Miles or Kilometers?

Why Your Data Schema Should Include Units

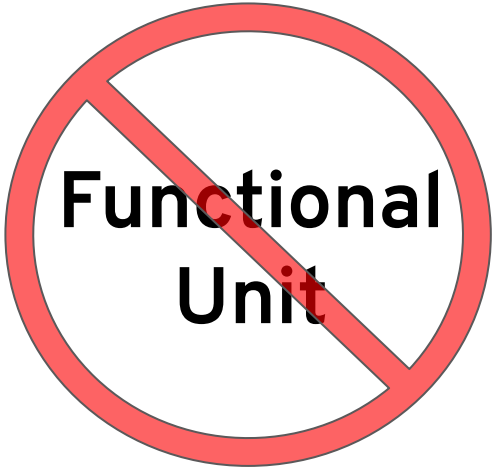
Erik Erlandson

eje@redhat.com
@manyangled

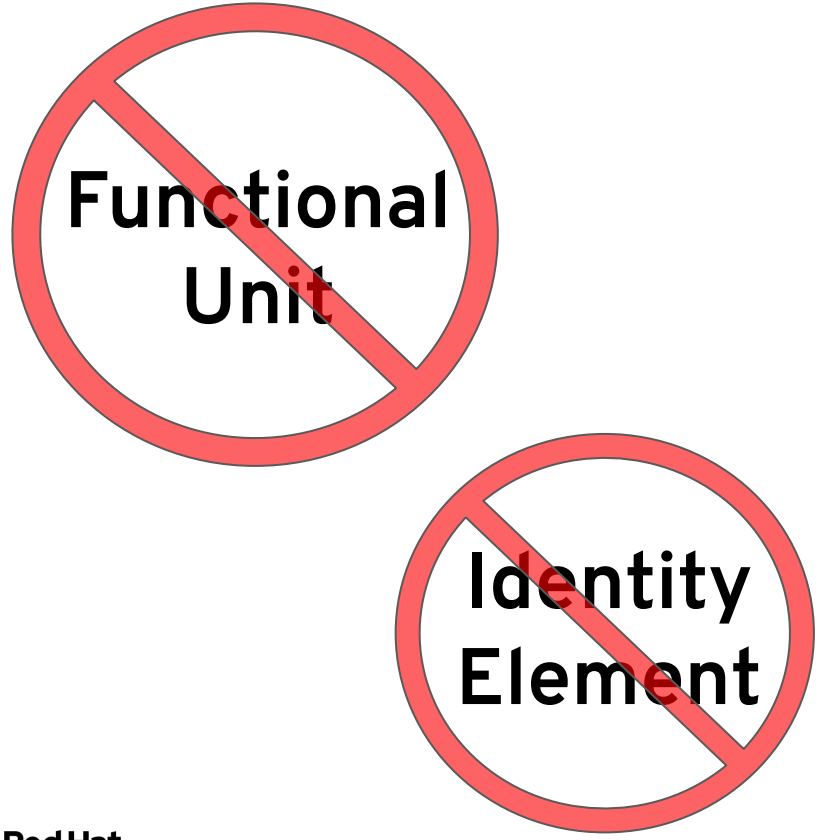
Red Hat

BERLIN BUZZWORDS 2019 JUNE 16-18

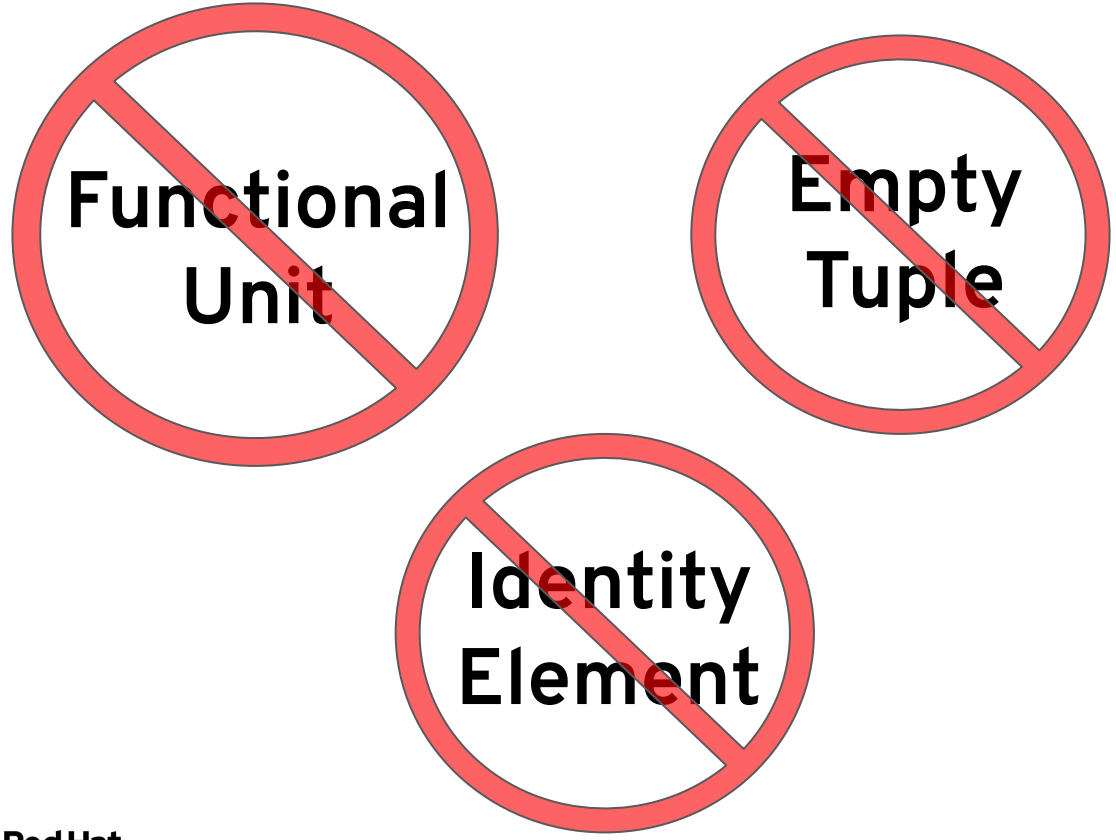
“Unit” is Overloaded



“Unit” is Overloaded



“Unit” is Overloaded



“Unit” is Overloaded

Functional Unit

Empty Tuple

Identity Element

in awe at the size of this lad.
absolute unit.

Units of Measure

Units of Measure

Units of Measure

Units from Software

latency: seconds, milliseconds

Units from Software

latency: seconds, milliseconds

memory limit: megabytes, gigabits, tebibytes

Units from Software

latency:	seconds, milliseconds
memory limit:	megabytes, gigabits, tebibytes
bandwidth:	megabytes / second, gigabits / second

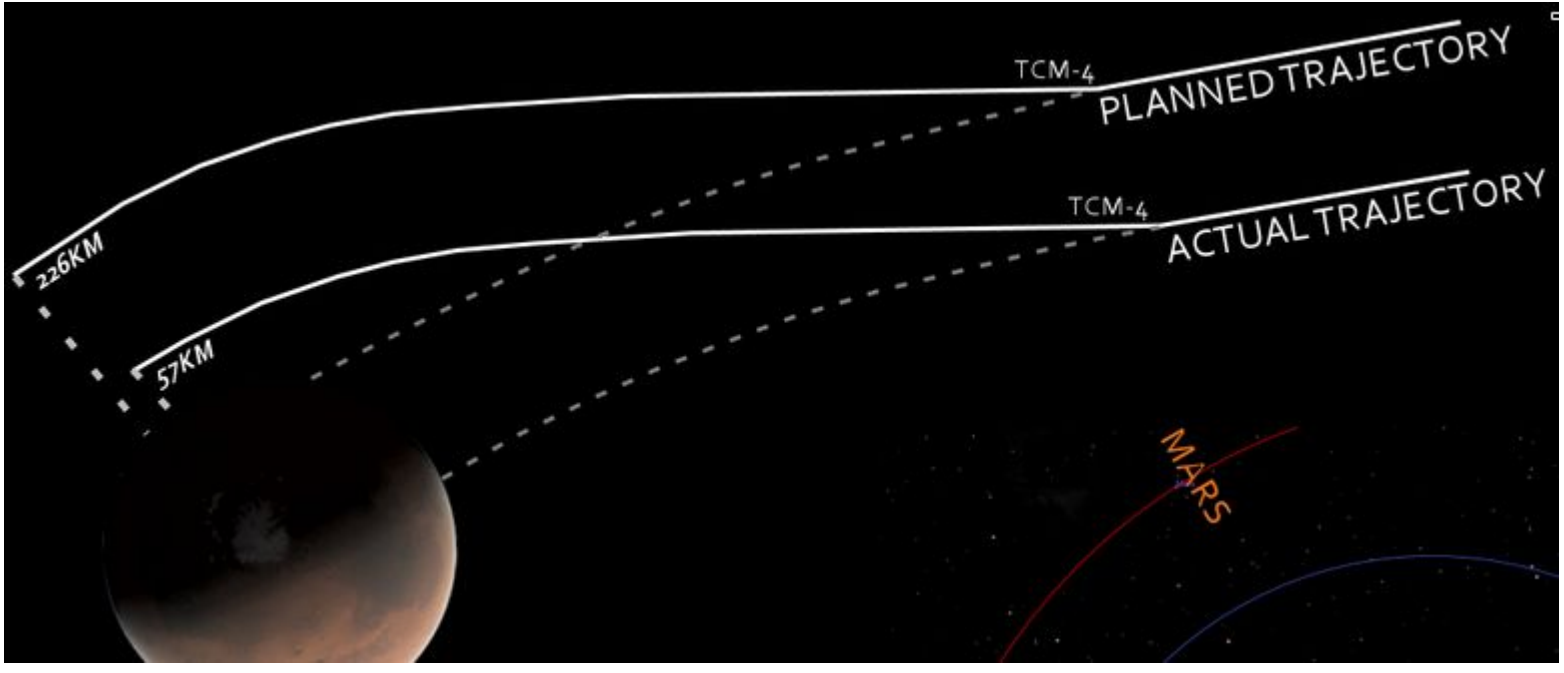
Units from Software

latency:	seconds, milliseconds
memory limit:	megabytes, gigabits, tebibytes
bandwidth:	megabytes / second, gigabits / second
throughput:	queries / second, inferences / second

Units from Software

latency:	seconds, milliseconds
memory limit:	megabytes, gigabits, tebibytes
bandwidth:	megabytes / second, gigabits / second
throughput:	queries / second, inferences / second
cluster measures:	nodes, pods / second, VCs

Unit Mistake: $327 million



Mars Climate Orbiter: lbf·s instead of N·s

“M” (1000^2) or “Mi” (1024^2) ?

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

https://github.com/apache-spark-on-k8s/spark/pull/470

“M” (1000^2) or “Mi” (1024^2) ?

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

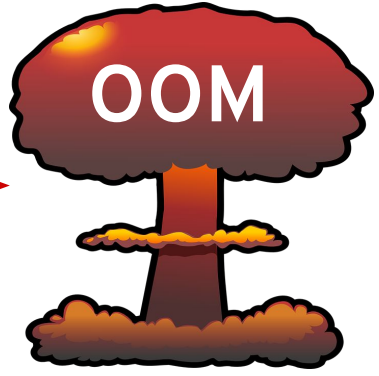
< JVM (in Mi)

https://github.com/apache-spark-on-k8s/spark/pull/470

“M” (1000^2) or “Mi” (1024^2) ?

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

< JVM (in Mi)



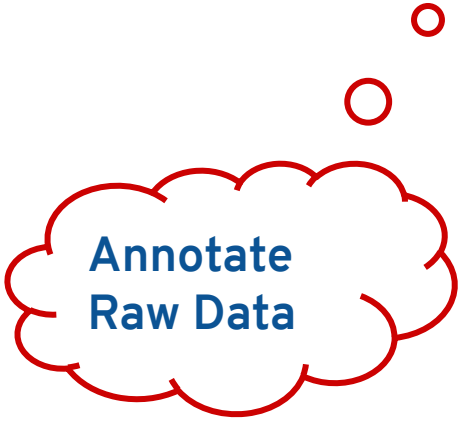
https://github.com/apache-spark-on-k8s/spark/pull/470

What Role Do Units Play?

100 milliseconds

2048 megabytes

10 gigabits / second



What Role Do Units Play?

100 milliseconds

2048 megabytes

10 gigabits / second

Define Legal Operations

Annotate Raw Data

What Role Do Units Play?

100 milliseconds

2048 megabytes

10 gigabits / second

Define Legal Operations

Annotate Raw Data

Forbid Illegal Operations

Legal

conversion

10 minutes = 600 seconds

1 gigabit / second = 7500 megabytes / minute

Legal

conversion

10 minutes = 600 seconds
1 gigabit / second = 7500 megabytes / minute

algebra

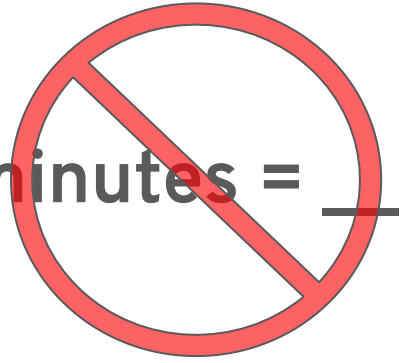
10 seconds + 1 minute = 70 seconds
(10 gigabits / second)(60 seconds) = 600 gigabits

Illegal!

conversion 10 minutes = __ bytes

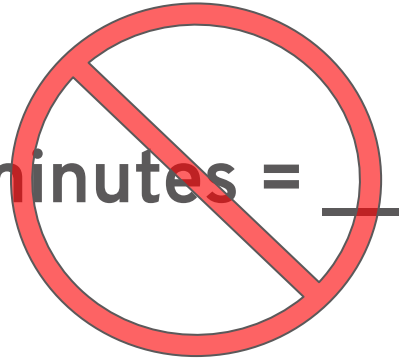
Illegal!

conversion 10 minutes = __ bytes



Illegal!

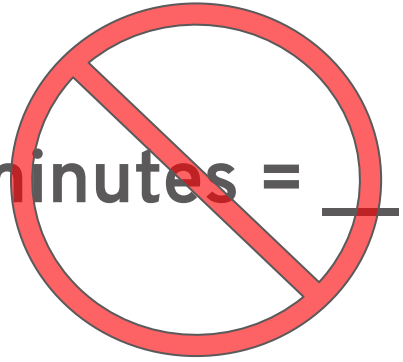
conversion 10 minutes = __ bytes



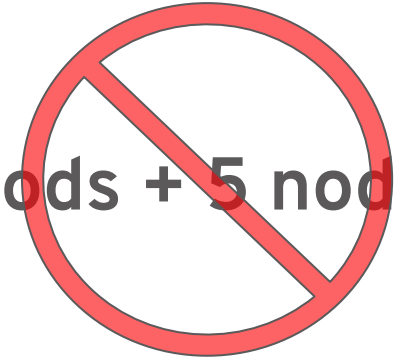
algebra 10 pods + 5 nodes

Illegal!

conversion 10 minutes = __ bytes



algebra 10 pods + 5 nodes

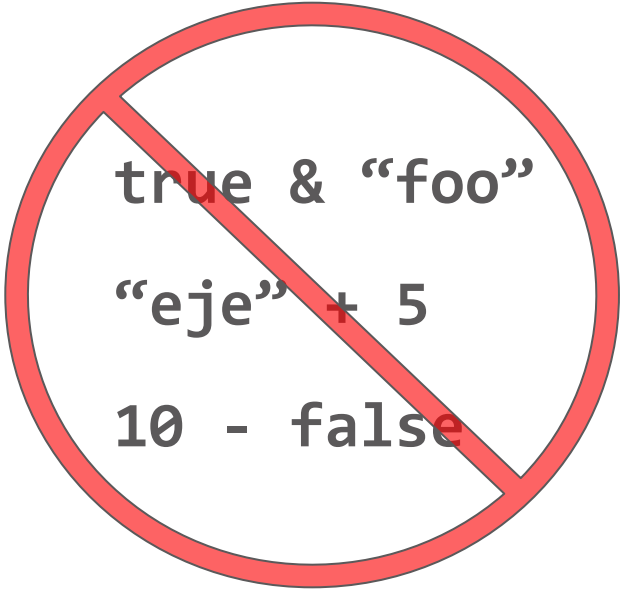


Data Types!

```
true & false

“eje” + “@redhat.com”

10 - 5
```

Data Types!

```
true & false

“eje” + “@redhat.com”

10 - 5
```



What If...

```
“eje” + 5
        ^ Error! Type Mismatch
```

What If...

```
“eje” + 5
        ^ Error! Type Mismatch
```

```
10 seconds + 5 bytes
```

What If...

```
“eje” + 5
        ^ Error! Type Mismatch


10 seconds + 5 bytes
             ^ Error! Type Mismatch
```

Find the Unit Types

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

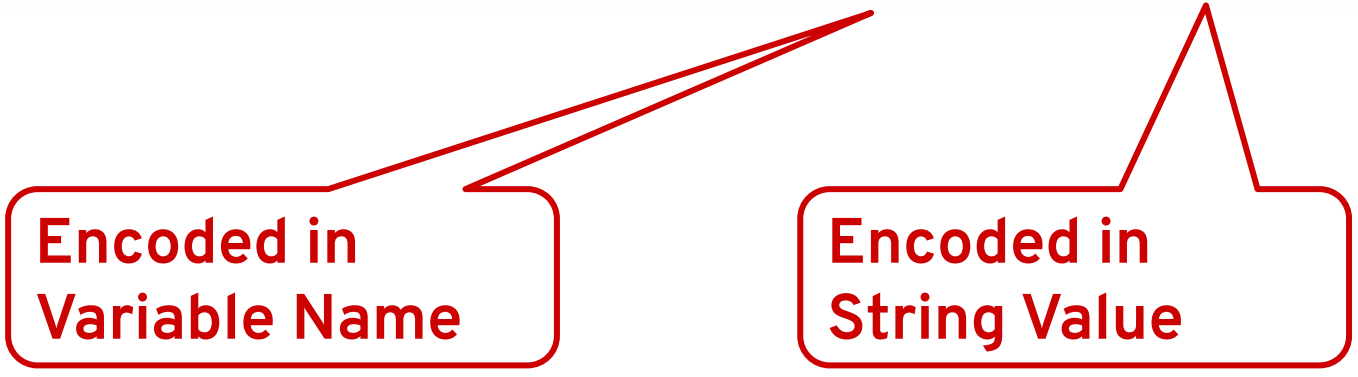
Find the Unit Types

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

Encoded in Variable Name

Find the Unit Types

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```



Units as Data Types

```
val mass = Quantity[Float, Kilogram](100.0)
```

Units as Data Types

```
val mass = Quantity[Float, Kilogram](100.0)

val duration = 30.withUnit[Second]
```

Units as Data Types

```
val mass = Quantity[Float, Kilogram](100.0)

val duration = 30.withUnit[Second]

val g = (9.8).withUnit[Meter %/ (Second %^ 2)]
```

Units as Data Types

```
val mass = Quantity[Float, Kilogram](100.0)

val duration = 30.withUnit[Second]

val g = (9.8).withUnit[Meter %/ (Second %^ 2)]

val ohms = (0.01).withUnit[
(Kilogram %* (Meter %^ 2)) %/ ((Second %^ 3) %*
(Ampere %^ 2))
]
```

Programming With Unit Types

```
def callKubeAPI(memLimit: Quantity[Int, Mega %* Byte]) =
    memLimit.show
```

Programming With Unit Types

```
def callKubeAPI(memLimit: Quantity[Int, Mega %* Byte]) =
    memLimit.show

// ....

val memRequest = 1000.withUnit[Mebi %* Byte]
```

Programming With Unit Types

```
def callKubeAPI(memLimit: Quantity[Int, Mega %* Byte]) =
    memLimit.show

// ....

val memRequest = 1000.withUnit[Mebi %* Byte]

// ....

scala> callKubeAPI(memRequest)
res0: String = 1024 MB
```

Compile-Time Checking

```
val timeout = 100.withUnit[Milli %* Second]
```

Compile-Time Checking

```
val timeout = 100.withUnit[Milli %* Second]

// ....

scala> callKubeAPI(timeout)
    error: type mismatch;
   found   : Quantity[Int,Milli %* Second]
   required: Quantity[Int,Mega %* Byte]
```

Unit Types for Configuration

```
val confTS = ConfigFactory.parseString("""
        "log-retention-time" = "24 hour"
        "log-segment-size" = "1 megabyte"
        "log-flush-interval" = "10 second"
        "log-bandwidth" = "10 megabyte / second"
      """)
```

Unit Aware Configuration

```
def sysCall(ms: Quantity[Int, Milli %* Second]) =
    ms.showFull
```

Unit Aware Configuration

```
def sysCall(ms: Quantity[Int, Milli %* Second]) =
    ms.showFull

val logRetentionTime =
    conf.getQuantity[Int, Second]("log-retention-time")
```

Unit Aware Configuration

```
def sysCall(ms: Quantity[Int, Milli %* Second]) =
    ms.showFull

val logRetentionTime =
    conf.getQuantity[Int, Second]("log-retention-time")

scala> sysCall(logRetentionTime.get)
res0: String = 86400000 millisecond
```

Unit Aware Configuration

```
def sysCall(ms: Quantity[Int, Milli %* Second]) =
    ms.showFull

val logRetentionTime =
    conf.getQuantity[Int, Second]("log-retention-time")

scala> sysCall(logRetentionTime.get)
res0: String = 86400000 millisecond

scala> conf.getQuantity[Int, Byte]("log-retention-time")
ToolBoxError: reflective compilation has failed:
```

Unit Aware Avro Schema

```
{
"type": "record",
"name": "smol",
"fields": [
    { "name": "latency", "type": "double", "unit": "second" },
    { "name": "bandwidth", "type": "double", "unit": "gigabyte / second" }
]}
```

Unit Aware Avro Schema

```
rec.putQuantity(qp)(
    "latency", 100.withUnit[Milli %* Second])

rec.putQuantity(qp)(
    "bandwidth", 1.withUnit[Tera %* Bit %/ Minute])
```

Unit Aware Avro Schema

```
rec.putQuantity(qp)(
    "latency", 100.withUnit[Milli %* Second])

rec.putQuantity(qp)(
    "bandwidth", 1.withUnit[Tera %* Bit %/ Minute])

scala> rec
res0: Record = {"latency": 0.1, "bandwidth": 2.083333}
```

Unit Aware Avro Schema

```
scala> rec.getQuantity[Double, Micro %* Second](qp)("latency").show
res0: String = 100000.0 µs

scala> rec.getQuantity[Double, Giga %* Bit %/ Minute](qp)("bandwidth").show
res1: String = 1000.0 Gb/min
```

Unit Aware Avro Schema

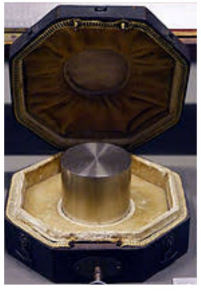
```
scala> rec.getQuantity[Double, Micro %* Second](qp)("latency").show
res0: String = 100000.0 µs

scala> rec.getQuantity[Double, Giga %* Bit %/ Minute](qp)("bandwidth").show
res1: String = 1000.0 Gb/min

scala> rec.getQuantity[Double, Byte](qp)("latency")
Exception: unit metadata "second" incompatible with "Byte"
```

More and Better Units

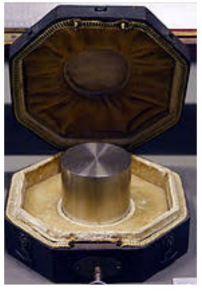
```
implicit val defineUnitPod = BaseUnit[Pod](abbv = “pod”)
```

```
implicit val defineUnitNode = BaseUnit[Node](abbv = “node”)
```

Review

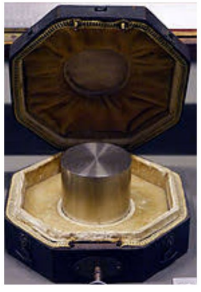


Review



```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

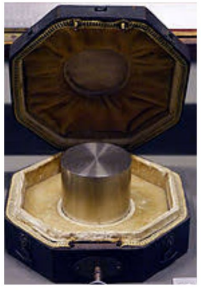
Review



```
val mass = Quantity[Float, Kilogram](100.0)

val duration = 30.withUnit[Second]

val g = (9.8).withUnit[Meter %/ (Second %^ 2)]

val ohms = (0.01).withUnit[
(Kilogram %* (Meter %^ 2)) %/ ((Second %^ 3) %*
(Ampere %^ 2))
]
```

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

Review

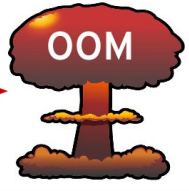


```
{
"type": "record",
"name": "smol",
"fields": [
    { "name": "latency", "type": "double", "unit": "second" },
    { "name": "bandwidth", "type": "double", "unit": "gigabyte / second" }
]}
```

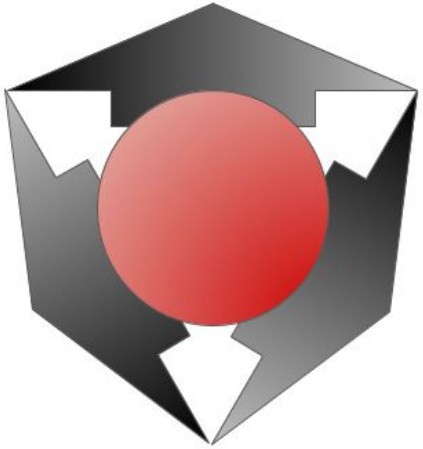
```
val mass = Quantity[Float, Kilogram](100.0)

val duration = 30.withUnit[Second]

val g = (9.8).withUnit[Meter %/ (Second %^ 2)]

val ohms = (0.01).withUnit[
(Kilogram %* (Meter %^ 2)) %/ ((Second %^ 3) %*
(Ampere %^ 2))
]
```

```
.withAmount(s"${driverMemoryMb}M")
.withAmount(s"${driverMemoryMiB}Mi")
```

< JVM (in Mi) → OOM



Explore



coulomb

Unit analysis for Scala

https://github.com/erikerlandson/coulomb#documentation

eje@redhat.com @manyangled