# How are you connecting to configuration client?

# How are you connecting to Azure Key Vault?

# How are you connecting and writing logs to Azure Application insights?

# How are you gracefully shutting down the application basically Kafka stream and how are you processing cached data?

* A configuration called stream checker is added to Azure application configuration which will be read and cached when the application is started.
* And then one thread will be started which runs at scheduled time and read this config. If this configuration Is disabled, then application will continue to run if the configuration is enabled then we will call Kafka Streams. close () So this way we will
* Then in the same thread we will set one static member called stream checker to true. This value will be read by each consumer threads once the queue size is empty. If queue size is empty and if stream checker is set to true, then it is understood that graceful shutdown configuration is enabled so we will come out of loop of thread and shutdown the thread gracefully so that we will not miss any data.

# How blocking queue is implemented?

com.shaurya.ds.blockingqueue.PreprocBlockingQueue

# How Processor topology is implemented in preprocessor?

* 2.5.0 Kafka stream and Kafka client jars are used
* Javadoc <https://kafka.apache.org/25/javadoc/index.html?org/apache/kafka/streams/>

// A logical representation of a ProcessorTopology. A topology is an acyclic graph of sources, processors, and sinks

*Topology topology = new Topology ();*

*//Add a new source to the topology that consumes from the topics, read records, and forward the records to child processor and/or sink nodes*

*topology.addSource(“NAME\_GIVEN\_TO\_SOURCE\_PROCESSOR”,”ONE OR MORE TOPIC NAMES”);*

*// Add a new processor node to the topology that receives and processes records output by one or more parent source or processor node. Any new record output by this processor will be forwarded to its child processor or sink nodes.*

*topology.addProcessor(“NAME\_OF\_THE\_PROCESSOR”, () -> fileExtractProcessor, “PARENT\_PROCESSOR\_NODE\_NAMES”);*

*() -> fileExtractProcessor is supplier - the supplier used to obtain this node's*[*Processor*](https://kafka.apache.org/25/javadoc/org/apache/kafka/streams/processor/Processor.html)*instance*

*//A Kafka client that allows for performing continuous computation on input coming from one or more input topics and sends output to zero, one, or more output topics.*

*KafkaStreams streams = new KafkaStreams(topology,****Properties of kafka configs****);*

*// Set the handler invoked when a internal thread abruptly terminates due to an uncaught exception*

*streams.setUncaughtExceptionHandler((Thread t2, Throwable e) -> {*

*log.error(e.getMessage());*

*});*

*//Start the KafkaStreams instance by starting all its threads. This function is expected to be called only once during the life cycle of the client.*

*streams.start();*

*//Shutdown this KafkaStreams instance by signaling all the threads to stop, and then wait for them to join. This will block until all threads have stopped.*

*Runtime.getRuntime().addShutdownHook(new Thread("shutdown-hook") {*

*@Override*

*public void run() {*

*streams.close();*

*latch.countDown();*

*}*

*});*

***Properties of kafka configs***

Properties props = new Properties();

props.put(StreamsConfig.CLIENT\_ID\_CONFIG, “UNIQUE\_ID\_FOR\_CLIENT”);

props.put(StreamsConfig.APPLICATION\_ID\_CONFIG, “GROUP\_ID”);

props.put(StreamsConfig.BOOTSTRAP\_SERVERS\_CONFIG,”In azure it is URL of eventhub namespace”);

props.put(StreamsConfig.REPLICATION\_FACTOR\_CONFIG, 1);

props.put(StreamsConfig.TOPOLOGY\_OPTIMIZATION, StreamsConfig.OPTIMIZE);

props.put(SACConstants.KAFKA\_DEFAULT\_API\_TIMEOUT\_MS,””);

props.put(StreamsConfig.DEFAULT\_KEY\_SERDE\_CLASS\_CONFIG, Serdes.String().getClass());

props.put(StreamsConfig.DEFAULT\_VALUE\_SERDE\_CLASS\_CONFIG, Serdes.ByteArray().getClass());

props.put(StreamsConfig.SECURITY\_PROTOCOL\_CONFIG,”SASL\_SSL”));

props.put(SaslConfigs.SASL\_MECHANISM, “PLAIN”);

props.put(SaslConfigs.SASL\_JAAS\_CONFIG, “SASL\_JASS\_CONFIG”);

# How ***Stream processor*** is implemented in Preprocessor?

FileExtractProcessor is the class which is stream processor in Preprocessor application.

*public class FileExtractProcessor* ***implements*** *Processor<String, byte[]>*

*@Override*

*public void* ***process****(String key, byte[] bytearray) {*

*try {*

*blockingQueue.put(bytearray);*

*} catch (Exception e) {*

*log.error("Exception when inserting message to blocking queue", e);*

*}*

*}*

***// A processor of key-value pair records.***

*public interface Processor<K, V> {*

*void init(ProcessorContext var1);*

*void process(K var1, V var2);*

*void close();*

*}*

# What is business logic of preprocessor?

Read metadata from event hub, from metadata get path of zip file specified and extract it. if it has xml file convert it to json. Then based on configurations segregate the data and write to multiple event hubs.

# How data is written to destination event hubs?

We are writing using KafkaProducer API of kafka-clients jar.

*KafkaProducer<?, ?> producer = new KafkaProducer(properties, new StringSerializer(), new ByteArraySerializer());*

*kafKaMsgProcess.getProducer().send(new ProducerRecord<String, Object>(kafKaMsgProcess.getTopic(),*

*kafKaMsgProcess.getRunKeyId(), kafKaMsgProcess.getCompressedBytes()), new Callback() {*

*public void onCompletion(RecordMetadata metadata, Exception exception) {*

*if (exception != null) {*

*throw new SacCollectionAzureException(exception.getMessage(),*

*exception.fillInStackTrace());*

*}*

*}*

*});*