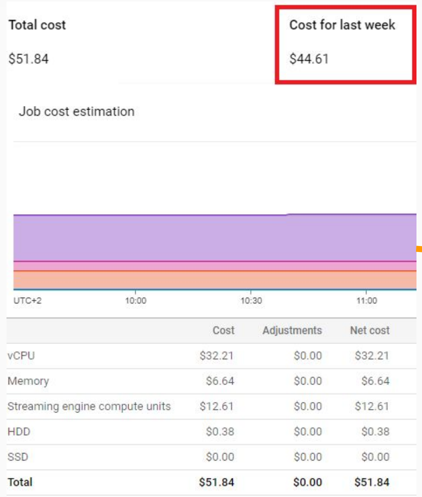
UTC+2 10:00 10:30 11:00

	Cost	Adjustments	Net cost
vCPU	$32.92	$0.00	$32.92
Memory	$6.79	$0.00	$6.79
Processed data	$0.06	$0.00	$0.06
HDD	$0.39	$0.00	$0.39
SSD	$0.00	$0.00	$0.00
Total	**$40.16**	**$0.00**	**$40.16**



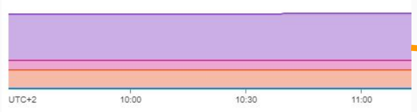
BTR -> Orchestrator (DroolsIO) -> Mediation (S & T with Async ParDo)

Conclusions - some tips

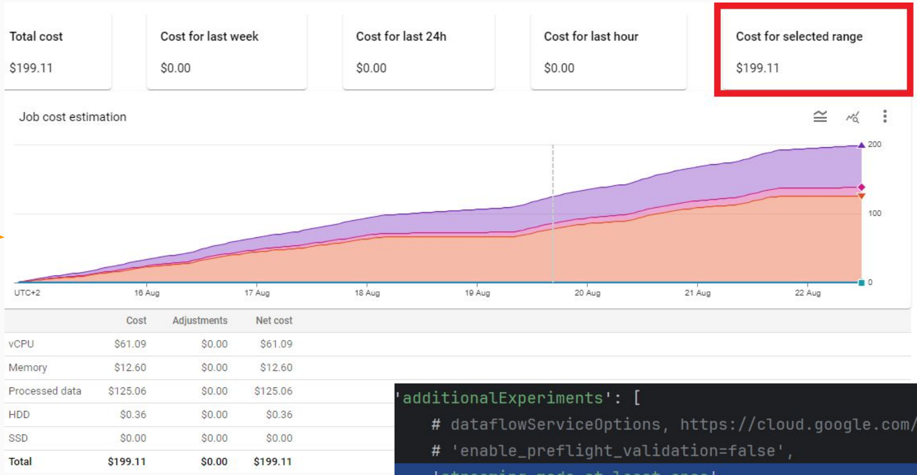


Total cost	Cost for last week
$51.84	$44.61

Job cost estimation

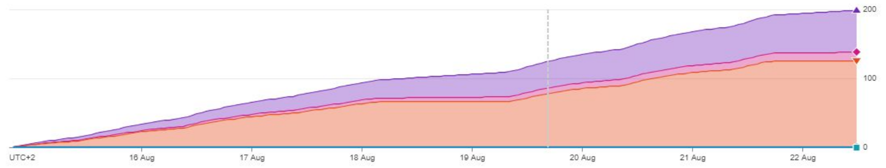


	Cost	Adjustments	Net cost
vCPU	$32.21	$0.00	$32.21
Memory	$6.64	$0.00	$6.64
Streaming engine compute units	$12.61	$0.00	$12.61
HDD	$0.38	$0.00	$0.38
SSD	$0.00	$0.00	$0.00
Total	**$51.84**	**$0.00**	**$51.84**

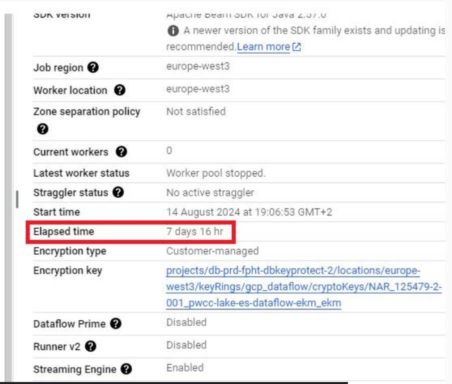


Total cost	Cost for last week	Cost for last 24h	Cost for last hour	Cost for selected range
$199.11	$0.00	$0.00	$0.00	$199.11

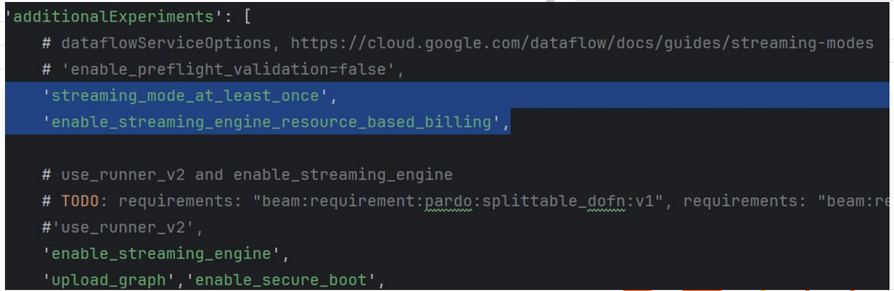
Job cost estimation



	Cost	Adjustments	Net cost
vCPU	$61.09	$0.00	$61.09
Memory	$12.60	$0.00	$12.60
Processed data	$125.06	$0.00	$125.06
HDD	$0.36	$0.00	$0.36
SSD	$0.00	$0.00	$0.00
Total	**$199.11**	**$0.00**	**$199.11**



SDK version	Apache Beam SDK for Java 2.57.0 A newer version of the SDK family exists and updating is recommended. Learn more
Job region	europe-west3
Worker location	europe-west3
Zone separation policy	Not satisfied
Current workers	0
Latest worker status	Worker pool stopped.
Straggler status	No active straggler
Start time	14 August 2024 at 19:06:53 GMT+2
Elapsed time	7 days 16 hr
Encryption type	Customer-managed
Encryption key	projects/db-prd-fpht-dbkeyprotect-2/locations/europe-west3/keyRings/gcp_dataflow/cryptoKeys/NAR_125479-2-001_pwcc-lake-es-dataflow-ekm_ekm
Dataflow Prime	Disabled
Runner v2	Disabled
Streaming Engine	Enabled



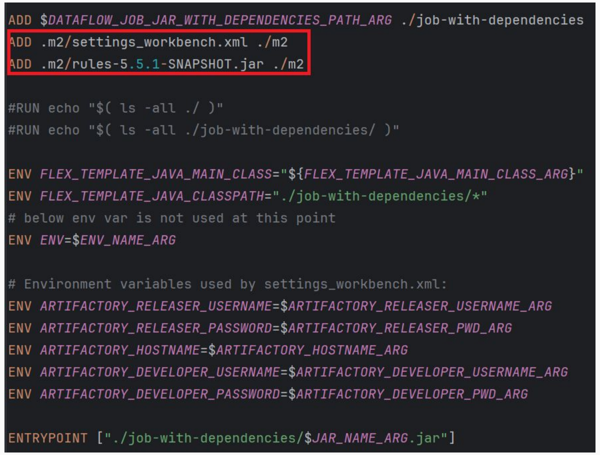
```
'additionalExperiments': [
    # dataflowServiceOptions, https://cloud.google.com/dataflow/docs/guides/streaming-modes
    # 'enable_preflight_validation=false',
    'streaming_mode_at_least_once',
    'enable_streaming_engine_resource_based_billing',

    # use_runner_v2 and enable_streaming_engine
    # TODO: requirements: "beam:requirement:pardo:splittable_dofn:v1", requirements: "beam:re
    #'use_runner_v2',
    'enable_streaming_engine',
    'upload_graph','enable_secure_boot',
```

BEAM SUMMIT

Future Work

- Pre load KieContainer ?
- Generic DroolsIO as ruleInput: **RuleInput[T]**



```
ADD $DATAFLOW_JOB_JAR_WITH_DEPENDENCIES_PATH_ARG ./job-with-dependencies
ADD .m2/settings_workbench.xml ./m2
ADD .m2/rules-5.5.1-SNAPSHOT.jar ./m2

#RUN echo "$( ls -all ./ )"
#RUN echo "$( ls -all ./job-with-dependencies/ )"

ENV FLEX_TEMPLATE_JAVA_MAIN_CLASS="${FLEX_TEMPLATE_JAVA_MAIN_CLASS_ARG}"
ENV FLEX_TEMPLATE_JAVA_CLASSPATH="./job-with-dependencies/*"
# below env var is not used at this point
ENV ENV=$ENV_NAME_ARG

# Environment variables used by settings_workbench.xml:
ENV ARTIFACTORY_RELEASER_USERNAME=$ARTIFACTORY_RELEASER_USERNAME_ARG
ENV ARTIFACTORY_RELEASER_PASSWORD=$ARTIFACTORY_RELEASER_PWD_ARG
ENV ARTIFACTORY_HOSTNAME=$ARTIFACTORY_HOSTNAME_ARG
ENV ARTIFACTORY_DEVELOPER_USERNAME=$ARTIFACTORY_DEVELOPER_USERNAME_ARG
ENV ARTIFACTORY_DEVELOPER_PASSWORD=$ARTIFACTORY_DEVELOPER_PWD_ARG

ENTRYPOINT ["./job-with-dependencies/$JAR_NAME_ARG.jar"]
```

Thank you!

Questions?

Medium Post:
https://medium.com/@serna.alberto.eng/drools-pardo-and-scio-dataflow-a-goodbye-microservices-tale-cb0946de1bc6

LinkedIn:
https://www.linkedin.com/in/albertolose

BEAM SUMMIT