General Notes

Note: This sample was entirely provided by Tim Brown. See the blog below.

Learn more about beam and the example we running

https://beam.apache.org/get-started/wordcount-example/

To understand the custom tracing stuff, see:

https://timbrowndatablog.medium.com/apache-beam-open-telemetry-8f72ba65fe9c

All the goodness to start understanding things for how we ran this is in WordCount.java and TrackableDoFn.java.

Learnings

When you see a trace with ProcessCombineResultsFn that does not have spanlinks it is because the line being processed–see the ExtractWordsFn attribute tracelements–contains all elements for the word in question. This means the word being counted is not in other ExtractWordsFn traces.

If you see multiple spanlinks under one ProcessCombineResultsFn it means the word is multiple other traces.

Environment requirements

1. `protoc --version`
    a. `libprotoc 3.21.8`

2. `mvn --version`
```
Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63)
Maven home: /usr/local/Cellar/maven/3.8.6/libexec
Java version: 18.0.1.1, vendor: Homebrew, runtime:
/usr/local/Cellar/openjdk/18.0.1.1/libexec/openjdk.jdk/Contents/Home
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "11.6.1", arch: "x86_64", family: "mac"
```
3. `/usr/libexec/java_home -V`
```
Matching Java Virtual Machines (4):
19.0.1 (x86_64) "Oracle Corporation" - "Java SE 19.0.1"
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home
```

```
17.0.5 (x86_64) "Oracle Corporation" - "Java SE 17.0.5"
/Library/Java/JavaVirtualMachines/jdk-17.0.5.jdk/Contents/Home
16.0.2 (x86_64) "Oracle Corporation" - "Java SE 16.0.2"
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home
11.0.11 (x86_64) "AdoptOpenJDK" - "AdoptOpenJDK 11"
/Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home
```

4. `export JAVA_HOME=$(/usr/libexec/java_home -v 11)`
5. Install protobuf-java locally (kept causing me errors without)
   a. `mvn install:install-file -Dpackaging=jar -DgeneratePom=true -DgroupId=com.google.protobuf   -DartifactId=protobuf-java -Dfile=protobuf-java-3.21.8.jar -Dversion=3.21.8`

Docker compose for the collector

My docker-compose setup for my collector as the example is sending Jaeger

```
version: "3.8"

services:

  otel-collector:
    image: otel/opentelemetry-collector
    command: ["--config=/etc/otel-collector-config.yaml"]
    volumes:
      - ./config/config.yml:/etc/otel-collector-config.yaml
    ports:
      - "1888:1888"   # pprof extension
      - "8888:8888"   # Prometheus metrics exposed by the collector
      - "8889:8889"   # Prometheus exporter metrics
      - "13133:13133" # health_check extension
      - "4317:4317"   # OTLP gRPC receiver
      - "4318:4318"   # OTLP http receiver
      - "55679:55679" # zpages extension
      - "14250:14250" # jaeger
    environment:
      - HONEYCOMB_APIKEY=${HONEYCOMB_API_KEY}

    environment:
      MYSQL_PASSWORD: honeycomb
      MYSQL_ROOT_PASSWORD: honeycomb
      MYSQL_DATABASE: honeycomb

volumes:
  collector:
```

Collector config

```
receivers:
  otlp:
    protocols:
      grpc: # on port 4317
  jaeger:
    protocols:
      grpc:
      thrift_binary:
      thrift_compact:
      thrift_http:

  prometheus:
    config:
      scrape_configs:
        - job_name: "honeycomb-metrics"
          scrape_interval: 60s
          static_configs:
            - targets: ["0.0.0.0:8888"]

processors:
  batch: {}


exporters:
  otlp/traces:
    endpoint: "api.honeycomb.io:443"
    tls:
      insecure_skip_verify: true
    headers:
      "x-honeycomb-team": <your key>

  otlp/metrics:
    endpoint: "api.honeycomb.io:443"
    tls:
      insecure_skip_verify: true
    headers:
      "x-honeycomb-team": <your key>
      "x-honeycomb-dataset": collector-metrics

  logging:
    loglevel: debug

extensions:
  health_check:
  zpages:

service:
  extensions: [health_check, zpages]
  pipelines:
    traces:
```

```
    receivers: [otlp,jaeger]
    processors: [batch]
    exporters: [otlp/traces, logging]
  metrics:
    receivers: [prometheus]
    processors: [batch]
    exporters: [otlp/metrics]

telemetry:
  logs:
    level: "debug"
```

Steps to Run It

1. Set the java version to 11 or if you can fix so works with recent stuff, all good
   a. `export JAVA_HOME=$(/usr/libexec/java_home -v 11)`
2. Make sure your environment requirements correspond to the above ones
3. Go to terminal or VS Code terminal
4. `mvn clean`
5. `mvn install -U`
6. Start collector using Docker compose
7. Compile and set main
   a.
   ```
   mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \
    -Dexec.args="--inputFile=holmes.txt --output=counts" -Pdirect-runner
   ```

Misc

1. Common errors that often require you to run it twice…I suspect timing with Apache Beam, and it's libraries.
2. Set things to Java 11
   a. [ERROR] Failed to execute goal org.codehaus.mojo:exec-maven-plugin:3.1.0:java (default-cli) on project word-count-beam: An exception occurred while executing the Java class. null: ExceptionInInitializerError: unsupported Java version: 18 -> [Help 1]
3. Make sure your collector is running and your Jaeger setup is pointed to that instance:
   a. Error: SEVERE: Failed to export spans. Server is UNAVAILABLE. Make sure your collector is running and reachable from this network. Full error message:io exception
   Oct 27, 2022 3:11:48 PM io.opentelemetry.sdk.internal.ThrottlingLogger doLog