Stream Processing with MicroProfile and Apache Kafka

Matthias Wessendorf – Red Hat | matzew AT redhat DOT com | @mwessendorf

Background: Motivation for MicroService and Kafka

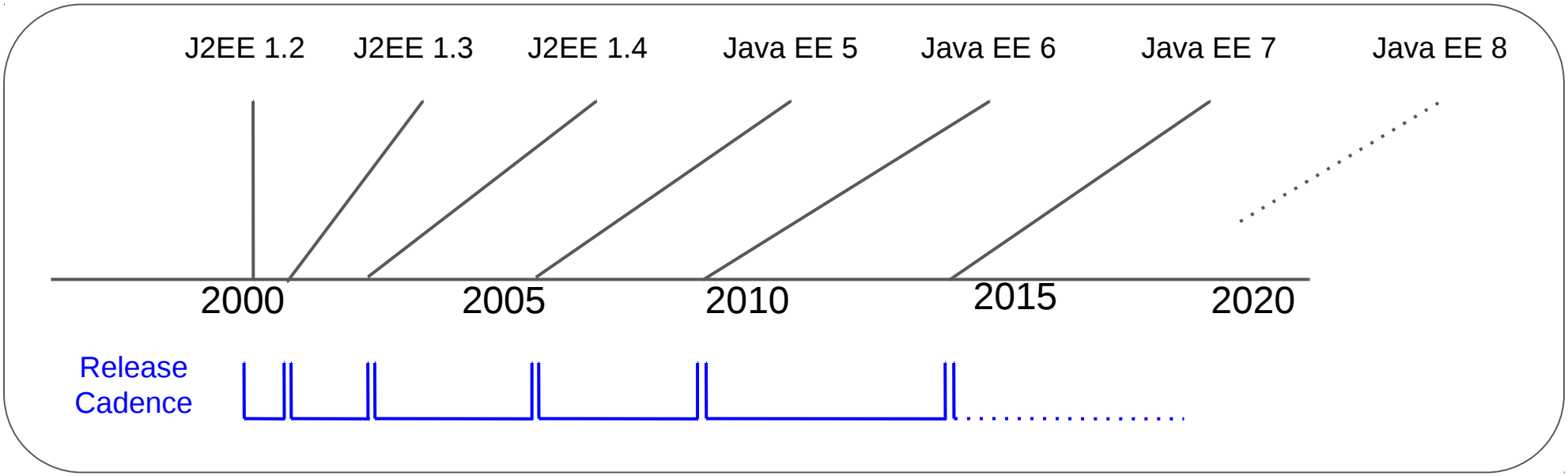
Matthias Wessendorf – Red Hat | matzew AT redhat DOT com | @mwessendorf

Agenda

- **MicroProfile**
- WildFly Swarm
- Apache Kafka
- Integrating MicroProfile and Kafka
- CDI Extension for Apache Kafka
- Outlook

MicroProfile

Enterprise Java Standards History



Matthias Wessendorf – Red Hat | matzew AT redhat DOT com | @mwessendorf

MicroProfile Background

Began as a collection of independent discussions

- Many innovative “microservices” efforts in existing Java EE projects
  - WildFly Swarm
  - WebSphere Liberty
  - Payara
  - TomEE
- Projects already leveraging both Java EE and non-Java EE technologies
- Creating new features/capabilities to address microservices architectures

Quickly realized there is common ground

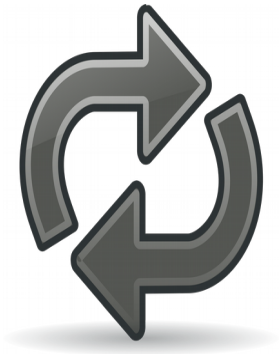
Java EE technologies are already being used for microservices,
but we can do better

MicroProfile Release Philosophy

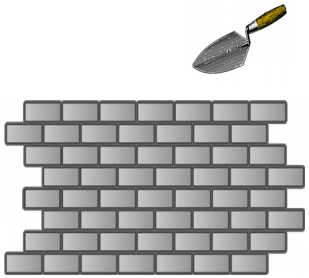
Release 1.0

JAX-RS
CDI
JSON-P

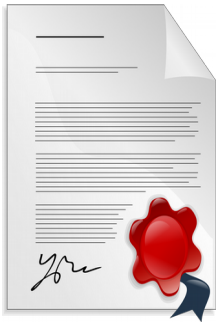
Rapidly iterate and innovate



Build consensus



Standardize



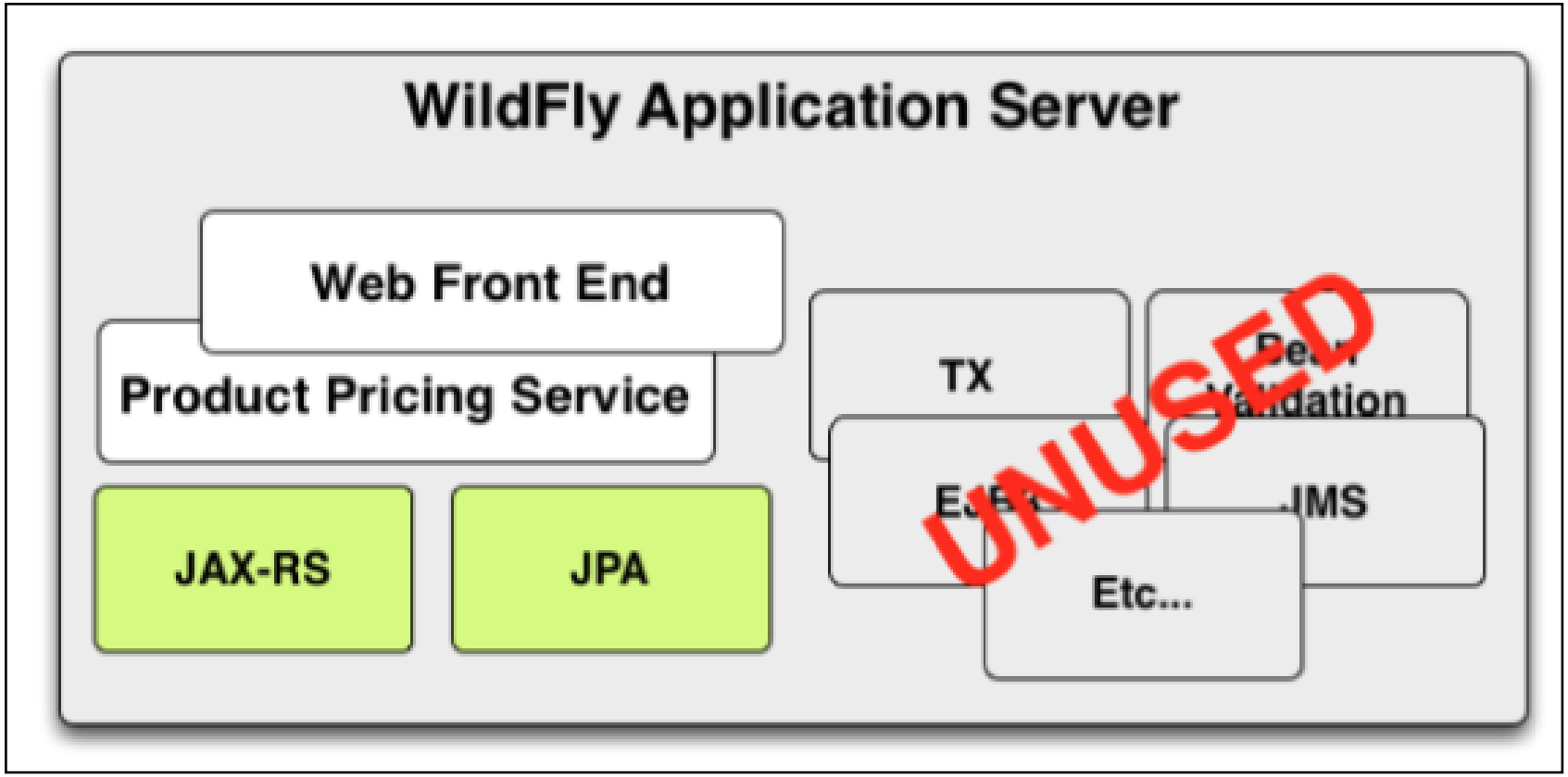
Sept 2016

Matthias Wessendorf – Red Hat | matzew AT redhat DOT com | @mwessendorf

Agenda

- MicroProfile
- **WildFly Swarm**
- Apache Kafka
- Integrating MicroProfile and Kafka
- CDI Extension for Apache Kafka
- Outlook

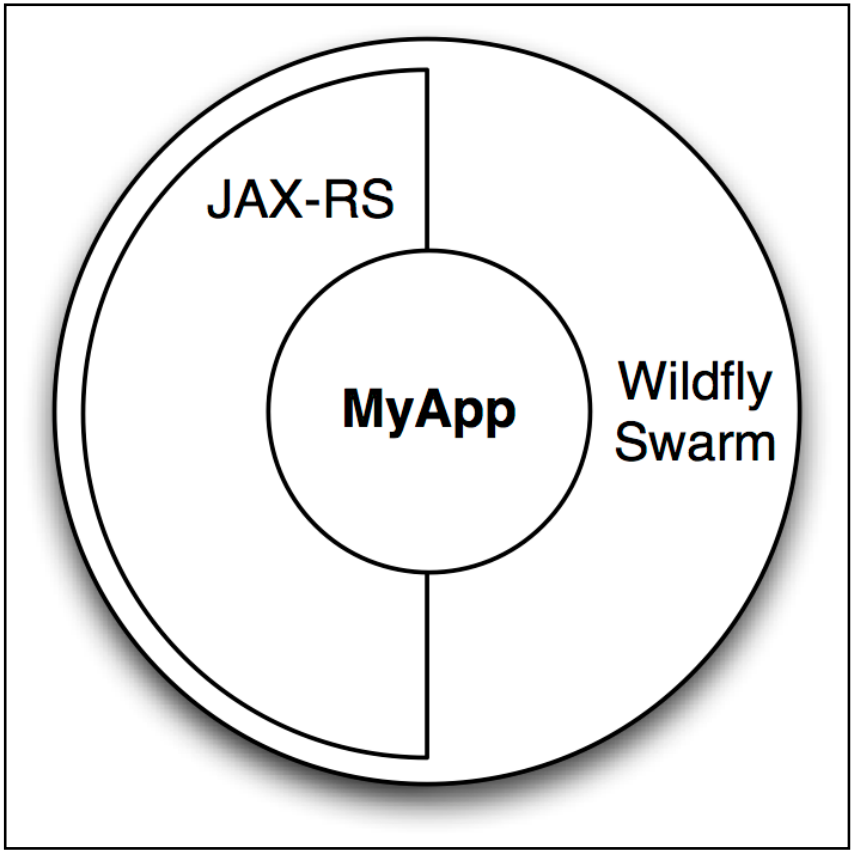
Just Enough App Server



- Use the API's you want
- Include the capabilities you need
- Wrap it up for deployment

Uberjar

- A single .jar file containing your application,
- the portions of WildFly required to support it,
- an internal Maven repository of dependencies,
- plus a shim to bootstrap it all



Fractions

- A well-defined collection of application capabilities.
  - May map directly to a WildFly subsystem,
  - or bring in external capabilities such as Netflix Ribbon.

What Fractions do

- Enable WildFly subsystems (JAX-RS, Infinispan)
- Integrate additional system capabilities (Topology)
- Provide deployments (ribbon-webapp, jolokia)
- Alter deployments (keycloak)

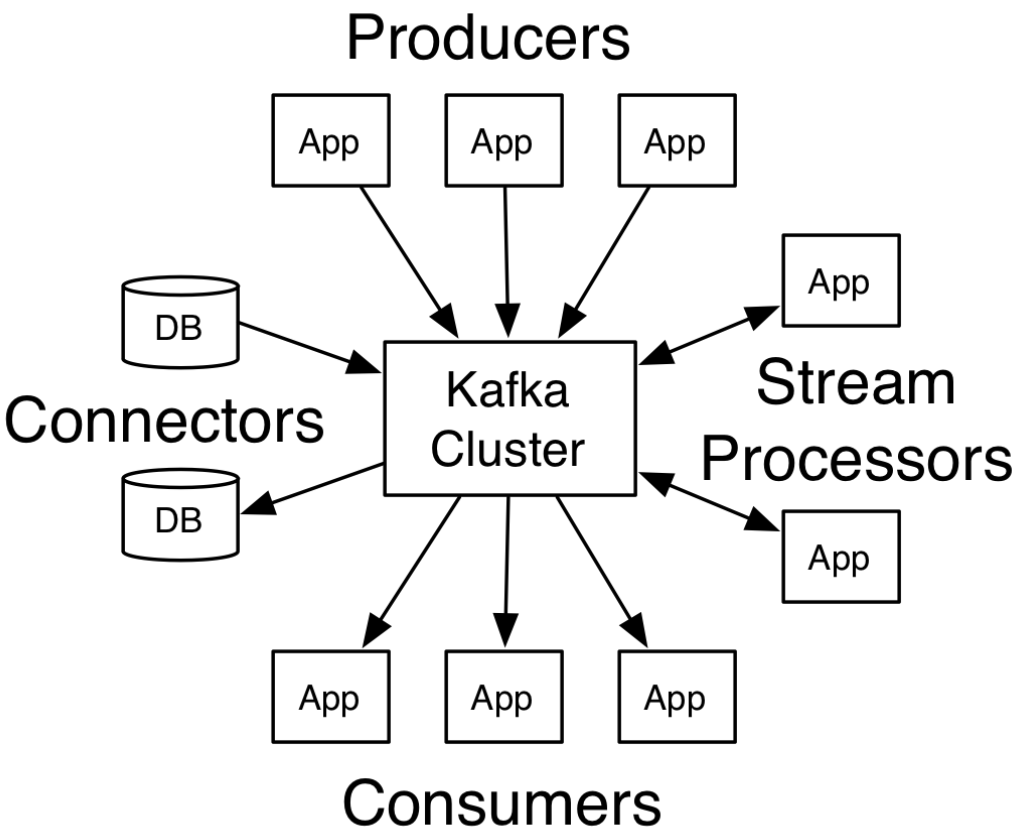
DEMO

… MicroProfile Server App using Swarm!

Agenda

- MicroProfile
- WildFly Swarm
- **Apache Kafka**
- Integrating MicroProfile and Kafka
- CDI Extension for Apache Kafka
- Outlook

Apache Kafka

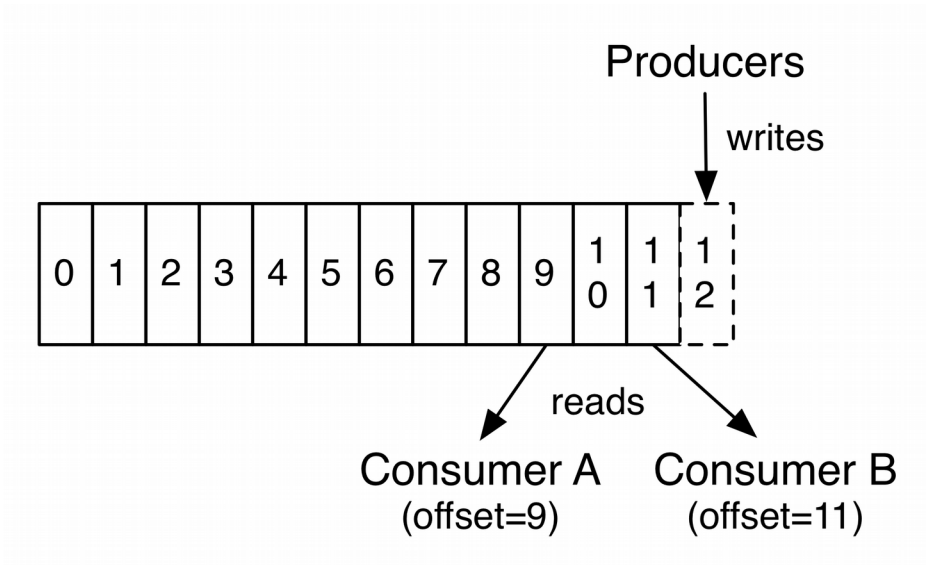


- like messaging system, but different
  - “distributed commit log”
- Clustering is CORE...
- Durability & Ordering Guarantees
- Typical Use-Cases
  - ETL / **C**hange **D**ata **C**apture
    - http://debezium.io (CDC)
  - Data Pipeline: Kafka as the HUB for other systems
  - User activity tracking/reporting
  - analytics….

DEMO

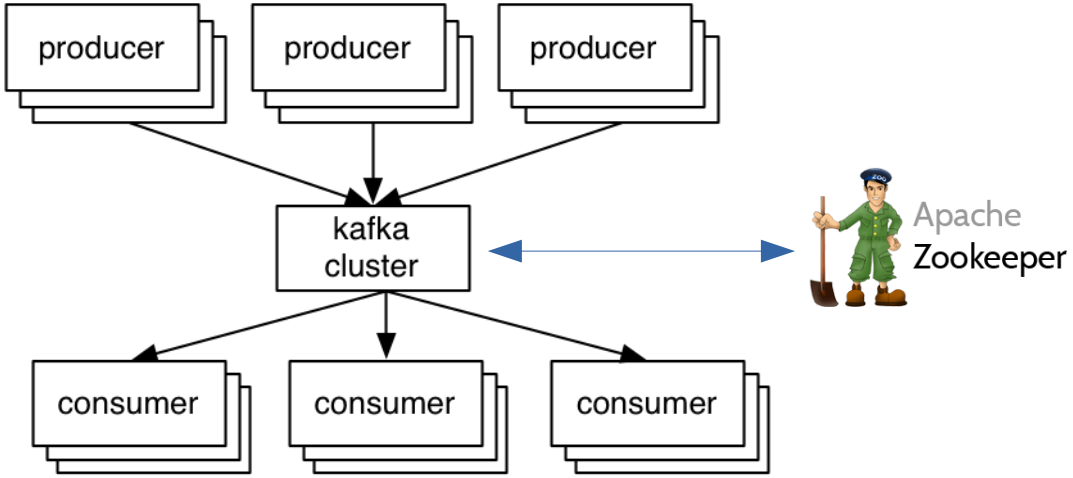
From WebSocket to Apache Kafka

Records (or Messages)



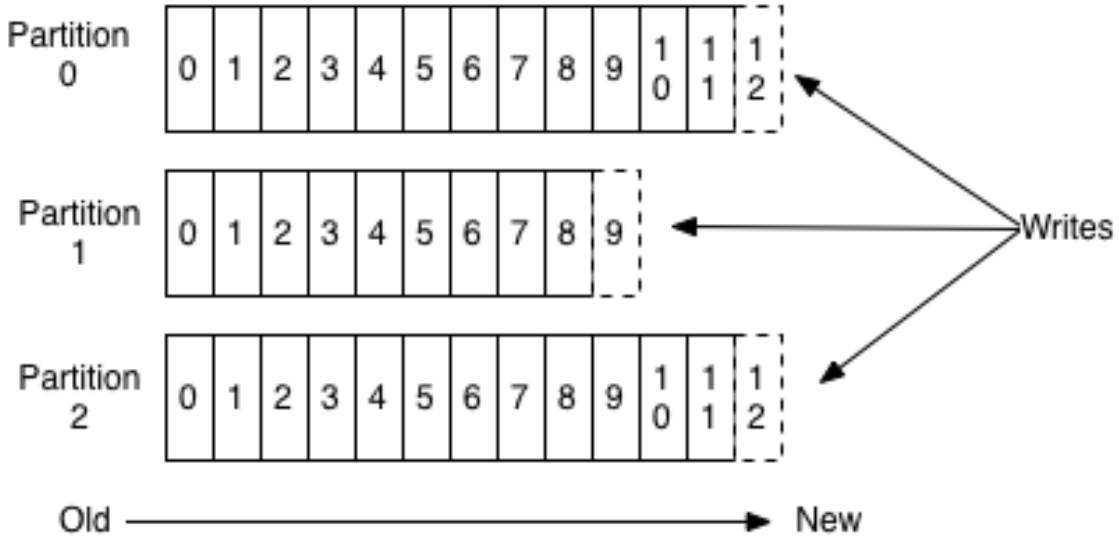
- Byte Array
  - Key/Value pairs
- Immutable
- Records (or messages, or events) are being appended
- Persisted to disk

Producers and Consumers



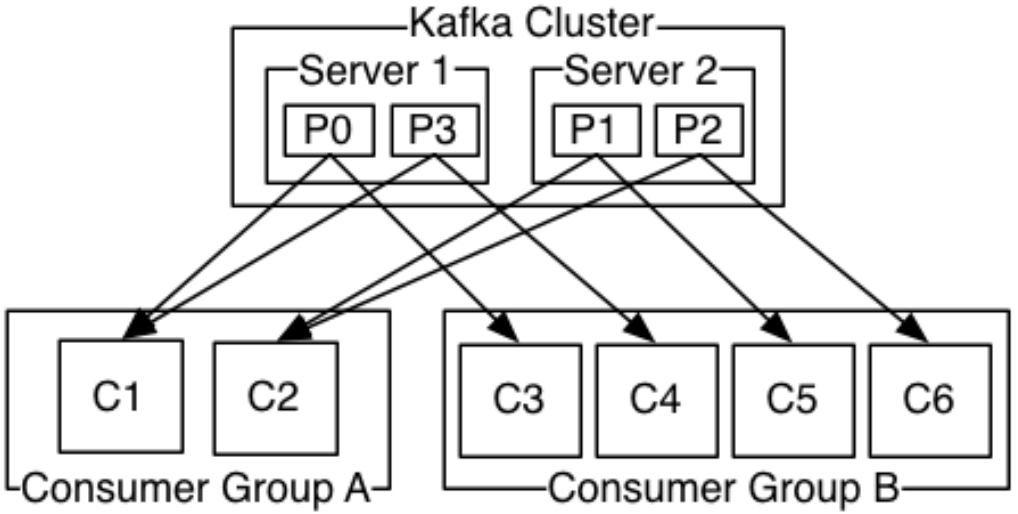
- n nodes/brokers → Kafka cluster (clients connect to bootstrap servers)
  - Apache Zookeeper
- Producer sends message to a broker
- Consumer is connected to a broker, and *polls* message from a broker
- Leader/Follower architecture...

Topics, Partitions and Offsets



- Topic is containing 1 or more partitions
  - Guaranteed ordering (“only” on a Partition of a Topic)
- Replication of the partitions (Leader/Follower)
  - Partitioning-Factor (per Topic) is configured when setting up a Topic
- Offset: unique sequential ID per TopicPartition
- Consumer keeps track of offset
  - Reply or handling consumers with different speed! :-)

Consumer Groups



- Logical grouping of some Kafka consumers
  - groups receive msg from Topic: ***AT_LEAST_ONCE***
    - individual consumer: assigned to partition(s) of the cluster
- Separate *scaling* for each consumer group (listening on same Topic)
  - Example:
    - Group A: expensive/non-time-sensitive → scale down....
    - Group B: realtime processing / time-sensitive → scale up

DEMO

WebSocket demo: behind the sceens…

Some details on Apache Kafka's Java API (0.10.2.0)

Agenda

- MicroProfile
- WildFly Swarm
- Apache Kafka
- **Integrating MicroProfile and Kafka**
- CDI Extension for Apache Kafka
- Outlook

Integration: Kafka and Microprofile

- Kafka's Java library is easy to integrate
- Wiring of Producers and Consumers with CDI
- Contexts and Dependency Injection (CDI) for the Java EE platform
  - Contexts: The ability to bind the lifecycle and interactions of stateful components to well-defined but extensible lifecycle contexts
  - Dependency injection: The ability to inject components into an application in a typesafe way, including the ability to choose at deployment time which implementation of a particular interface to inject
- CDI is intended to be a foundation for frameworks, extensions and integration with other technologies!

```java
public class ProcessorBean {

    @Inject
    private KafkaProducer<String, String> producer;

    public void publishEventRecord(final String payload) {
        producer.send(new ProducerRecord<String, String>( topic: "some_topic", payload));
    }
}
```

```java
@Produces
public KafkaProducer createProducer() {

    final Properties properties = new Properties();

    properties.put(BOOTSTRAP_SERVERS_CONFIG, "172.17.0.3:9092");
    properties.put(KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
    properties.put(VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class);

    return new KafkaProducer<>(properties);

}
```

```java
@ApplicationScoped
public class ConsumerRegistrationFactory {

    Logger logger = Logger.getLogger(ConsumerRegistrationFactory.class.getName());

    // This will add a new thread to our pool, to subscribe to our Observable
    @Resource(name = "DefaultManagedExecutorService")
    private ManagedExecutorService executor;

    public void init(@Observes @Initialized(ApplicationScoped.class) Object init) {
        logger.severe( msg: "Setup my consumer");

        // create (all) consumer(s)
        MyKafkaConsumer myKafkaConsumer = new MyKafkaConsumer();

        // submit to managed executor service
        executor.submit(myKafkaConsumer);

    }
}
```

Agenda

- MicroProfile
- WildFly Swarm
- Apache Kafka
- Integrating MicroProfile and Kafka
- **CDI Extension for Apache Kafka**
- Outlook

CDI *portable* extensions for Apache Kafka

- CDI is intended to be a foundation for frameworks, extensions and integration with other technologies!
  - Customize the platform for individual needs!
- Removes boilerplate code, makes Kafka usage really easy!
- CDI extension requires 3 “things”
  - beans.xml (*optional since CDI 1.1*)
  - services file
  - Implementation class: POJO observing the CDI lifecycle events
- CDI: A ***great!*** way for extending the standardized platform!
  - Hence it was critical for MicroProfile too!

Meet kafka-cdi

… A simple CDI extension for Apache Kafka

https://github.com/matzew/kafka-cdi

Matthias Wessendorf – Red Hat | matzew AT redhat DOT com | @mwessendorf

Agenda

- MicroProfile
- WildFly Swarm
- Apache Kafka
- Integrating MicroProfile and Kafka
- CDI Extension and Swarm Fraction
- **Outlook**

AeroGear UPS

POC:
- Swarm based JAX-RS endpoint for Push
- Kafka as the event stream
- Consumer to process Push Metrics (e.g. from Apple)

There is more…. much more…! Believe me, it's true!

- KStream API
  - New API, build on-top of Kafka's Java client
    - Functional programming to filter/map/reduce streams
  - No need for complexer frameworks like Spark or Flink
- Vert.x
  - Nice and simple wrapper around Kafka's Java client
- Debezium platform for CDC
  - contains `KafkaCluster` class for testing!, or demos :-)
- Future options:
  - More CDI / Swarm enhancements (e.g. JCA, Swarm Fraction)

THANKS!

Questions ?

Beer !

Cocktails!

<u>Slides and (some) demos:</u>

https://github.com/matzew/kafka-microprofile

Matthias Wessendorf – Red Hat | matzew AT redhat DOT com | @mwessendorf