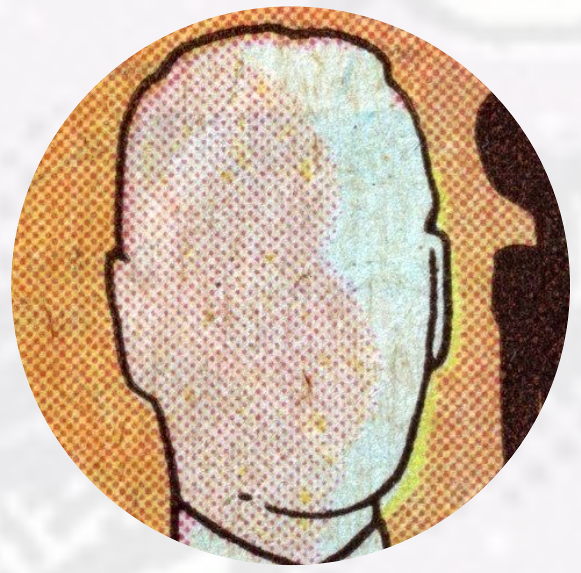
Scala in Akka

Za najzahtevnejše izzive

Oto Brglez

/me
Oto Brglez



- Engineering Manager and Architect @ GlobalWebIndex
- Contractor
- Ex: CTO, Tech lead, Engineering Lead, Senior, …
- Geekatrons Member

@otobrglez

otobrglez@gmail.com

epic.blog

Kaj je danes “težko”?

Real-time data 🐰

Big-data 🌆

Distributed systems 🗺️

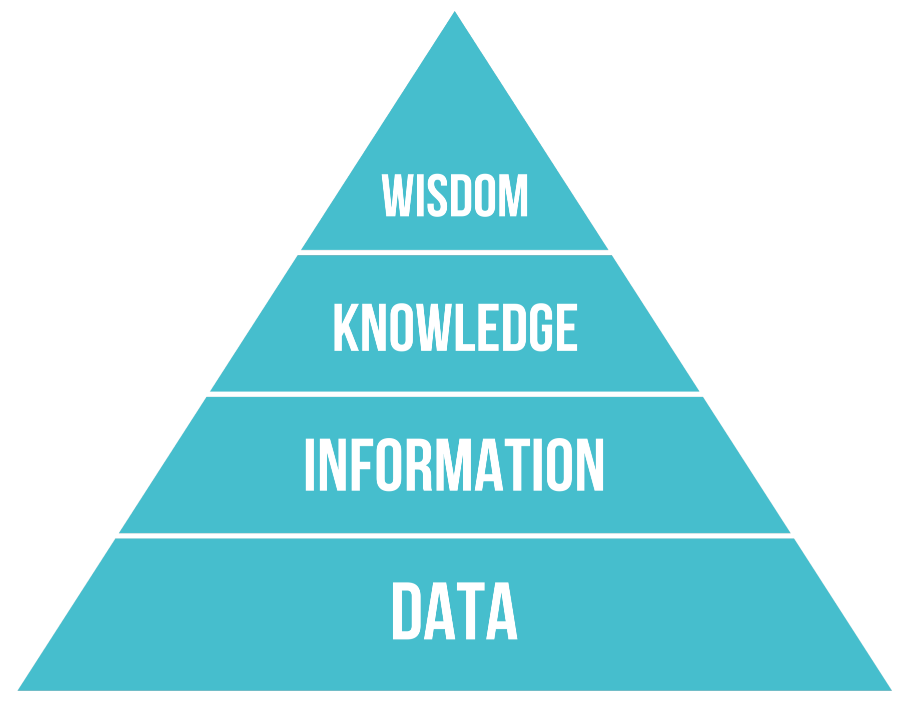
High-performance

Speed 🚀

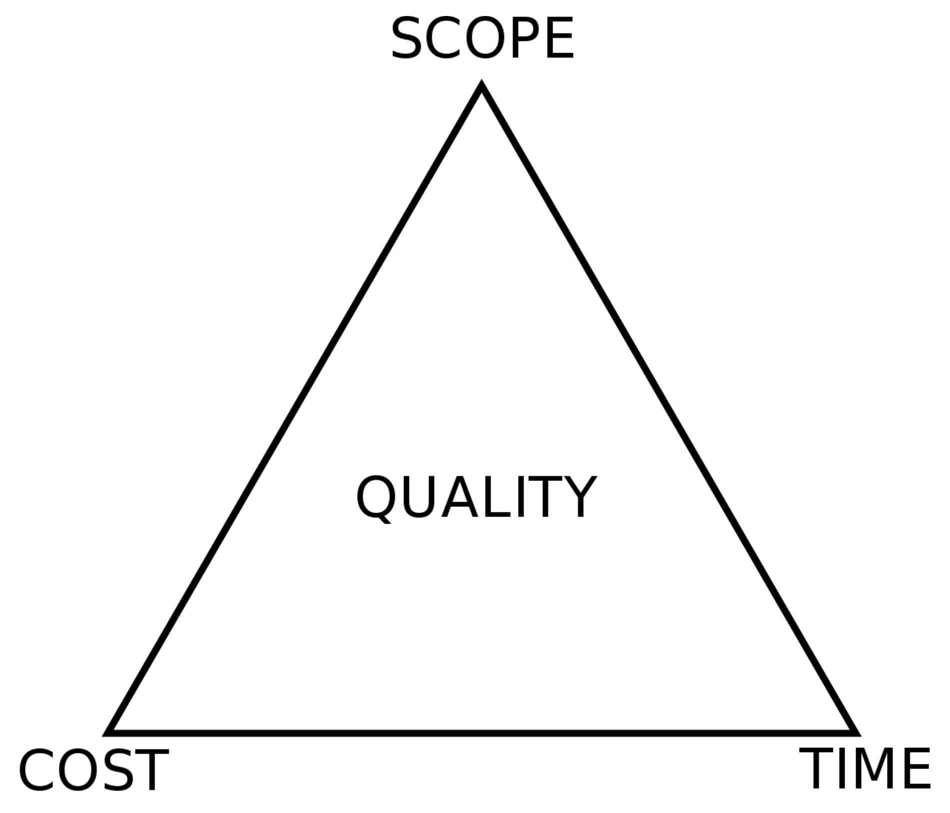
Security

Quality

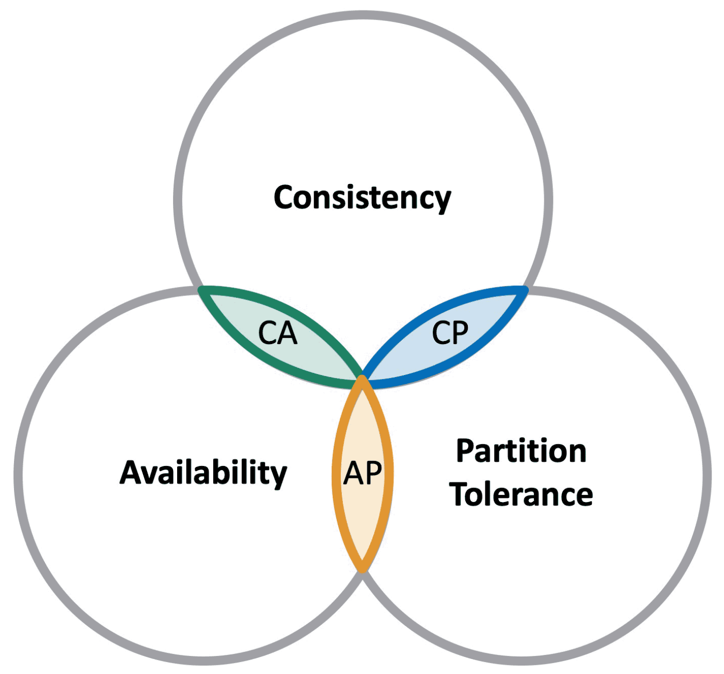
Fun 😉



Wiki: DIKW pyramid

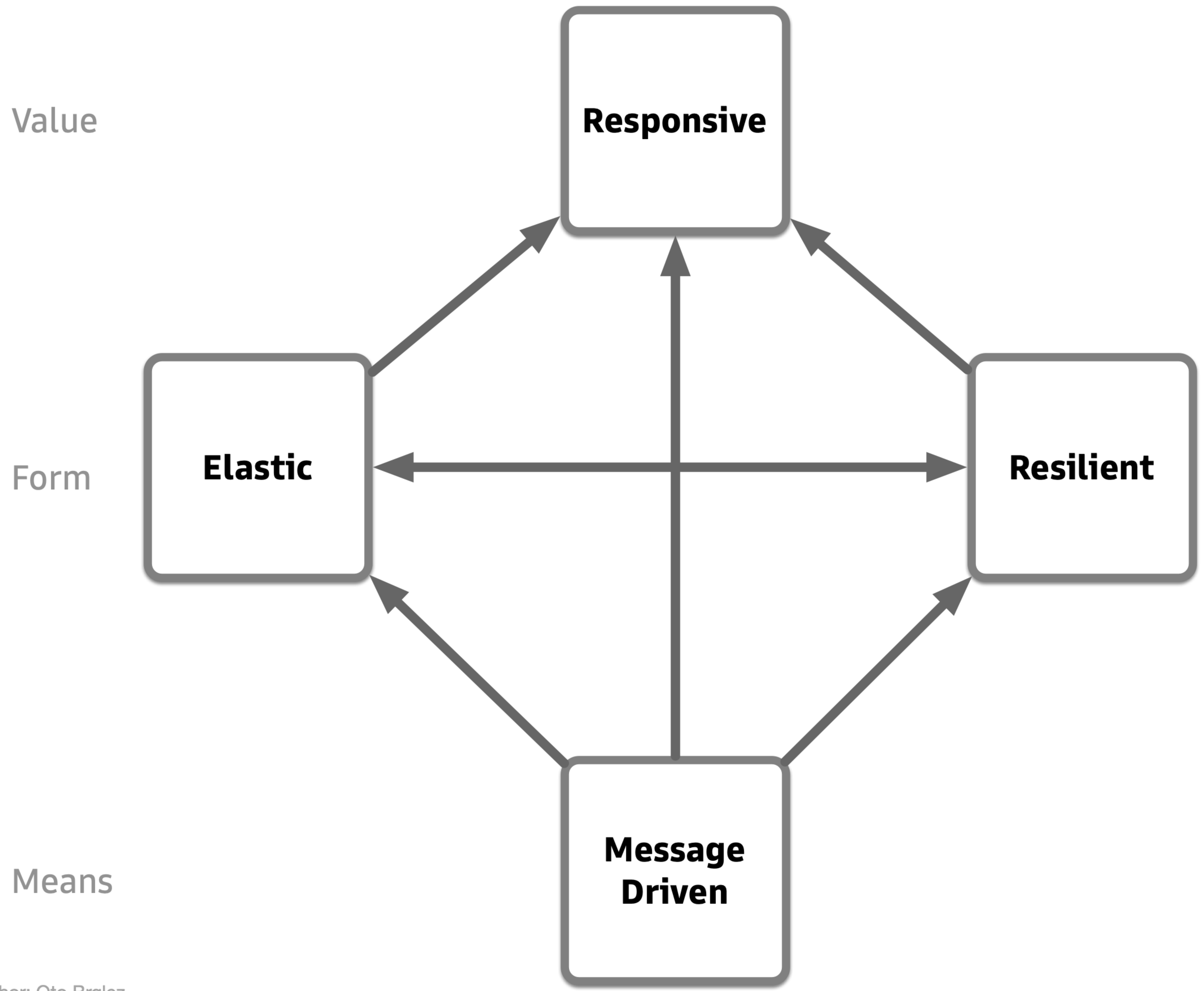


Wiki: Project Management Triangle



Wiki: CAP Theorem

The Reactive Manifesto



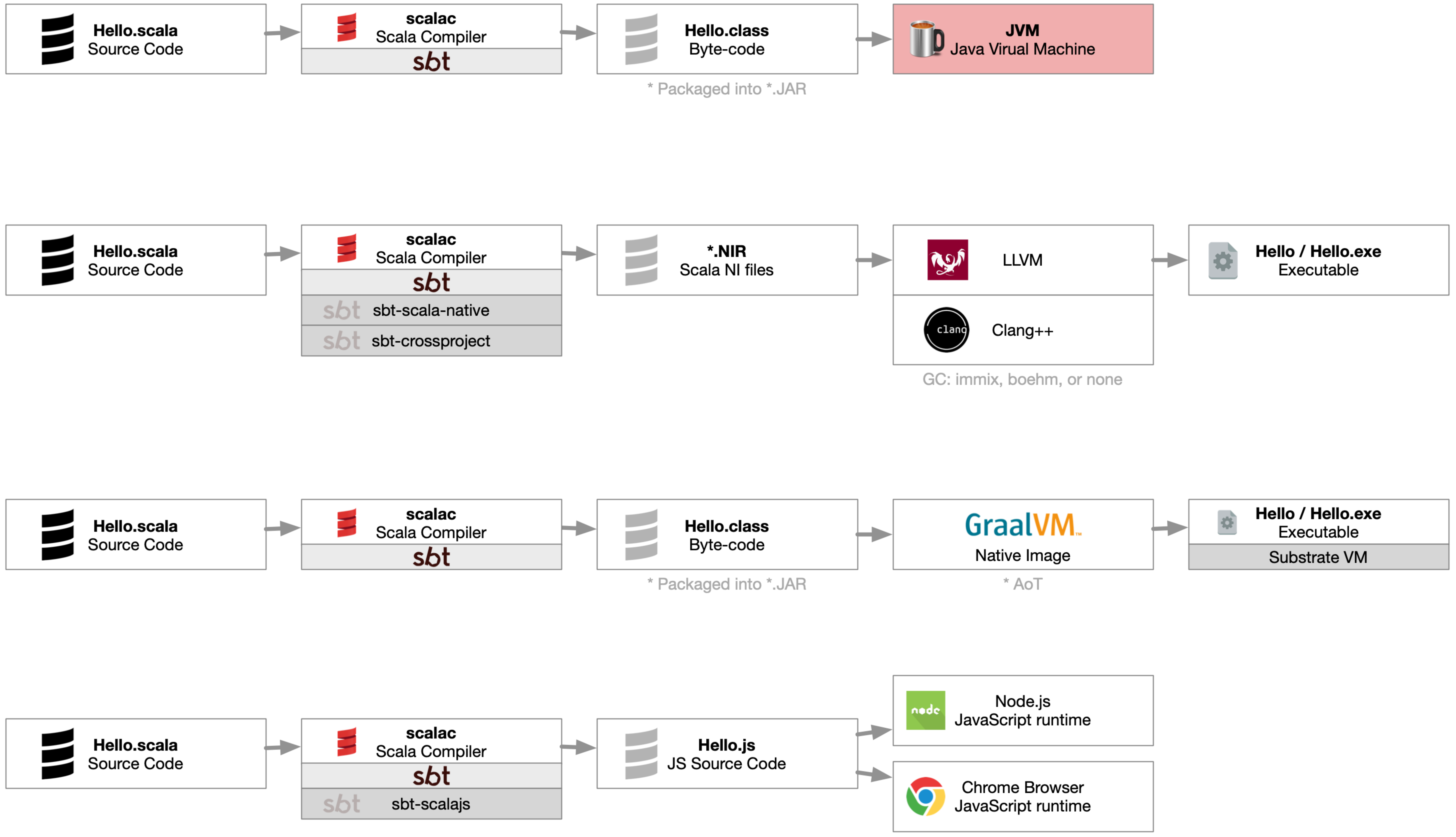
Author: Oto Brglez
Mail: otobrglez@gmail.com
Date: 13th of October 2020
Version: 0.1

More: reactivemanifesto.org

Scala

- Programming style (OOP and/or FP)
- Interoperability with Java
- JVM*
- Aesthetics;
- Pattern Matching
- Collections
  - Mutable and immutable
- Advanced Type System
  - Either, Option, Try, Tuple, etc...
- Concurrency constructs
- Case classes
- Mixins (Traits)
- *Advanced features*
  - Functions = 1st class citizens
  - Currying
  - Higher Kinded Types
  - Higher-order Functions
  - Monads,...
  - Implicit parameters
  - Implicit conversions
  - Named arguments
  - Compound types
  - For-comprehensions

Ways to compile Scala into ___ ?



When code talks, bullshit walks,...

Scala

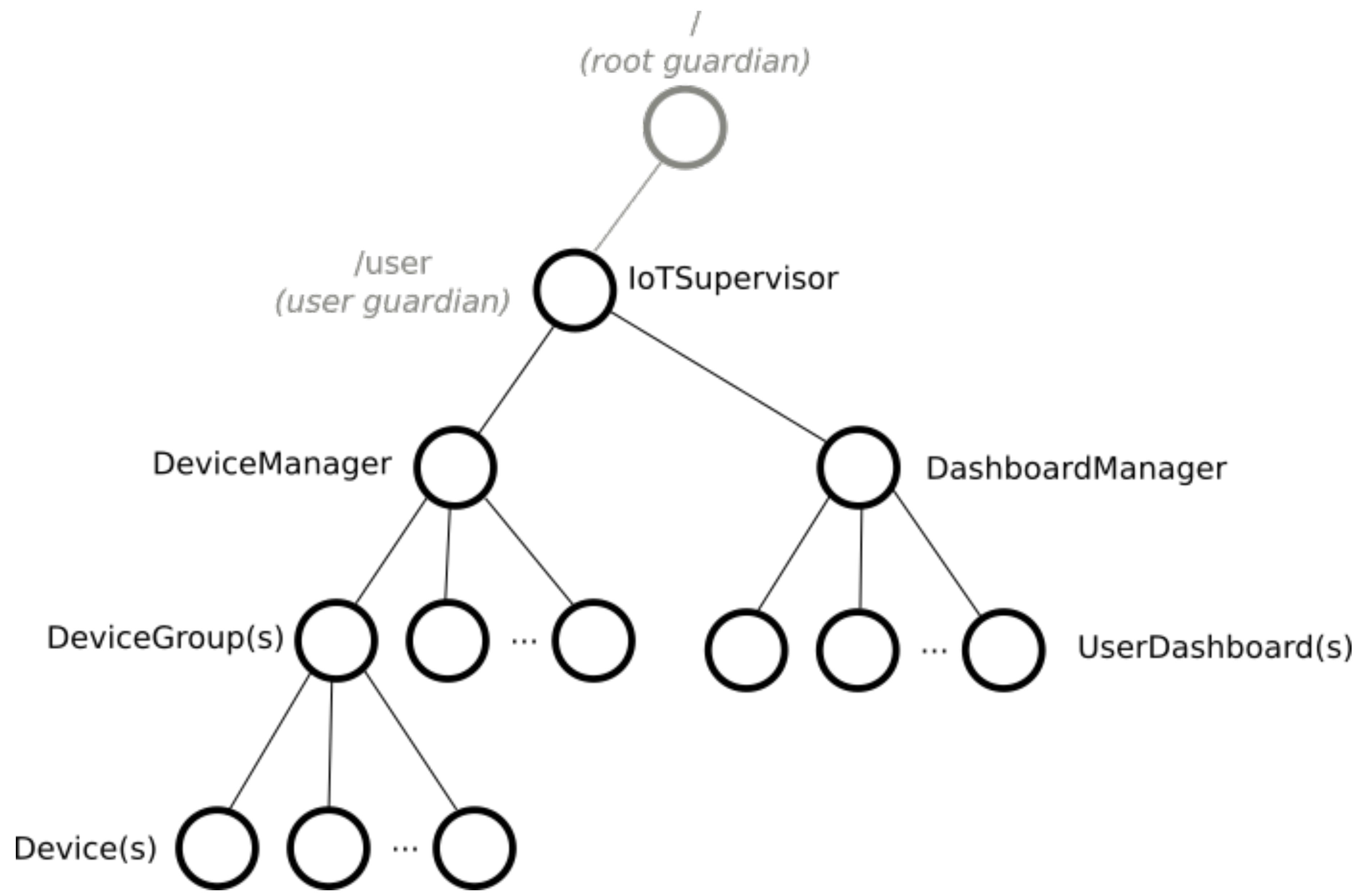
Akka

- **Simpler Concurrent & Distributed Systems**
  Actors and **Streams** let you build systems that scale up, using the resources of a server more efficiently, and out, using multiple servers.
- **Resilient by Design**
  Building on the principles of **The Reactive Manifesto** Akka allows you to write systems that self-heal and stay responsive in the face of failures.
- **High Performance**
  Up to 50 million msg/sec on a single machine. Small memory footprint; ~2.5 million actors per GB of heap.
- **Elastic & Decentralized**
  Distributed systems without single points of failure. Load balancing and adaptive routing across nodes. **Event Sourcing** and **CQRS** with Cluster Sharding. Distributed Data for eventual consistency using **CRDT**s.
- **Reactive Streaming Data**
  Asynchronous non-blocking stream processing with backpressure. Fully async and streaming HTTP server and client provides a great platform for building microservices. Streaming integrations with **Alpakka**.

Akka

- Akka Actors
- Akka Streams
- Akka HTTP
- Akka Cluster
- Akka Cluster Sharding
- Akka Distributed Data (CRDTs)
- Akka Persistance (ES/CQRS)
- Alpakka for Integrations
- Akka gRPC
- Lagom*
- Cloudflow*
- Cloudstate*

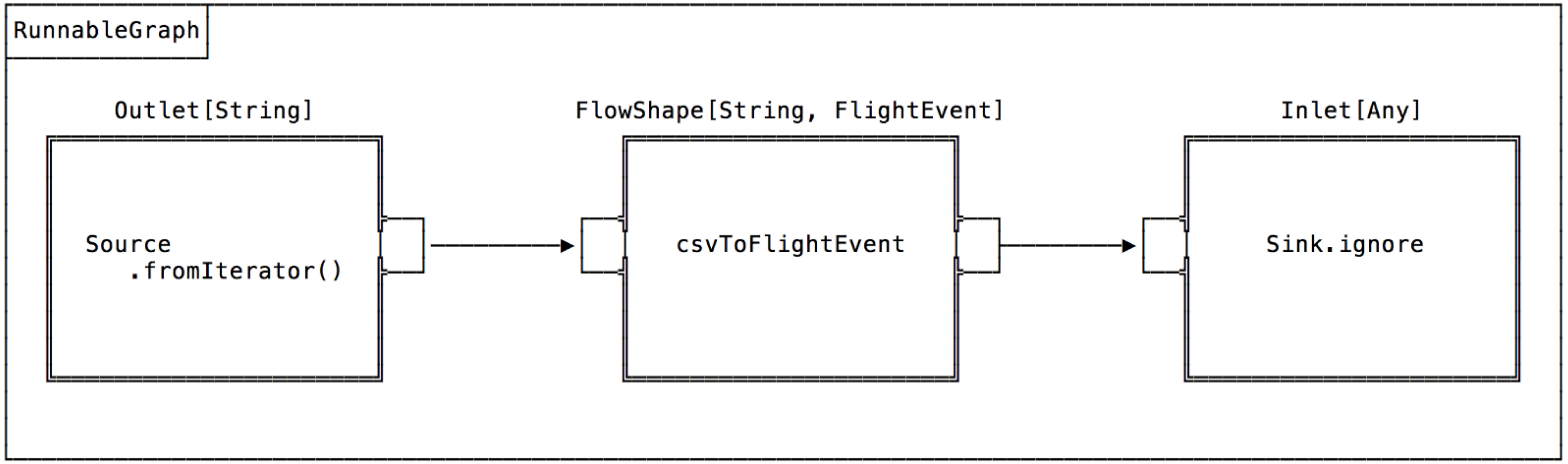
Akka Actors


/
(root guardian)
/user
(user guardian)
IoTSupervisor
DeviceManager
DashboardManager
DeviceGroup(s)
...
...
UserDashboard(s)
Device(s)
...

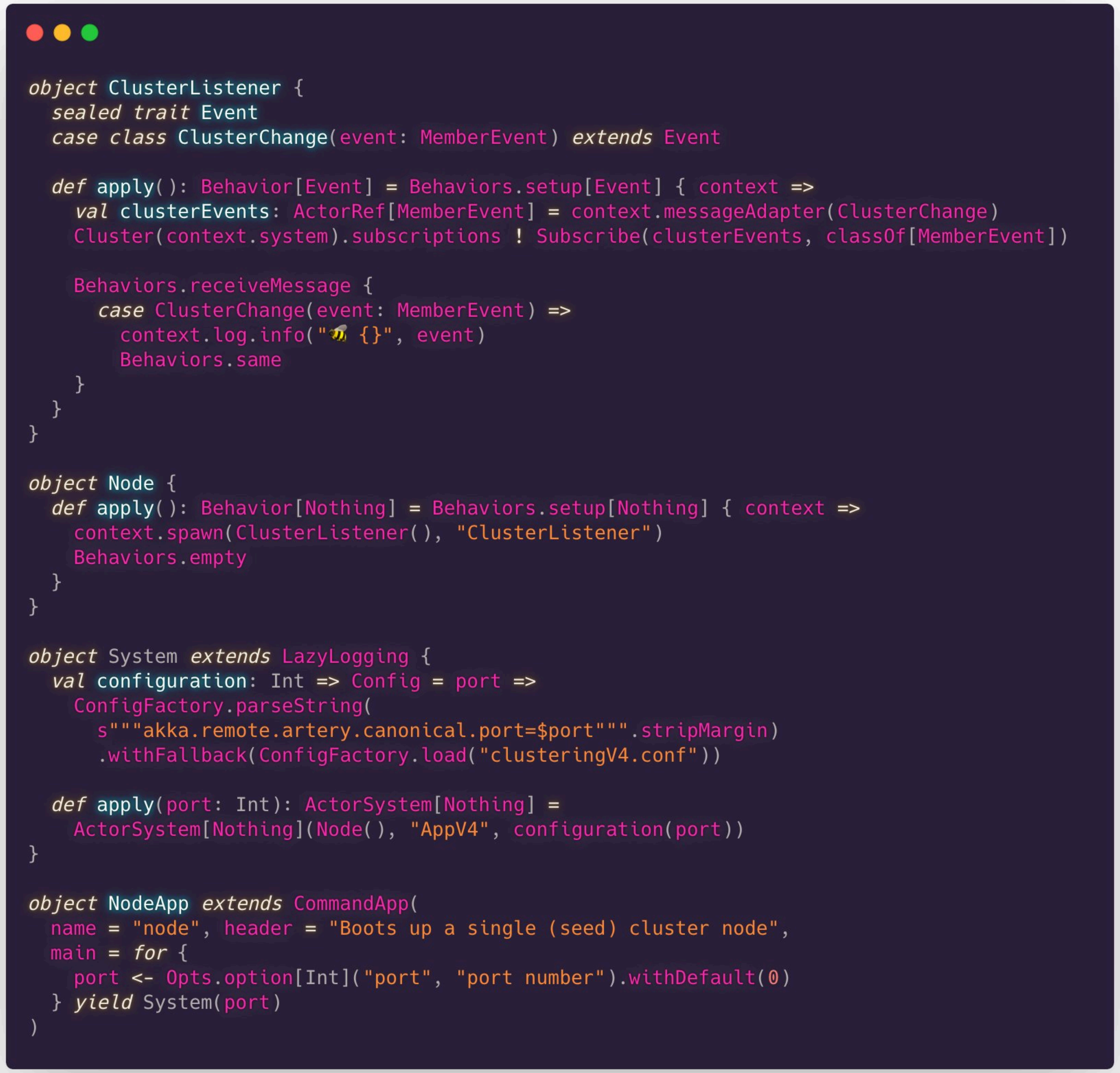
A splash of,...

Akka

Akka Streams



Akka Clustering



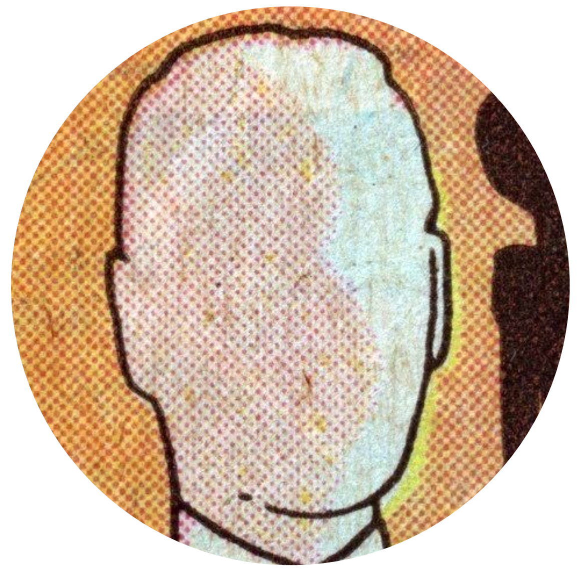
```scala
object ClusterListener {
  sealed trait Event
  case class ClusterChange(event: MemberEvent) extends Event

  def apply(): Behavior[Event] = Behaviors.setup[Event] { context =>
    val clusterEvents: ActorRef[MemberEvent] = context.messageAdapter(ClusterChange)
    Cluster(context.system).subscriptions ! Subscribe(clusterEvents, classOf[MemberEvent])

    Behaviors.receiveMessage {
      case ClusterChange(event: MemberEvent) =>
        context.log.info("🐝 {}", event)
        Behaviors.same
    }
  }
}

object Node {
  def apply(): Behavior[Nothing] = Behaviors.setup[Nothing] { context =>
    context.spawn(ClusterListener(), "ClusterListener")
    Behaviors.empty
  }
}

object System extends LazyLogging {
  val configuration: Int => Config = port =>
    ConfigFactory.parseString(
      s"""akka.remote.artery.canonical.port=$port""".stripMargin)
      .withFallback(ConfigFactory.load("clusteringV4.conf"))

  def apply(port: Int): ActorSystem[Nothing] =
    ActorSystem[Nothing](Node(), "AppV4", configuration(port))
}

object NodeApp extends CommandApp(
  name = "node", header = "Boots up a single (seed) cluster node",
  main = for {
    port <- Opts.option[Int]("port", "port number").withDefault(0)
  } yield System(port)
)
```

alpakka

- AMQP
- Apache Camel
- Apache Cassandra
- Apache Geode
- Apache Kafka
- Apache Kudu
- Apache Solr
- Avro Parquet
- AWS EventBridge
- AWS DynamoDB
- AWS Kinesis and Firehose
- AWS Lambda
- AWS S3
- AWS SNS
- AWS SQS
- Azure Event Hubs
- Azure IoT Hub
- Azure Storage Queue
- Couchbase
- Elasticsearch
- Eventuate
- File
- FS2
- FTP
- Google Cloud BigQuery
- Google Cloud Pub/Sub
- Google Cloud Pub/Sub gRPC
- Google Cloud Storage
- Google FCM
- gRPC
- Hadoop Distributed File System - HDFS
- HBase
- HTTP
- IBM Bluemix Cloud Object Storage
- IBM Db2 Event Store
- InfluxDB
- IronMQ
- JMS
- MapR Database
- MongoDB
- MQTT
- MQTT Streaming
- OrientDB
- Pulsar
- Pravega
- Server-sent Events (SSE)
- Slick (JDBC)
- Spring Web
- TCP
- UDP
- Unix Domain Socket

Final notes

Q&A



Fin.

@otobrglez

otobrglez@gmail.com

epic.blog