https://jagstalk.page.link/examples

```kotlin
private suspend fun processOrders(orders: Flow<Menu.Cappuccino>,
        espressoMachine: EspressoMachine): Flow<Beverage.Cappuccino> =
    orders.map {
        val groundBeans = grindCoffeeBeans(it.beans())
        coroutineScope {
            val espressoDeferred = async {
                espressoMachine.pullEspressoShot(groundBeans)
            }
            val steamedMilkDeferred = async {
                espressoMachine.steamMilk(it.milk())
            }
            makeCappuccino(it, espressoDeferred.await(), steamedMilkDeferred.await())
        }
    }
```

```kotlin
fun main() = runBlocking {
    val orders = listOf(...)
    val ordersFlow: Flow<Menu.Cappuccino> = orders.asFlow()


    val espressoMachine = EspressoMachine(this)
    val time = measureTimeMillis {

        ...
    }
    log("time: $time ms")
    espressoMachine.destroy()
}
```

```kotlin
fun main() = runBlocking {
    val orders = listOf(...)
    val ordersFlow: Flow<Menu.Cappuccino> = orders.asFlow()

    val espressoMachine = EspressoMachine(this)
    val time = measureTimeMillis {

        ...

    }
    log("time: $time ms")
    espressoMachine.destroy()
}

private suspend fun processOrders(orders: Flow<Menu.Cappuccino>,
        espressoMachine: EspressoMachine): Flow<Beverage.Cappuccino> =
    orders.map {
        val groundBeans = grindCoffeeBeans(it.beans())
        coroutineScope {
            val espressoDeferred = async {
                espressoMachine.pullEspressoShot(groundBeans)
            }
            val steamedMilkDeferred = async {
                espressoMachine.steamMilk(it.milk())
            }
            makeCappuccino(it, espressoDeferred.await(), steamedMilkDeferred.await())
        }
    }
```

```kotlin
fun main() = runBlocking {
    val orders = listOf(...)
    val ordersFlow: Flow<Menu.Cappuccino> = orders.asFlow()

    val espressoMachine = EspressoMachine(this)
    val time = measureTimeMillis {
        coroutineScope {
            launch(CoroutineName("barista-1")) {
                processOrders(ordersChannel, espressoMachine)
            }
            launch(CoroutineName("barista-2")) {
                processOrders(ordersChannel, espressoMachine)
            }
        }
    }
    log("time: $time ms")
    espressoMachine.destroy()
}



private suspend fun processOrders(orders: Flow<Menu.Cappuccino>,
        espressoMachine: EspressoMachine): Flow<Beverage.Cappuccino> {
    ...
}
```