Дмитрий
Морозовский
Huawei

Женим Kafka на Kotlin Native

Или удивительные приключения Java DeVa в стране Kotlin Native

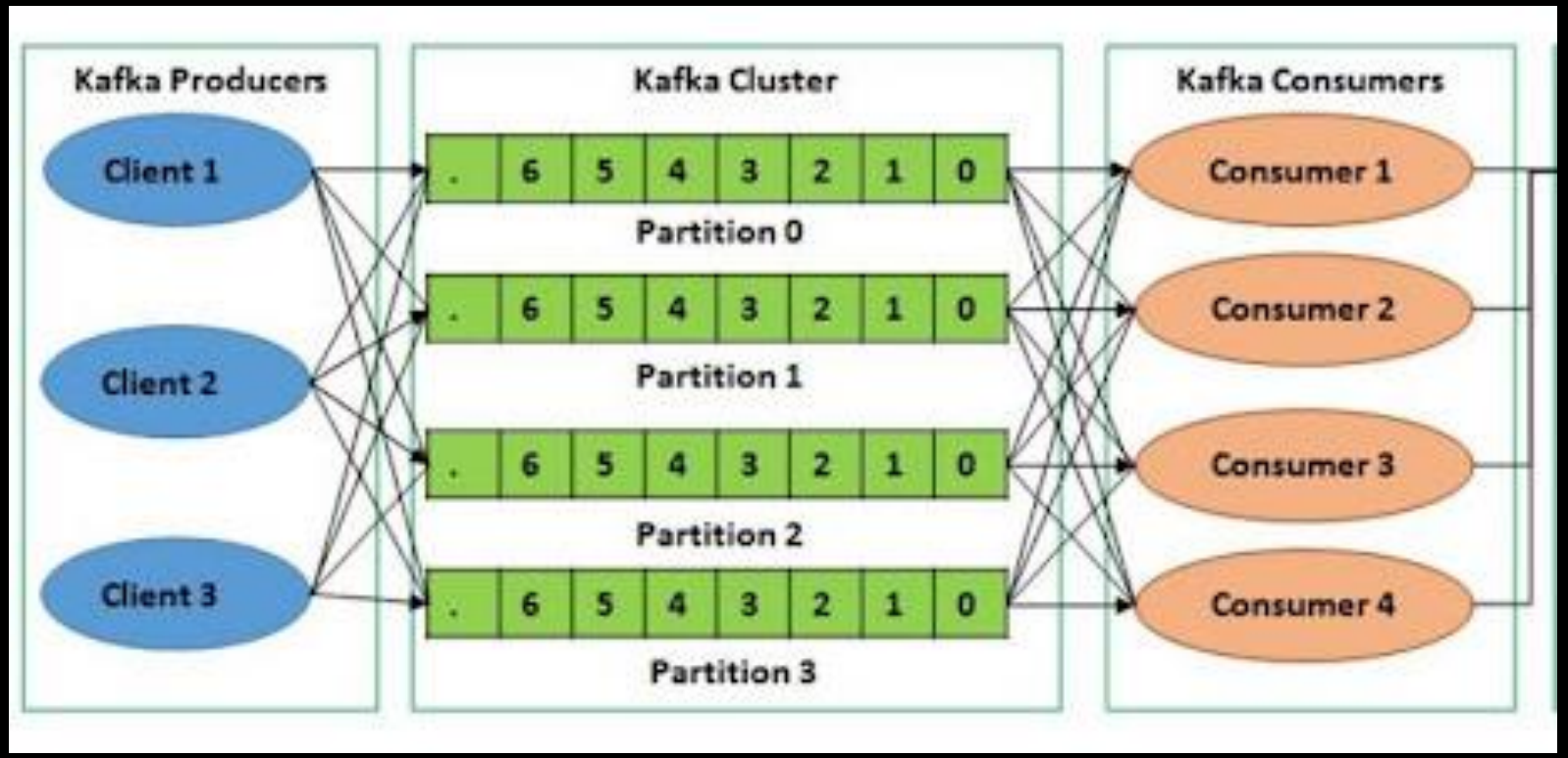
Structure of the talk

- What is Kafka client and why we need it on Kotlin native
- How Kotlin MPP project build and compilations
- C-interop problems
- Multithreading programming problems

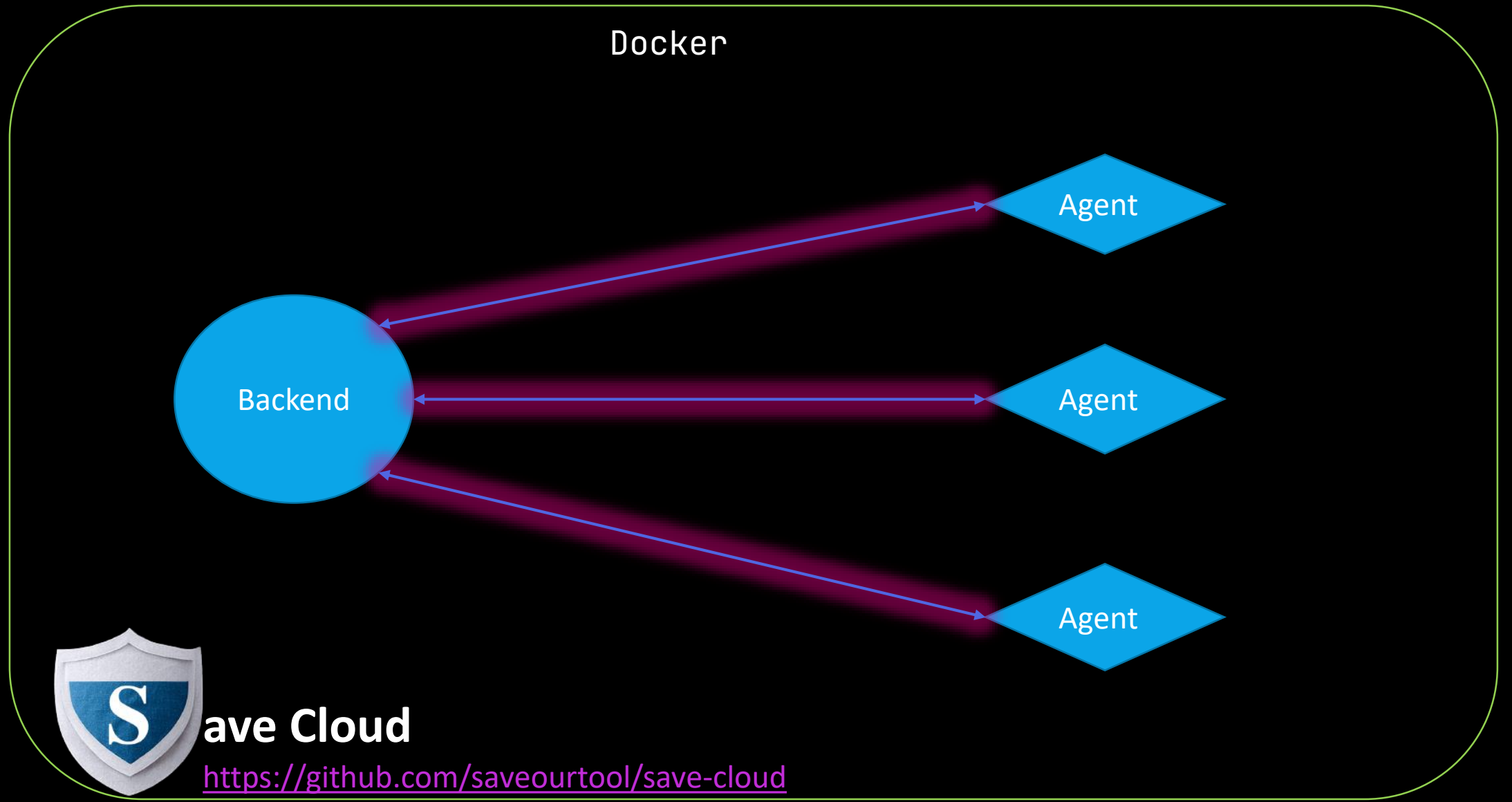
“ 'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop.' ”

Lewis Carroll (Alice's Adventures in Wonderland)

What is kafka client?



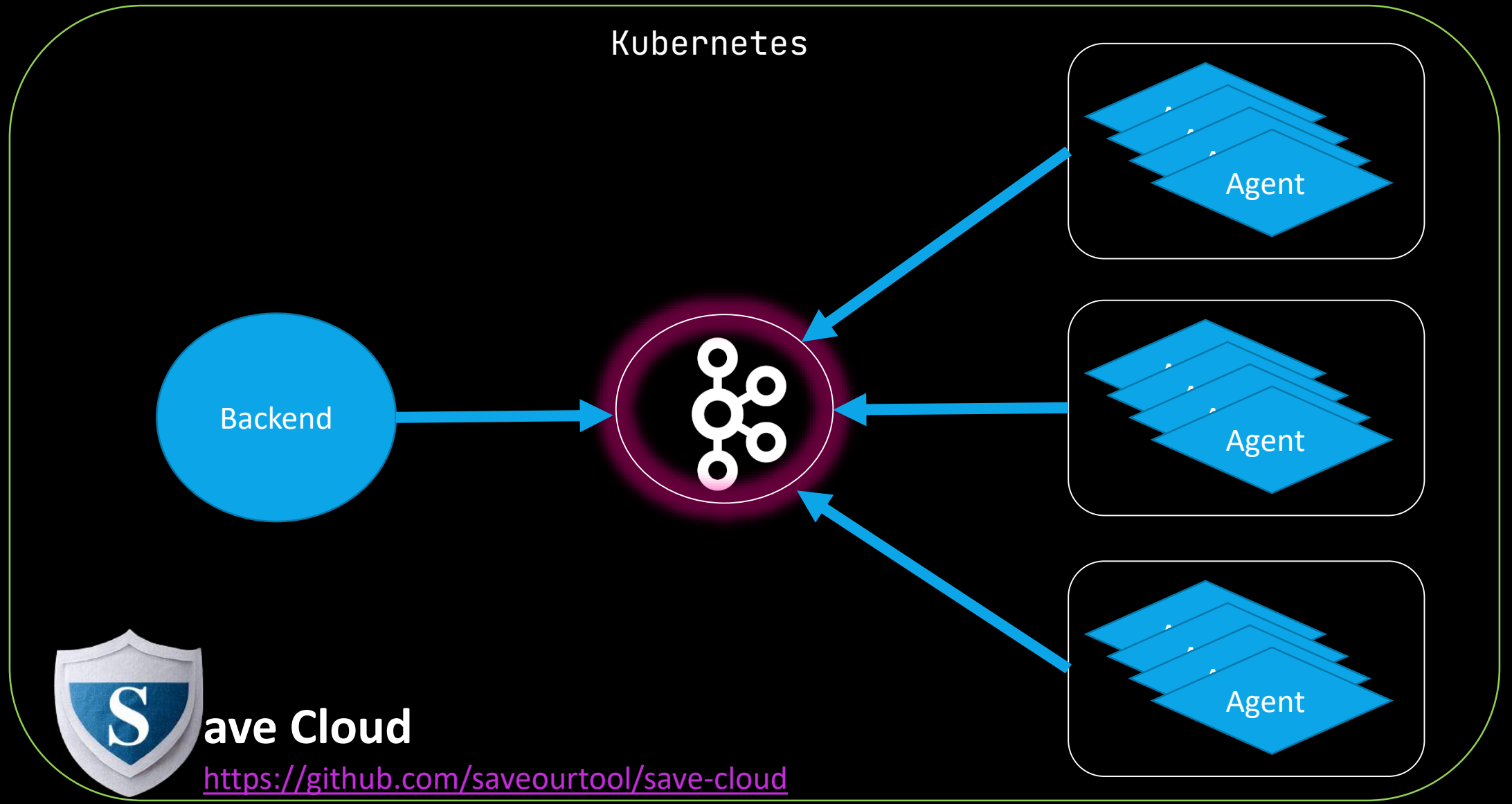
Kafka client on Kotlin Native.Why?



Save Cloud

https://github.com/saveourtool/save-cloud

Kafka client on Kotlin Native.Why?



Kafka client on Kotlin Native.Why?

Kubernetes

Backend

Agent

Agent

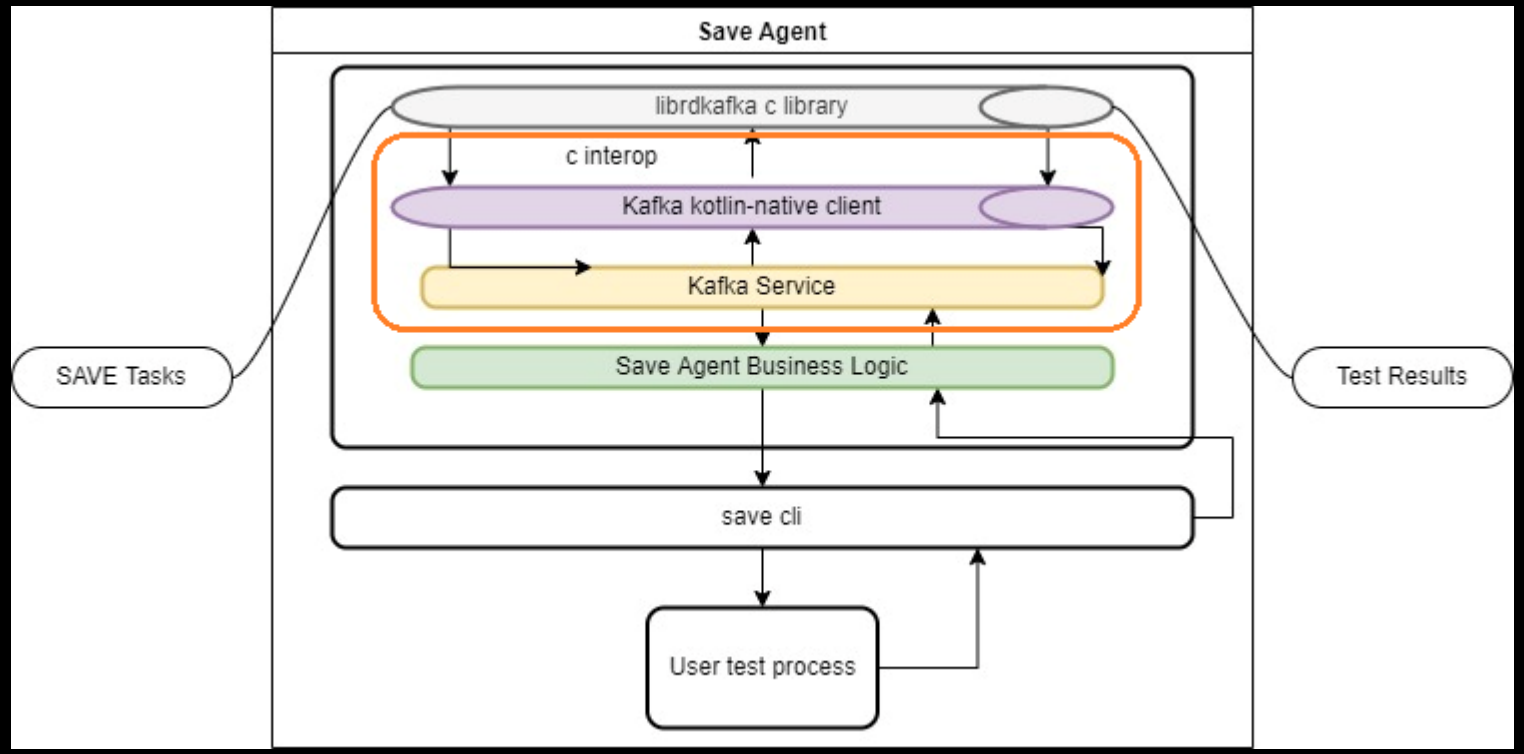
Agent

Save Cloud

https://github.com/saveourtool/save-cloud

Kafka client on Kotlin Native.Why?

- Should run from kotlin native component
- Want to use one language for all cloud components



https://github.com/icemachined/kafka-kotlin-native

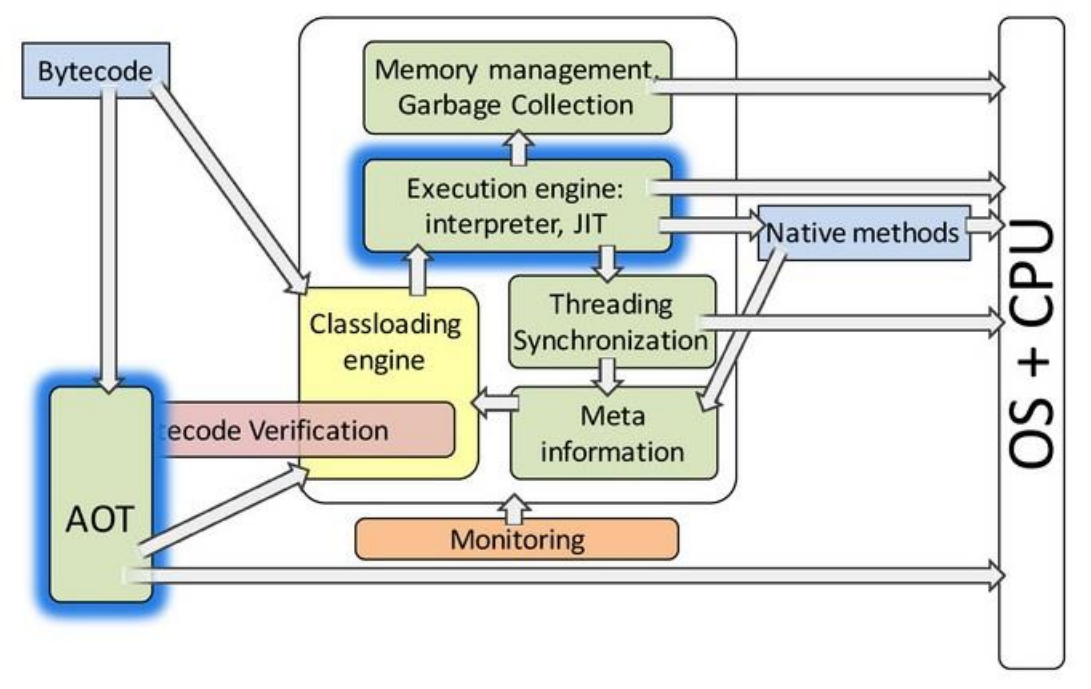
Librdkafka - c/c++ kafka client

- Implementing from scratch is too complicated
- Industry standard used by many companies
- Has high performance
- Used by some other kafka clients Python, Node.js, C#, Haskel etc

https://github.com/edenhill/librdkafka

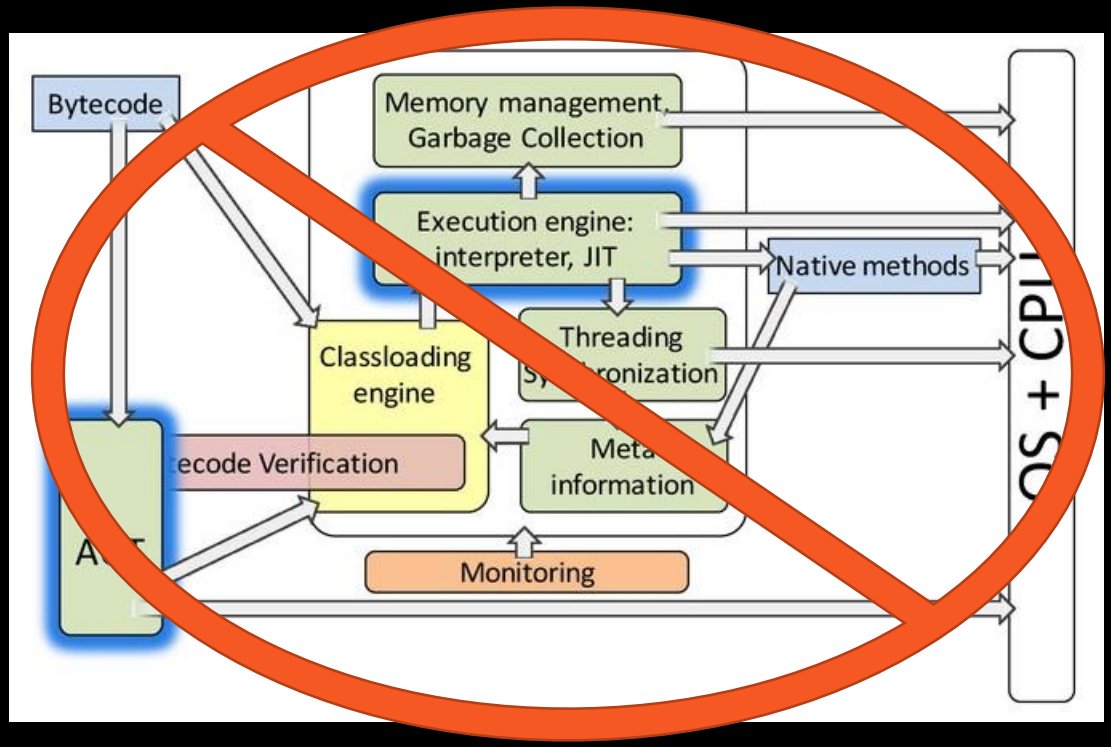
Kotlin Native advantages

- Pure native, no dependency on jvm runtime.



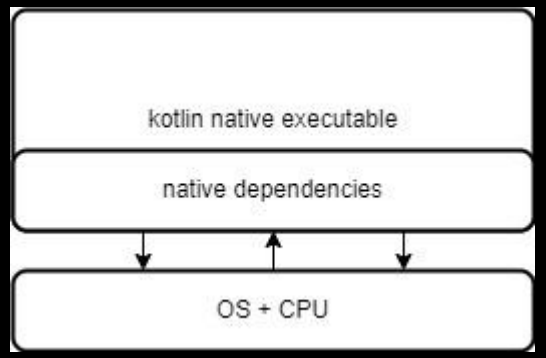
Kotlin Native advantages

- Pure native, no dependency on JVM runtime.



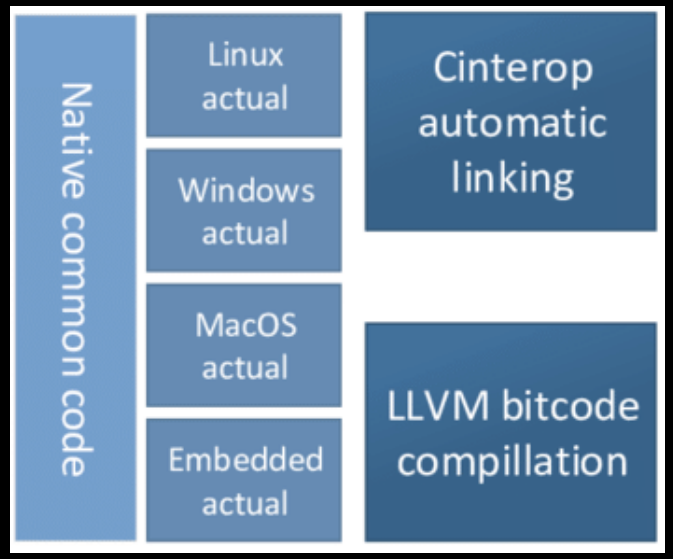
Kotlin Native advantages

- Small executable size 4M instead of ~200M with spring native
- Faster start time,
- Less memory consumption



Kotlin Native advantages

- More efficient interoperability with native libraries then JNI,JNA
- Less concurrency errors as they are detected at compile time
- Multiplatform build for Linux Windows and MacOS.

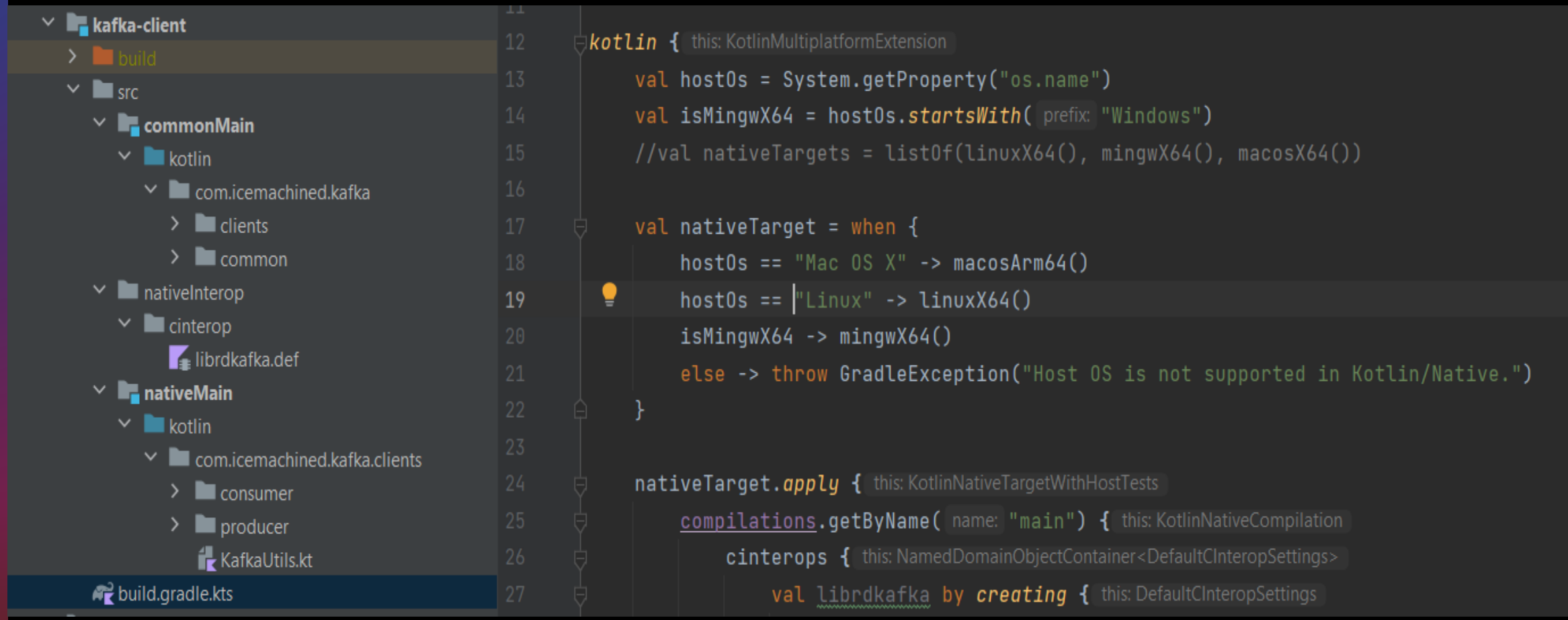


Kotlin Native disadvantages

- Lack of good tooling for debugging
- Less ready libraries, sometimes need to do interops

Kotlin Multiplatform Project

- Build.gradle.kts



C-interop library definition

```
headers = librdkafka/rdkafka.h
headerFilter = librdkafka/*

compilerOpts.linux = -I/usr/include/ -I/usr/include/x86_64-linux-gnu
compilerOpts.mingw = -I../packages/librdkafka.redist.1.9.2/build/native/include
compilerOpts.osx = -I/opt/homebrew/include -I/usr/local/include
linkerOpts.osx = -L/opt/local/lib -L/usr/local/opt/librdkafka/lib -L/opt/homebrew/lib -
L/usr/local/lib -lrdkafka
linkerOpts.linux = -L/usr/lib/x86_64-linux-gnu -lrdkafka
linkerOpts.mingw = -Lpackages/librdkafka.redist.1.9.2/build/native/lib/win/x64/win-x64-
Release/v142 -llibrdkafka -v
```

Put dependency in proper source set

```
sourceSets {
    val commonMain by getting {
        dependencies {
            implementation("com.icemachined:kafka-client:0.2.0")
            implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.6.1-native-mt")
        }
    }
    val nativeMain by creating {
        dependsOn(commonMain)
    }
    nativeTarget.let {
        getByName("${it.name}Main").dependsOn(nativeMain)
    }
}
```

Между мировосприятием и действительностью часто существует болезненное несоответствие.

Franz Kafka

Magic of linking

- On windows depends on which compiler and options are used
- By default target env is mingw64 which use cmake
- If you build by visual studio it use ms build tools cl compiler, you get linker error

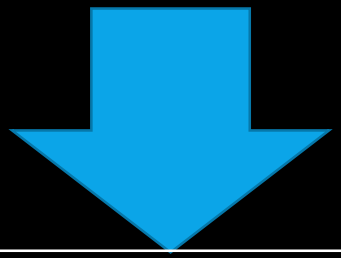
```
lld-link: error: undefined symbol: __declspec(dllimport) rd_kafka_err2str
….
Execution failed for task ':kafka-client-test:linkDebugExecutableMingwX64'.
```

C function with vararg

```
RD_EXPORT
rd_kafka_resp_err_t rd_kafka_producev(rd_kafka_t *rk, ...);
```

C function with vararg

```
RD_EXPORT
rd_kafka_resp_err_t rd_kafka_producev(rd_kafka_t *rk, ...);
```



```
@kotlinx.cinterop.internal.CCall public external expect
fun rd_kafka_producev(
rk: kotlinx.cinterop.CValuesRef<librdkafka.rd_kafka_t /* =
cnames.structs.rd_kafka_s */>?,
vararg variadicArguments: kotlin.Any?):
librdkafka.rd_kafka_resp_err_t /* = kotlin.Int */ { /* compiled code
*/ }
```

Kafka send call example from C

```
err = rd_kafka_producev(
    /* Producer handle */
    rk,
    /* Topic name */
    RD_KAFKA_V_TOPIC(topic),
    /* Make a copy of the payload. */
    RD_KAFKA_V_MSGFLAGS(RD_KAFKA_MSG_F_COPY),
    /* Message value and length */
    RD_KAFKA_V_VALUE(buf, len),
    /* Per-Message opaque, provided in
     * delivery report callback as
     * msg_opaque. */
    RD_KAFKA_V_OPAQUE(NULL),
    /* End sentinel */
    RD_KAFKA_V_END);
```

Kafka send call example from C

```
err = rd_kafka_producev(
    /* Producer handle */
    rk,
    /* Topic name */
    RD_KAFKA_V_TOPIC(topic),
    /* Make a copy of the payload. */
    RD_KAFKA_V_MSGFLAGS(RD_KAFKA_MSG_F_COPY),
    /* Message value and length */
    RD_KAFKA_V_VALUE(buf, len),
    /* Per-Message opaque, provided in
     * delivery report callback as
     * msg_opaque. */
    RD_KAFKA_V_OPAQUE(NULL),
    /* End sentinel */
    RD_KAFKA_V_END);
```

```
#define RD_KAFKA_V_VALUE(VALUE, LEN)                    \
    _LRK_TYPECHECK2(RD_KAFKA_VTYPE_VALUE, void *, VALUE, size_t, LEN),    \
      (void *)VALUE, (size_t)LEN
```

Kafka send call example from C

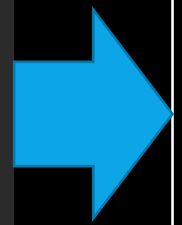
```
err = rd_kafka_producev(
    /* Producer handle */
    rk,
    /* Topic name */
    RD_KAFKA_VTYPE_TOPIC, (const char*)topic,
    /* Make a copy of the payload. */
    RD_KAFKA_VTYPE_MSGFLAGS, (int)RD_KAFKA_MSG_F_COPY,
    /* Message value and length */
    RD_KAFKA_VTYPE_VALUE, (void *)buf, (size_t)len,
    /* Per-Message opaque, provided in
     * delivery report callback as
     * msg_opaque. */
    RD_KAFKA_VTYPE_OPAQUE, (void *)NULL,
    /* End sentinel */
    RD_KAFKA_VTYPE_END);
```

Translate call from C to Kotlin

```
err = rd_kafka_producev(
    rk,
    RD_KAFKA_VTYPE_TOPIC,
    (const char*)topic,
    RD_KAFKA_VTYPE_MSGFLAGS,
    (int)RD_KAFKA_MSG_F_COPY,
    RD_KAFKA_VTYPE_VALUE,
    (void *)buf, (size_t)len,
    RD_KAFKA_VTYPE_OPAQUE,
    (void *)NULL,
    RD_KAFKA_VTYPE_END);
```

Translate call from C to Kotlin

```
err = rd_kafka_producev(
    rk,
    RD_KAFKA_VTYPE_TOPIC,
    (const char*)topic,
    RD_KAFKA_VTYPE_MSGFLAGS,
    (int)RD_KAFKA_MSG_F_COPY,
    RD_KAFKA_VTYPE_VALUE,
    (void *)buf, (size_t)len,
    RD_KAFKA_VTYPE_OPAQUE,
    (void *)NULL,
    RD_KAFKA_VTYPE_END);
```



```
val payload = "some text"
val err = rd_kafka_producev(
    producerHandle,
    RD_KAFKA_VTYPE_TOPIC,
    topic.cstr,
    RD_KAFKA_VTYPE_MSGFLAGS,
    RD_KAFKA_MSG_F_COPY,
    RD_KAFKA_VTYPE_VALUE,
    payload.cstr,
    payload.cstr.size,
    RD_KAFKA_VTYPE_OPAQUE,
    null,
    RD_KAFKA_V_END)
```

Try to call it directly

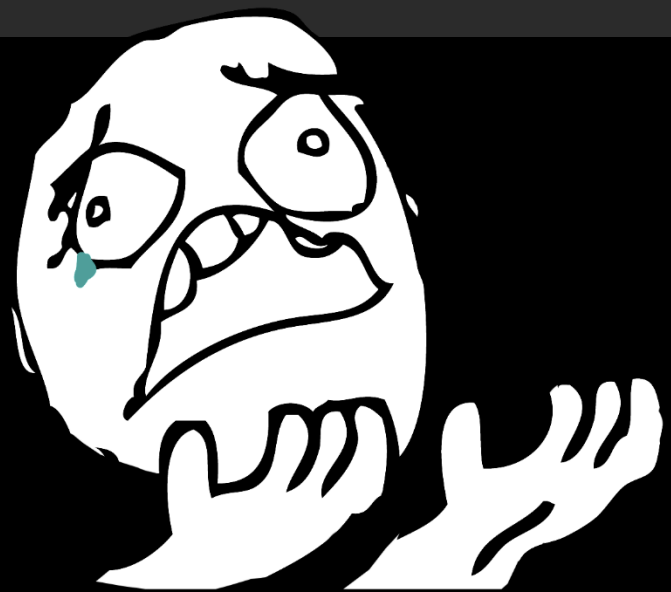
```
val payload = "some text"
rd_kafka_producev(
    producerHandle,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_TOPIC, record.topic.cstr,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_MSGFLAGS, RD_KAFKA_MSG_F_COPY,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_VALUE, payload.cstr, payload.cstr.size,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_OPAQUE, null,
    RD_KAFKA_V_END
)
```

Try to call it directly

```
val payload = "some text"
rd_kafka_producev(
    producerHandle
                                    A_VTYPE_TOPIC, record.topic.cstr,
                                    A_VTYPE_MSGFLAGS, RD_KAFKA_MSG_F_COPY,
                                    A_VTYPE_VALUE, payload.cstr, payload.cstr.size,
                                    A_VTYPE_OPAQUE, null,
```

Try to call it directly

```
val payload = "some text"
rd_kafka_producev(
    producerHandle,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_TOPIC, record.topic.cstr,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_MSGFLAGS, RD_KAFKA_MSG_F_COPY,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_VALUE, payload.cstr, payload.cstr.size,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_OPAQUE, null,
    RD_KAFKA_V_END
)
```



Create arguments builder

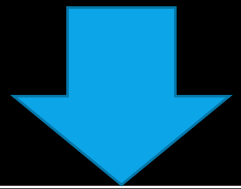
```
class ProducerArguments {
    private val variadicArgumentsList = MutableList<Any?>(0) {}

    fun addValue(pValuePointer: CPointer<ByteVarOf<Byte>>?, valueSize: size_t): ProducerArguments {
        variadicArgumentsList.addAll(listOf(RD_KAFKA_VTYPE_VALUE, pValuePointer, valueSize))
        return this
    }

    fun end(): Array<Any?> {
        variadicArgumentsList.add(RD_KAFKA_V_END)
        return variadicArgumentsList.toTypedArray()
    }
}
```

Apply arguments builder

```
val pinnedValue = value?.pin()
try {
    val valuePointer = pinnedValue?.addressOf(0)
    val variadicArgumentsArray = ProducerArguments()
        .addValue(valuePointer, valueSize)
        .end()
    rd_kafka_producev(producerHandle, *variadicArgumentsArray)
} finally {
    pinnedValue?.unpin()
}
```

Apply arguments builder

```
val pinnedValue = value?.pin()
try {
    val valuePointer = pinnedValue?.addressOf(0)
    val variadicArgumentsArray = ProducerArguments()
        .addValue(valuePointer, valueSize)
        .end()
    rd_kafka_producev(producerHandle, *variadicArgumentsArray)
} finally {
    pinnedValue?.unpin()
}
```



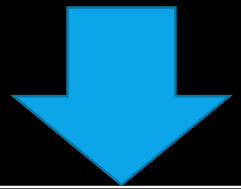
ERROR: When calling variadic C functions spread operator is supported only for *arrayOf(...)

Apply arguments builder

```
val pinnedValue = value?.pin()
try {
    val valuePointer = pinnedValue?.addressOf(0)
    val variadicArgumentsArray = ProducerArguments()
        .addValue(valuePointer, valueSize)
        .end()
    rd_kafka_producev(producerHandle,*arrayOf(*variadicArgumentsArray))
} finally {
    pinnedValue?.unpin()
}
```

Apply arguments builder

```
val pinnedValue = value?.pin()
try {
    val valuePointer = pinnedValue?.addressOf(0)
    val variadicArgumentsArray = ProducerArguments()
        .addValue(valuePointer, valueSize)
        .end()
    rd_kafka_producev(producerHandle,*arrayOf(*variadicArgumentsArray))
} finally {
    pinnedValue?.unpin()
}
```



ERROR: When calling variadic C functions spread operator is supported only for *arrayOf(...)

Apply arguments builder

```
val pinnedValue = value?.pin()
try {
    val valuePointer = pinnedValue?.addressOf(0)
    val variadicArgumentsArray = ProducerArguments()
        .addValue(valuePointer, valueSize)
        .end()
    rd_kafka_producev(producerHandle,*arrayOf(*variadicArgumentsArray))
} finally {
    dValue?.unpin()
```

g variadic C functions spread operator is supported only for *arrayOf(...)

Add wrapper to c-interop

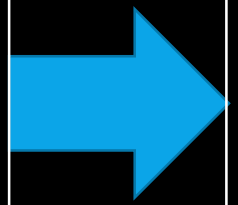
```
---

rd_kafka_resp_err_t kafka_send( rd_kafka_t *rk,
                                const char *topic,
                                int32_t partition,
                                int msgflags,
                                void *key, size_t keyLen,
                                void *value, size_t valueLen,
                                rd_kafka_headers_t *hdrs,
                                void *opaque) {
    return rd_kafka_producev( rk,
                RD_KAFKA_V_TOPIC(topic),
                RD_KAFKA_V_PARTITION(partition),
                RD_KAFKA_V_MSGFLAGS(msgflags),
                RD_KAFKA_V_KEY(key, keyLen),
                RD_KAFKA_V_VALUE(value, valueLen),
                RD_KAFKA_V_HEADERS(hdrs),
                RD_KAFKA_V_OPAQUE(opaque),
                RD_KAFKA_V_END );
}
```

Add wrapper to c-interop

```
---

rd_kafka_resp_err_t
kafka_send(
    rd_kafka_t *rk,
    const char *topic,
    int32_t partition,
    int msgflags,
    void *key,
    size_t keyLen,
    void *value,
    size_t valueLen,
    rd_kafka_headers_t *hdrs,
    void *opaque) {
```

Add wrapper to c-interop

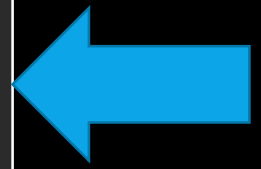
```
---

rd_kafka_resp_err_t
kafka_send(
    rd_kafka_t *rk,
    const char *topic,
    int32_t partition,
    int msgflags,
    void *key,
    size_t keyLen,
    void *value,
    size_t valueLen,
    rd_kafka_headers_t *hdrs,
    void *opaque) {
```



```
@kotlinx.cinterop.internal.CCall
public external expect
fun kafka_send(
rk: CValuesRef<librdkafka.rd_kafka_t>?,
topic: kotlin.String?,
partition: platform.posix.int32_t,
msgflags: kotlin.Int,
key: CValuesRef<*>?,
keyLen: platform.posix.size_t,
value: CValuesRef<*>?,
valueLen: platform.posix.size_t,
hdrs: CValuesRef<rd_kafka_headers_t>?,
opaque: CValuesRef<*>?):
librdkafka.rd_kafka_resp_err_t {
```

Add wrapper to c-interop

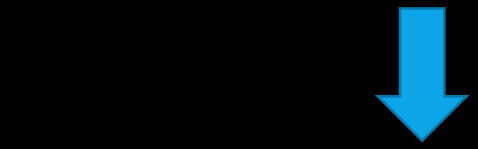
```
val err = kafka_send(
    producerHandle,
    topic,
    RD_KAFKA_PARTITION_UA,
    RD_KAFKA_MSG_F_COPY,
    keyPointer,
    keySize,
    valuePointer,
    valueSize,
    headersPointer,
    flowPointer
)
```



```
@kotlinx.cinterop.internal.CCall
public external expect
fun kafka_send(
rk: CValuesRef<librdkafka.rd_kafka_t>?,
topic: kotlin.String?,
partition: platform.posix.int32_t,
msgflags: kotlin.Int,
key: CValuesRef<*>?,
keyLen: platform.posix.size_t,
value: CValuesRef<*>?,
valueLen: platform.posix.size_t,
hdrs: CValuesRef<rd_kafka_headers_t>?,
opaque: CValuesRef<*>?):
librdkafka.rd_kafka_resp_err_t {
```

Try to call it directly again

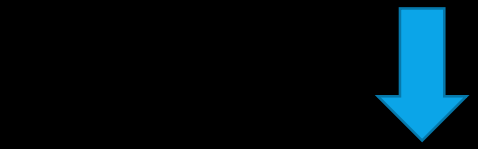
```
val payload = "some text"
rd_kafka_producev(
    producerHandle,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_TOPIC, record.topic.cstr,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_MSGFLAGS, RD_KAFKA_MSG_F_COPY,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_VALUE, payload.cstr, payload.cstr.size,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_OPAQUE, null,
    RD_KAFKA_V_END
)
```



```
val payload = "some text"
val err = kafka_send(
    producerHandle,
    topic,RD_KAFKA_PARTITION_UA,RD_KAFKA_MSG_F_COPY,
    null, 0, valuePointer, payload.cstr, payload.cstr.size
    null, null
)
```

Try to call it directly again

```
val payload = "some text"
rd_kafka_producev(
    producerHandle,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_TOPIC, record.topic.cstr,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_MSGFLAGS, RD_KAFKA_MSG_F_COPY,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_VALUE, payload.cstr, payload.cstr.size,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_OPAQUE, null,
    RD_KAFKA_V_END
)
```



```
val payload = "some text"
val err = kafka_send(
    producerHandle,
    topic,RD_KAFKA_PARTITION_UA,RD_KAFKA_MSG_F_COPY,
    null, 0, valuePointer, payload.cstr, payload.cstr.size.convert()
    null, null
)
```

Try to call it directly again

```
val payload = "some text"
rd_kafka_producev(
    producerHandle,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_TOPIC, record.topic.cstr,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_MSGFLAGS, RD_KAFKA_MSG_F_COPY,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_VALUE, payload.cstr, payload.cstr.size,
    rd_kafka_vtype_t.RD_KAFKA_VTYPE_OPAQUE, null,
    RD_KAFKA_V_END
)
```

```
val payload = "some text"
val err = kafka_send(
    producerHandle,
    topic,RD_KAFKA_PARTITION_UA,RD_KAFKA_MSG_F_COPY,
    null, 0, valuePointer, payload.cstr, payload.cstr.size.convert()
    null, null
```

see KafkaProducer on https://github.com/icemachined/kafka-kotlin-native

Pass value by reference from C

```
size_t idx = 0;
const char *name;
const void *val;
size_t size;
rd_kafka_header_get_all(hdrs, idx++, &name, &val, &size))
```

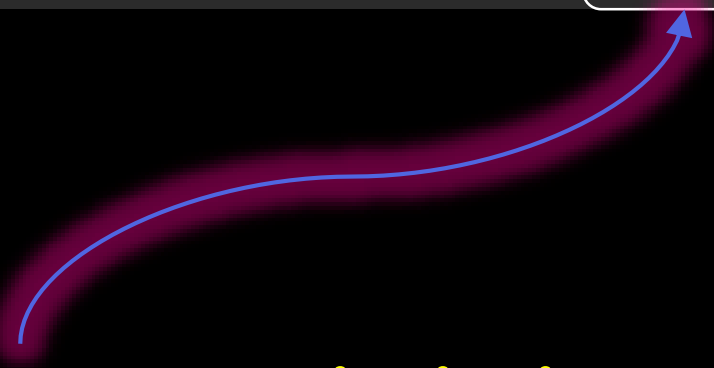
Pass value by reference from C

```
size_t idx = 0;
const char *name;
const void *val;
size_t size;
rd_kafka_header_get_all(hdrs, idx++, &name, &val, &size))
```

References to pointers

Pass value by reference from C

```
size_t idx = 0;
const char *name;
const void *val;
size_t size;
rd_kafka_header_get_all(hdrs, idx++, &name, &val, &size))
```



Reference to primitive type

How c-interop looks like

```
RD_EXPORT rd_kafka_resp_err_t
rd_kafka_header_get_all(
   const rd_kafka_headers_t *hdrs,
   size_t idx,
   const char **namep,
   const void **valuep,
   size_t *sizep);
```

How c-interop looks like

```
RD_EXPORT rd_kafka_resp_err_t
rd_kafka_header_get_all(
   const rd_kafka_headers_t *hdrs,
   size_t idx,
   const char **namep,
   const void **valuep,
   size_t *sizep);
```

```
@kotlinx.cinterop.internal.CCall
public external expect
fun rd_kafka_header_get_all(
hdrs:CValuesRef<rd_kafka_headers_t>?,
idx: platform.posix.size_t,
namep: CValuesRef<CPointerVar<ByteVar>>?,
valuep: CValuesRef<COpaquePointerVar>?,
sizep: CValuesRef<platform.posix.size_tVar>?):
librdkafka.rd_kafka_resp_err_t {
```

nativePlacement in memScope

```
memScoped {
    val headersPointer = allocPointerTo<rd_kafka_headers_t>()

    if (rd_kafka_message_headers(rkmessage, headersPointer.ptr) == 0) {
```

nativePlacement in memScope

```
memScoped {
    val headersPointer = allocPointerTo<rd_kafka_headers_t>()

    if (rd_kafka_message_headers(rkmessage, headersPointer.ptr) == 0) {
        val valRef: COpaquePointerVar = alloc()
        val sizeRef: size_tVar = alloc()
        val nameRef: CPointerVar<ByteVar> = alloc()
        var idx = 0
        while (rd_kafka_header_get_all(
            headersPointer.value,
            idx.convert(),
            nameRef.ptr,
            valRef.ptr,
            sizeRef.ptr
        ) == 0
```

nativePlacement in memScope

```
memScoped {
    val headersPointer = allocPointerTo<rd_kafka_headers_t>()

    if (rd_kafka_message_headers(rkmessage, headersPointer.ptr) == 0) {
        val valRef: COpaquePointerVar = alloc()
        val sizeRef: size_tVar = alloc()
        val nameRef: CPointerVar<ByteVar> = alloc()
        var idx = 0
        while (rd_kafka_header_get_all(
            headersPointer.value,
            idx.convert(),
            nameRef.ptr,
            valRef.ptr,
            sizeRef.ptr
        ) == 0
```

nativePlacement in memScope

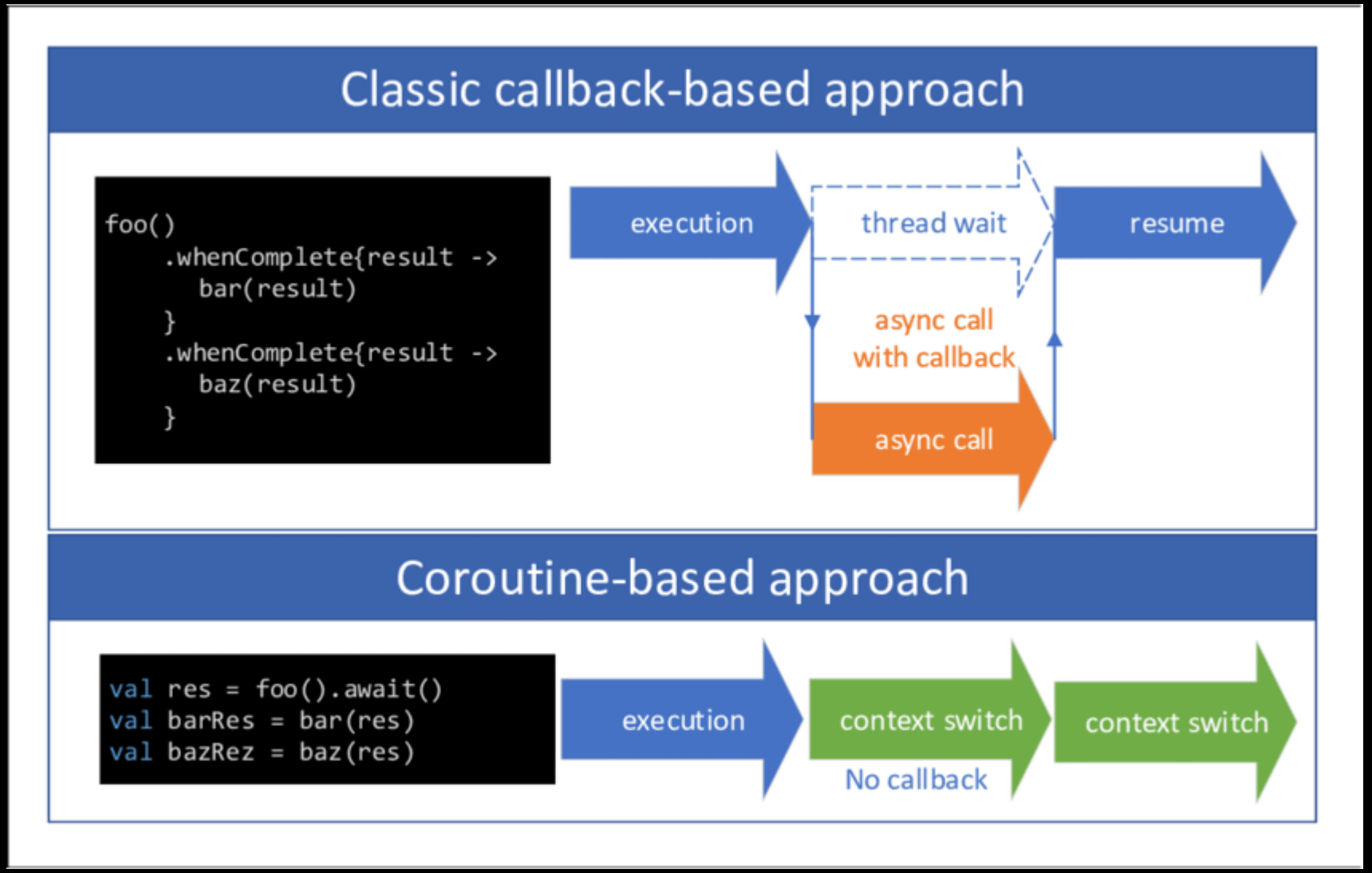
```
memScoped {
    val headersPointer = allocPointerTo<rd_kafka_headers_t>()

    if (rd_kafka_message_headers(rkmessage, headersPointer.ptr) == 0) {
        val valRef: COpaquePointerVar = alloc()
        val sizeRef: size_tVar = alloc()
        val nameRef = allocPointerTo <ByteVar> ()
        var idx = 0
        while (rd_kafka_header_get_all(
            headersPointer.value,
            idx.convert(),
            nameRef.ptr,
            valRef.ptr,
            sizeRef.ptr
        ) == 0
```

nativePlacement in memScope

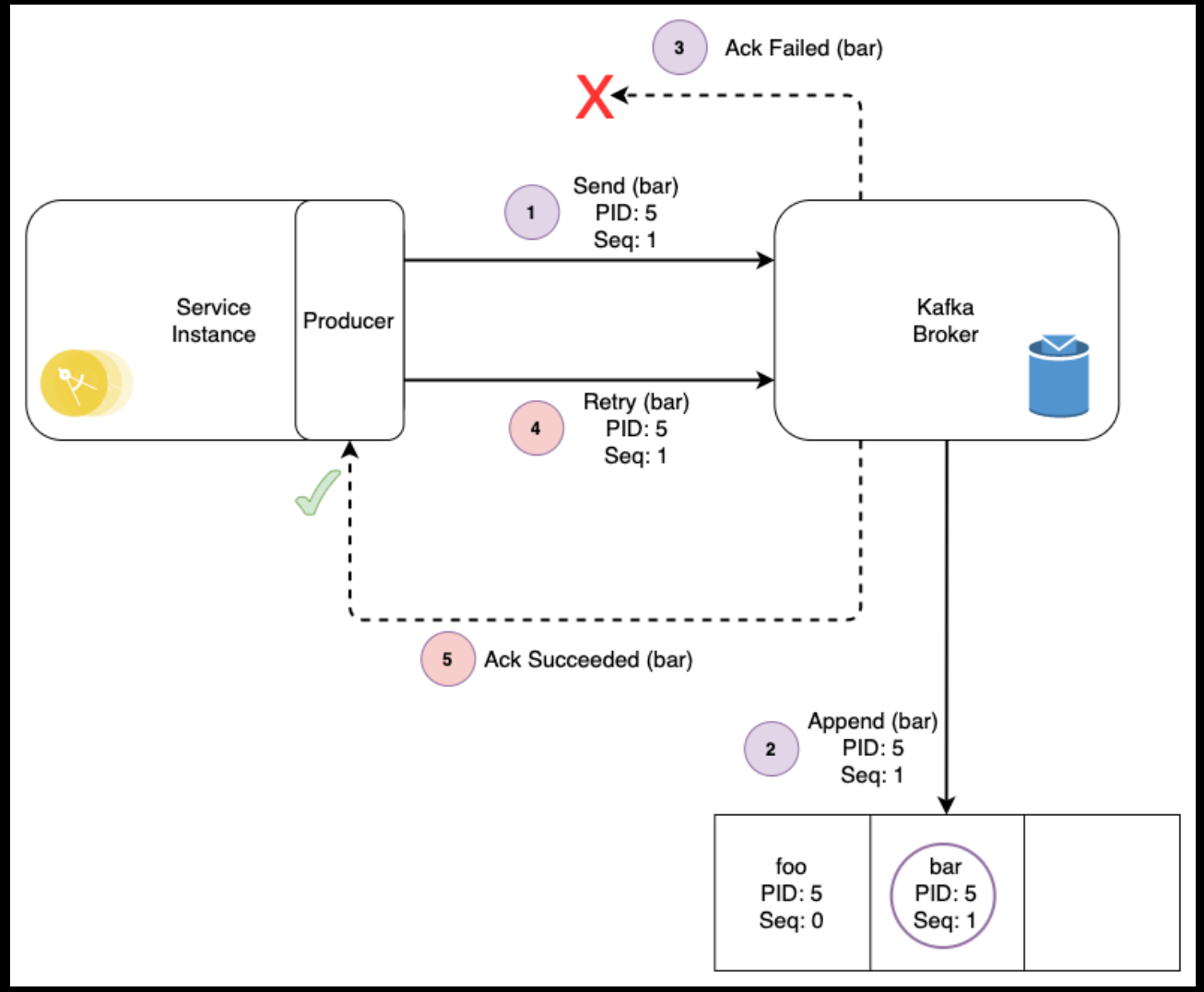
```
headers.add(
    RecordHeader(
        nameRef.value?.toKString(),
        valRef.value?.readBytes(sizeRef.value.toInt())
    )
)
```

see KafkaConsumer on https://github.com/icemachined/kafka-kotlin-native

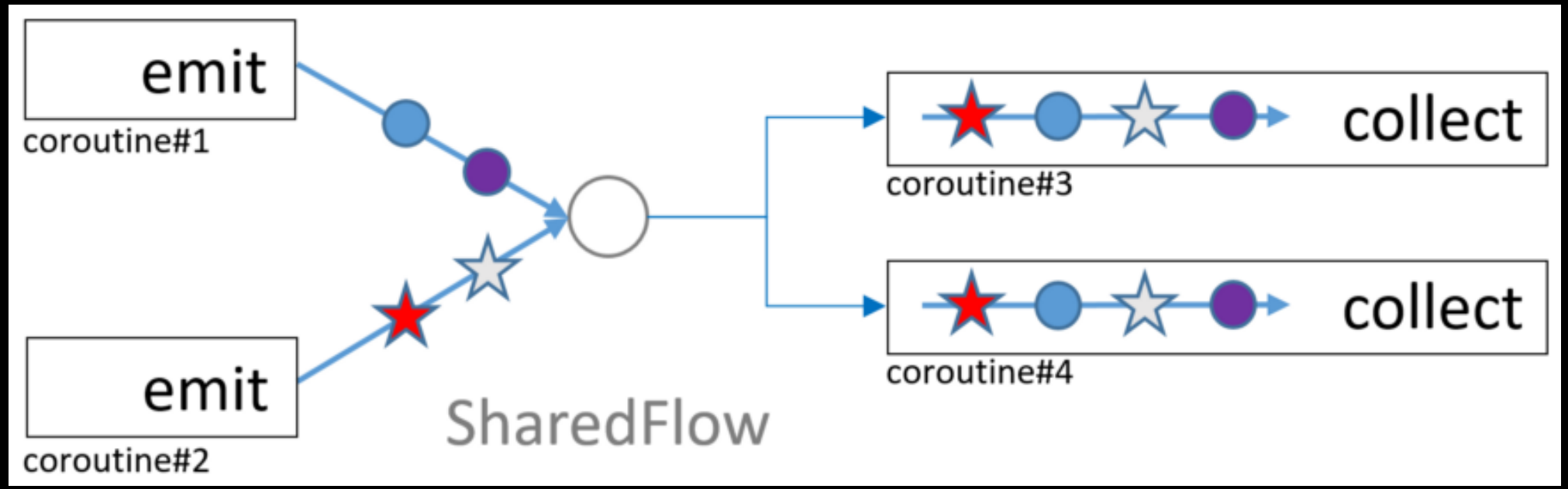
Notify from callback



Kafka send with retry



Shared Flow



https://elizarov.medium.com/shared-flows-broadcast-channels-899b675e805c

We can apply flow operations

```
val flow = producer.send(
    ProducerRecord(
        "test-topic", "test value", "test key"
    )
)

flow.first { it.isOk }
```

see KafkaProducer on https://github.com/icemachined/kafka-kotlin-native

Polling cycles

- Kafka use pull delivery strategy to ensure scalability and throughtput
- There are 2 type of polling cycles on client
  1. Producer polling cycle to dispatch delivery acks
  2. Consumer pollying cycle to dispatch uncommitted records from topic

Polling Cycle: Worker

“Instead of threads, Kotlin/Native runtime offers the concept of workers concurrently executed control flow streams with an associated request queue. Workers are very similar to the actors in the Actor Model. ”

https://kotlinlang.org/docs/native-immutability.html#workers

Polling Cycle: Worker

“Instead of threads, Kotlin/Native runtime offers the concept of workers concurrently executed control flow streams with an associated request queue. Workers are very similar to the actors in the Actor Model. ”

https://kotlinlang.org/docs/native-immutability.html#workers

Worker under the hood

`Worker.start`

Worker under the hood

```
Worker.start
```

```
Kotlin_Worker_startInternal
```

Worker under the hood

`Worker.start`

`Kotlin_Worker_startInternal`

```
void Worker::startEventLoop() {
  kotlin::ThreadStateGuard guard(ThreadState::kNative);
  pthread_create(&thread_, nullptr, workerRoutine, this);
}
```

Worker under the hood

```
Worker.start
```

```
Kotlin_Worker_startInternal
```

```
void Worker::startEventLoop() {
  kotlin::ThreadStateGuard guard(ThreadState::kNative);
  pthread_create(&thread_, nullptr, workerRoutine, this);
}
```

Worker under the hood

`Worker.start`

`Kotlin_Worker_startInternal`

```
void Worker::startEventLoop() {
  kotlin::ThreadStateGuard guard(ThreadState::kNative);
  pthread_create(&thread_, nullptr, workerRoutine, this);
}
```

Polling cycle: Worker + coroutine

```
val kafkaPollingJobFuture = worker.execute(TransferMode.SAFE,    {
        KafkaConsumerJob(config, consumer,
isConsumerPollingActive).freeze()  }) {
    it.pollingCycle()
}
```

Polling cycle: Worker + coroutine

```
val kafkaPollingJobFuture = worker.execute(TransferMode.SAFE,    {
        KafkaConsumerJob(config, consumer,
isConsumerPollingActive).freeze()  }) {
    it.pollingCycle()
}
```

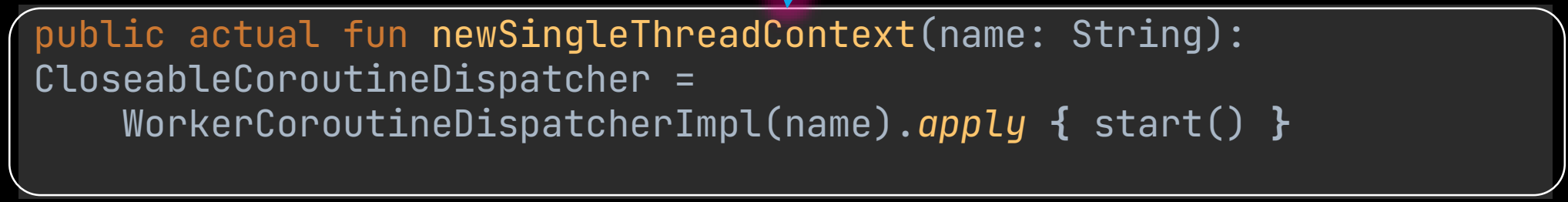
```
fun pollingCycle() {
    runBlocking {
        launch {
            while(isActive) {
                val records =
consumer.poll()
                delay(timeout)
            } finally {
                consumer.close()
            }
        }
    }
}
```

Dispatchers.Default under the hood

```
private fun getOrCreateDefaultDispatcher() = lock.withLock {
    _delegate.value ?: newSingleThreadContext("DefaultDispatcher").also
{ _delegate.value = it }
}
```

Dispatchers.Default under the hood

```
private fun getOrCreateDefaultDispatcher() = lock.withLock {
    _delegate.value ?: newSingleThreadContext("DefaultDispatcher").also
{ _delegate.value = it }
}
```



```
public actual fun newSingleThreadContext(name: String):
CloseableCoroutineDispatcher =
    WorkerCoroutineDispatcherImpl(name).apply { start() }
```

SingleThreadContext under the hood

```
private class WorkerCoroutineDispatcherImpl(name: String) :
CloseableCoroutineDispatcher(), ThreadBoundInterceptor, Delay {
    override val worker = Worker.start(name = name)
    private val isClosed = atomic(false)

    init { freeze() }

    fun start() {
        worker.execute {
            workerMain {
                runEventLoop(ThreadLocalEventLoop.eventLoop) { isClosed.value }
            }
        }
    }
```

Producer polling job

```
class KafkaProducerPollingJob(
    private val producerHandle: CPointer<rd_kafka_t>,
    private val kafkaPollingIntervalMs: Long,
    private val coroutineDispatcher: CoroutineDispatcher =
newSingleThreadContext("")
) {
    suspend fun pollingCycle() =
        withContext(coroutineDispatcher) {
            launch {
                try {
                    while (isActive) {
                        delay(kafkaPollingIntervalMs)
                        rd_kafka_poll (producerHandle, 0 /* non-blocking*/)
                        println("producer poll happened")
                    }
                } catch (e: CancellationException) {
                    println("poll cancelled it's ok")
                } catch (e: Throwable) {
                    println("Unexpected exception in kafka polling job:")
                } finally {
                    println("exiting poll ")
                }
```

Try start polling job

```
override suspend fun send(record: ProducerRecord<K, V>): SharedFlow<SendResult> {
  print("Start producer send")
    ensurePollingCycleInitialized()
```

Try start polling job

```
override suspend fun send(record: ProducerRecord<K, V>): SharedFlow<SendResult> {
  print("Start producer send")
    ensurePollingCycleInitialized()
```

```
private suspend fun ensurePollingCycleInitialized() {
    if (!this::producerCoroutineJob.isInitialized) {
        producerCoroutineJob = producerPollingJob.value.pollingCycle()
    }
}
```

Try start polling job

```
override suspend fun send(record: ProducerRecord<K, V>): SharedFlow<SendResult> {
  print("Start producer send")
    ensurePollingCycleInitialized()
```

```
private suspend fun ensurePollingCycleInitialized() {
    if (!this::producerCoroutineJob.isInitialized) {
        producerCoroutineJob = producerPollingJob.value.pollingCycle()
    }
}
```

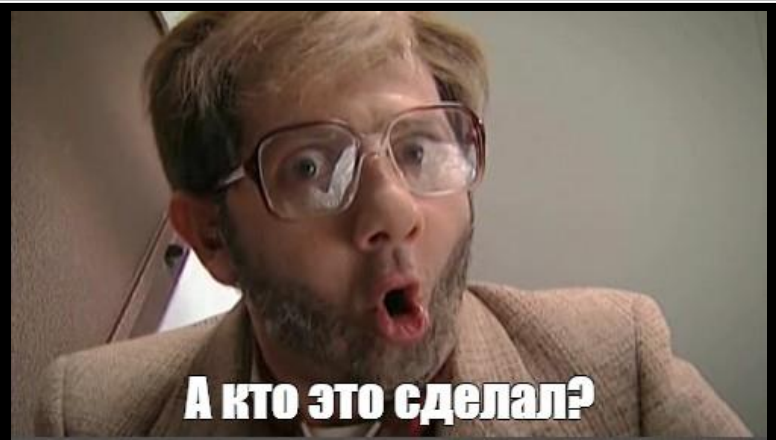
Start producer send
producer poll happened …

Try start polling job

```
override suspend fun send(record: ProducerRecord<K, V>): SharedFlow<SendResult> {
  print("Start producer send")
    ensurePollingCycleInitialized()
```

```
private suspend fun ensurePollingCycleInitialized() {
    if (!this::producerCoroutineJob.isInitialized) {
        producerCoroutineJob = producerPollingJob.value.pollingCycle()
    }
}
```

Start producer send
producer poll happened …



Try start polling job

```
override suspend fun send(record: ProducerRecord<K, V>): SharedFlow<SendResult> {
  print("Start producer send")
    ensurePollingCycleInitialized()
```

```
private suspend fun ensurePollingCycleInitialized() {
    if (!this::producerCoroutineJob.isInitialized) {
        producerCoroutineJob = producerPollingJob.value.pollingCycle()
    }
}
```

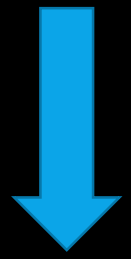
Start producer send
producer poll happened …

What happened with coroutine

```
suspend fun pollingCycle() =
    withContext(coroutineDispatcher) {
```

What happened with coroutine

```
suspend fun pollingCycle() =
    withContext(coroutineDispatcher) {
```



```
launch {
        try {
            while (isActive) {
```

CoroutineScope help to run parallel

CoroutineScope help to run parallel

`GlobalScope.launch`

`MainScope.launch`

`CoroutineScope(Job())`

`CoroutineScope(SupervisorJob())`

CoroutineScope help to run parallel

`GlobalScope.launch`

`MainScope.launch`

`CoroutineScope(Job())`

`CoroutineScope(SupervisorJob())`

```
private val consumerGroupScope: CoroutineScope = CoroutineScope(SupervisorJob())
```

```
CoroutineScope(
    Job(consumerGroupScope.coroutineContext.job) +
            (coroutineDispatcher ?: newSingleThreadContext("consumer-context-$clientId"))
)
```

CoroutineScope help to run parallel

`GlobalScope.launch`

`MainScope.launch`

`CoroutineScope(Job())`

`CoroutineScope(SupervisorJob())`

```
private val consumerGroupScope: CoroutineScope = CoroutineScope(SupervisorJob())
```

```
CoroutineScope(
    Job(consumerGroupScope.coroutineContext.job) +
            (coroutineDispatcher ?: newSingleThreadContext("consumer-context-$clientId"))
)
```

```
consumerGroupScope.cancel()
```

see KafkaParallelConsumerService on https://github.com/icemachined/kafka-kotlin-native


THE BEST WAY TO
EXPLAIN IT IS
TO DO IT
Alice in Wonderland

What we’ve learned

- What is Kafka client and why we need it on Kotlin native
- How Kotlin MPP project is organized
- How to cope with C functions with vararg built by templates
- How to pass value by reference from C
- How to do notify from callback using SharedFlow
- What is Worker and Dispatchers under the hood
- How coroutines work across the threads and what is CoroutineScope

Useful links

- Kafka on Kotlin Native client
  - https://github.com/icemachined/kafka-kotlin-native
- Save Cloud project
  - https://github.com/saveourtool/save-cloud
- Kotlin Native documentation
  - https://kotlinlang.org/docs/native-get-started.html
  - https://kotlinlang.org/docs/coroutines-guide.html
- Kotlin Native Sources
  - https://github.com/JetBrains/kotlin/tree/master/kotlin-native
  - https://github.com/kotlin/kotlinx.coroutines/tree/native-mt