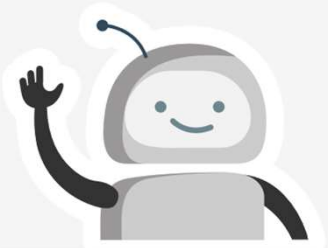
exp.oCon 2025

THE ACTOR MODEL

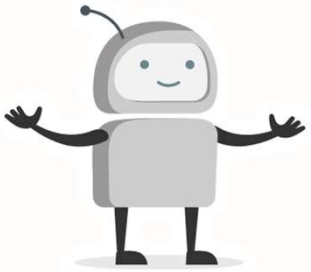


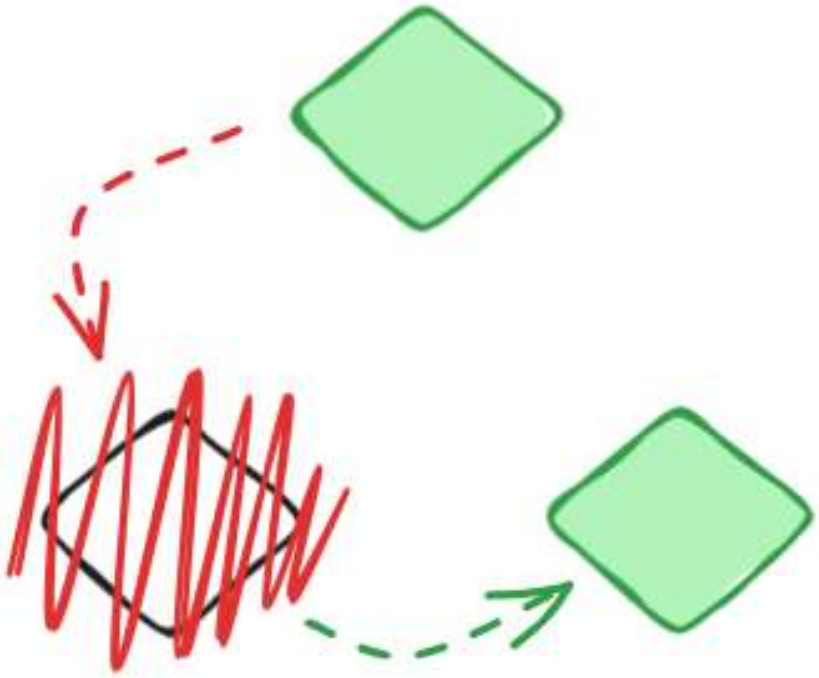
Fault Tolerant – Highly Concurrent – Distributed Applications

AGENDA

The Plan

1. Motivation
2. Branding
3. Definitions
4. Fault Tolerance
5. High Concurrency
6. Distributed Applications
7. Summary

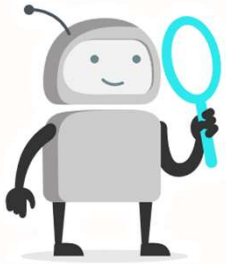


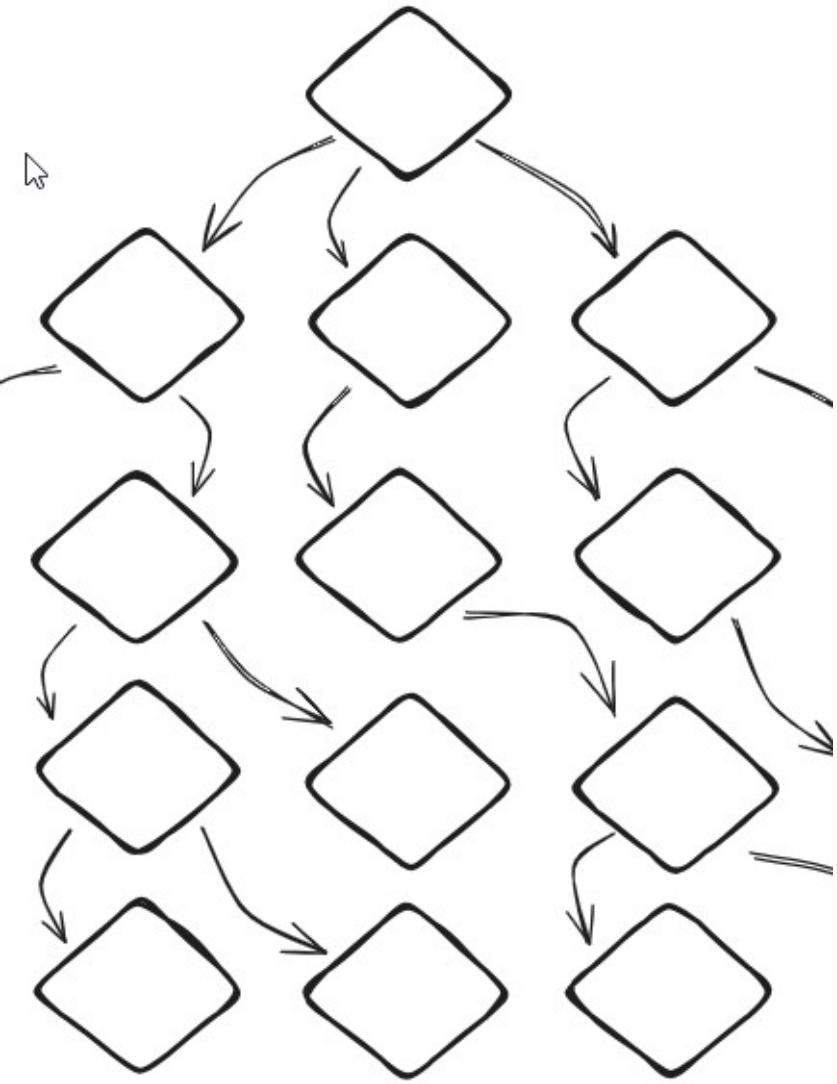


MOTIVATION

Fault Tolerance

Build fault tolerance into applications at a structural level

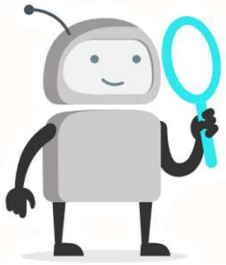


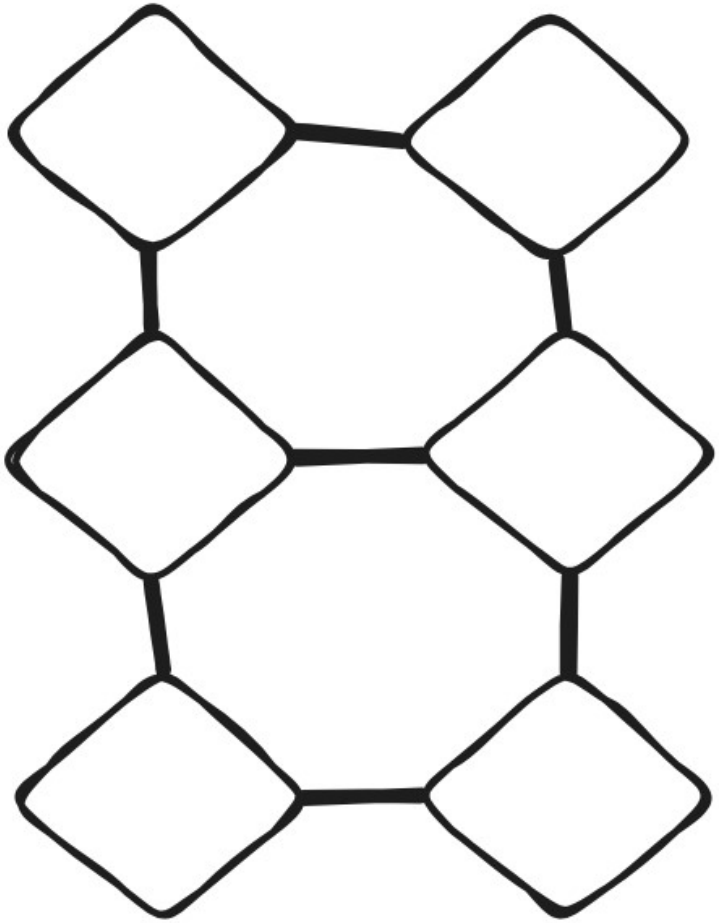


MOTIVATION

Massive Multithreading

Increase throughput with performant high concurrency

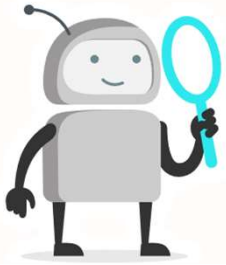




MOTIVATION

Distributed Applications

Design applications that scale across nodes



BRANDING

Implementations of the Actor Model

And Who's Using Akka?

akka Pekko

JVM

C++

elixir

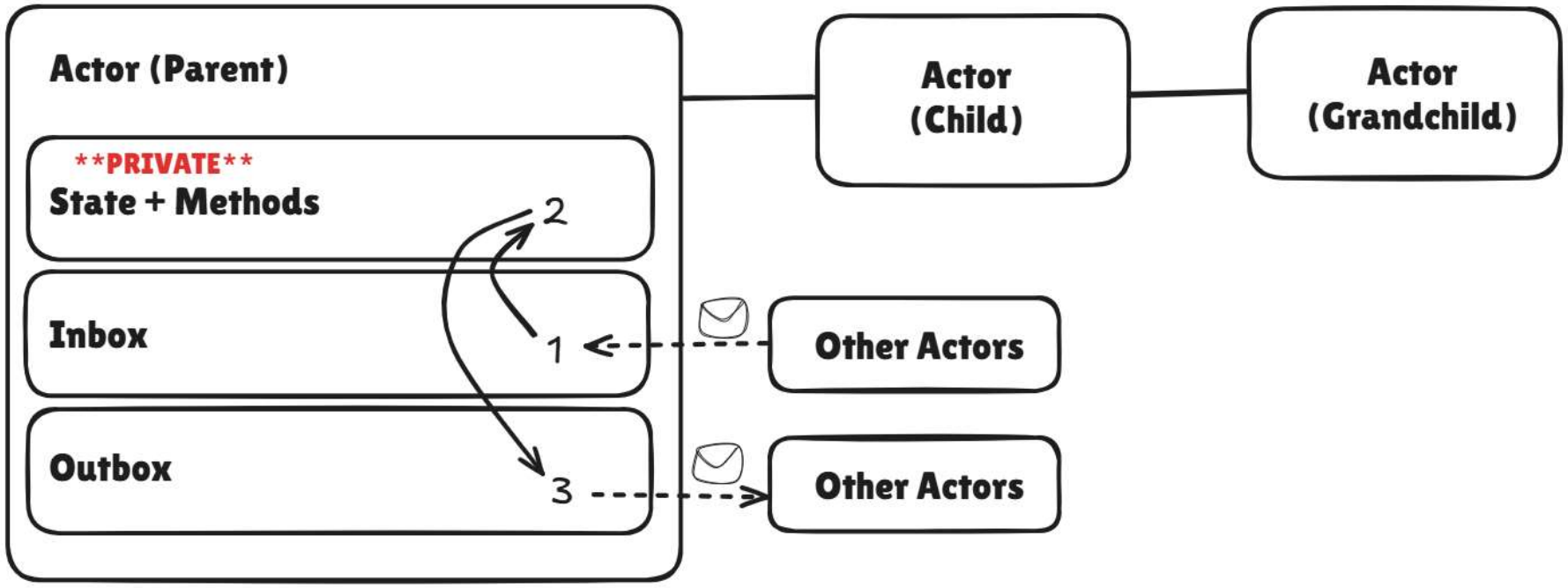
Erlang VM

ACTIX

Rust

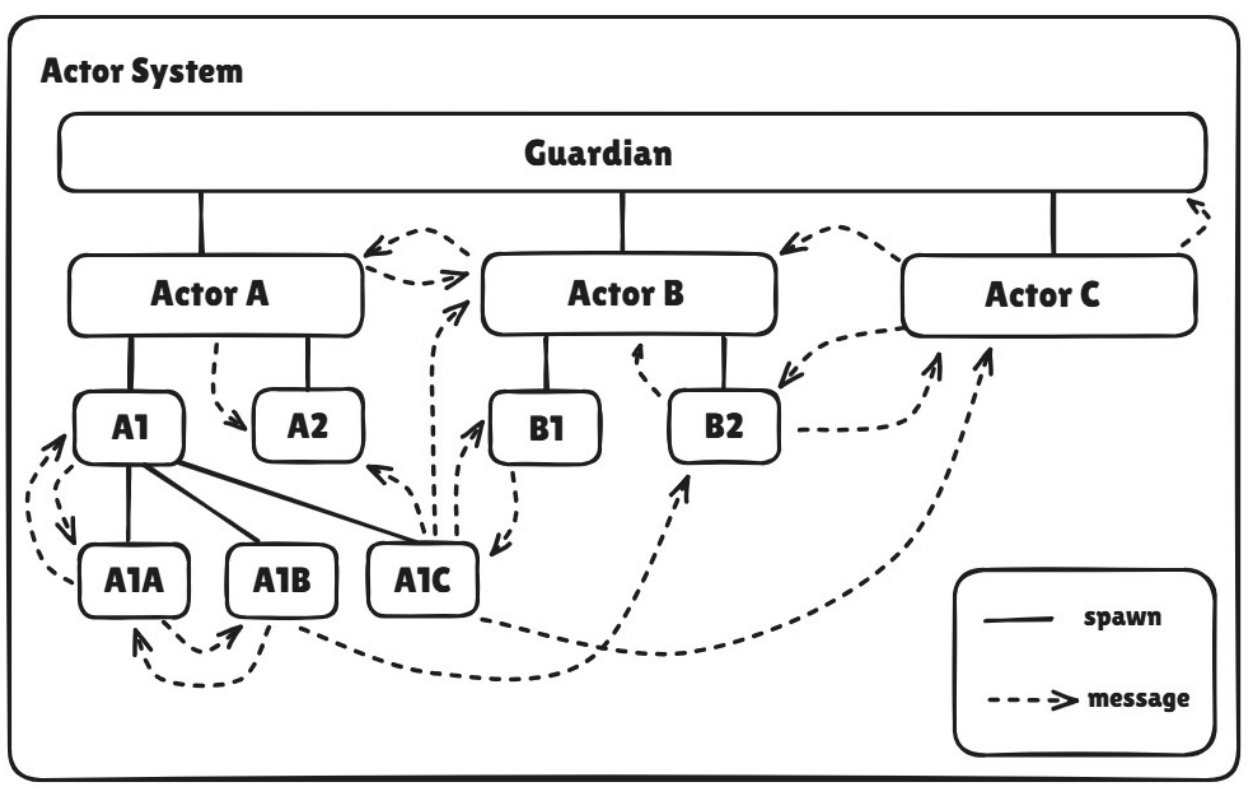
ACTORS

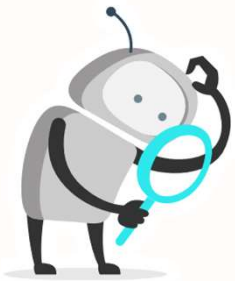
Anatomy of an Actor



THE ACTOR SYSTEM

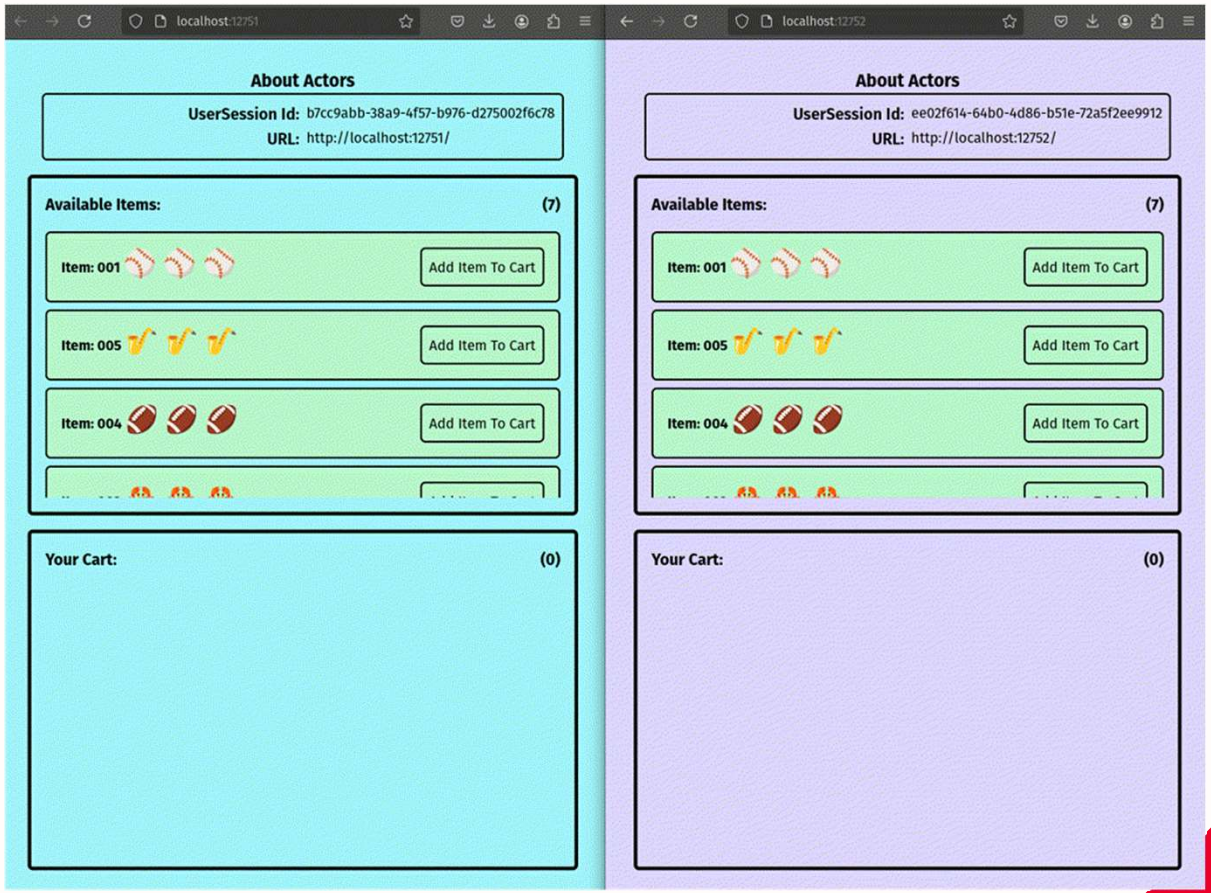
Who's Managing the Actors?

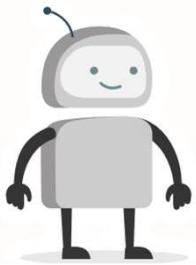




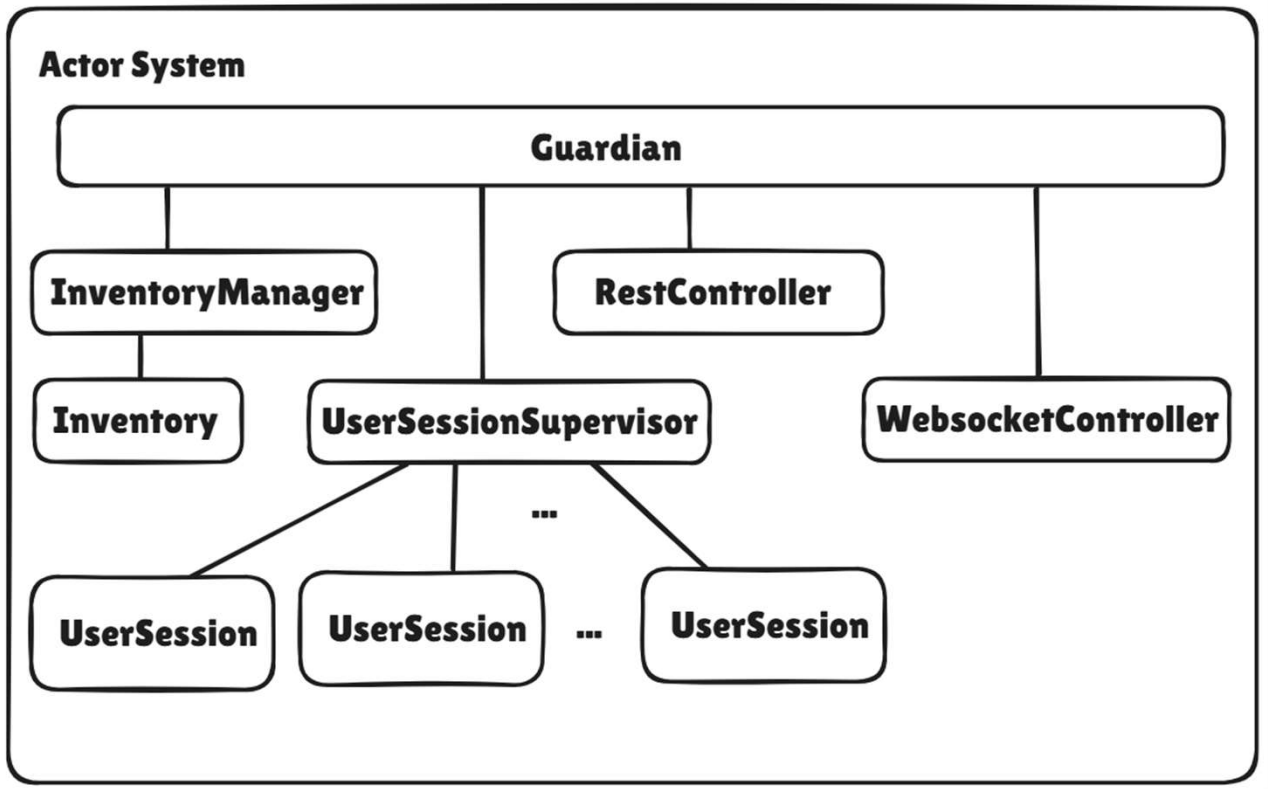
EXAMPLE – INVENTORY APP

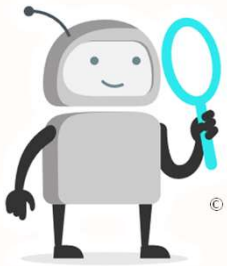
- **Repo:** gh/nathanielbellamy/aboutactors
- **Language:** Scala 3.3.4
- **Build Tool:** Maven
- **Akka Version:** 2.10.2
- **IDE:** Intellij Idea





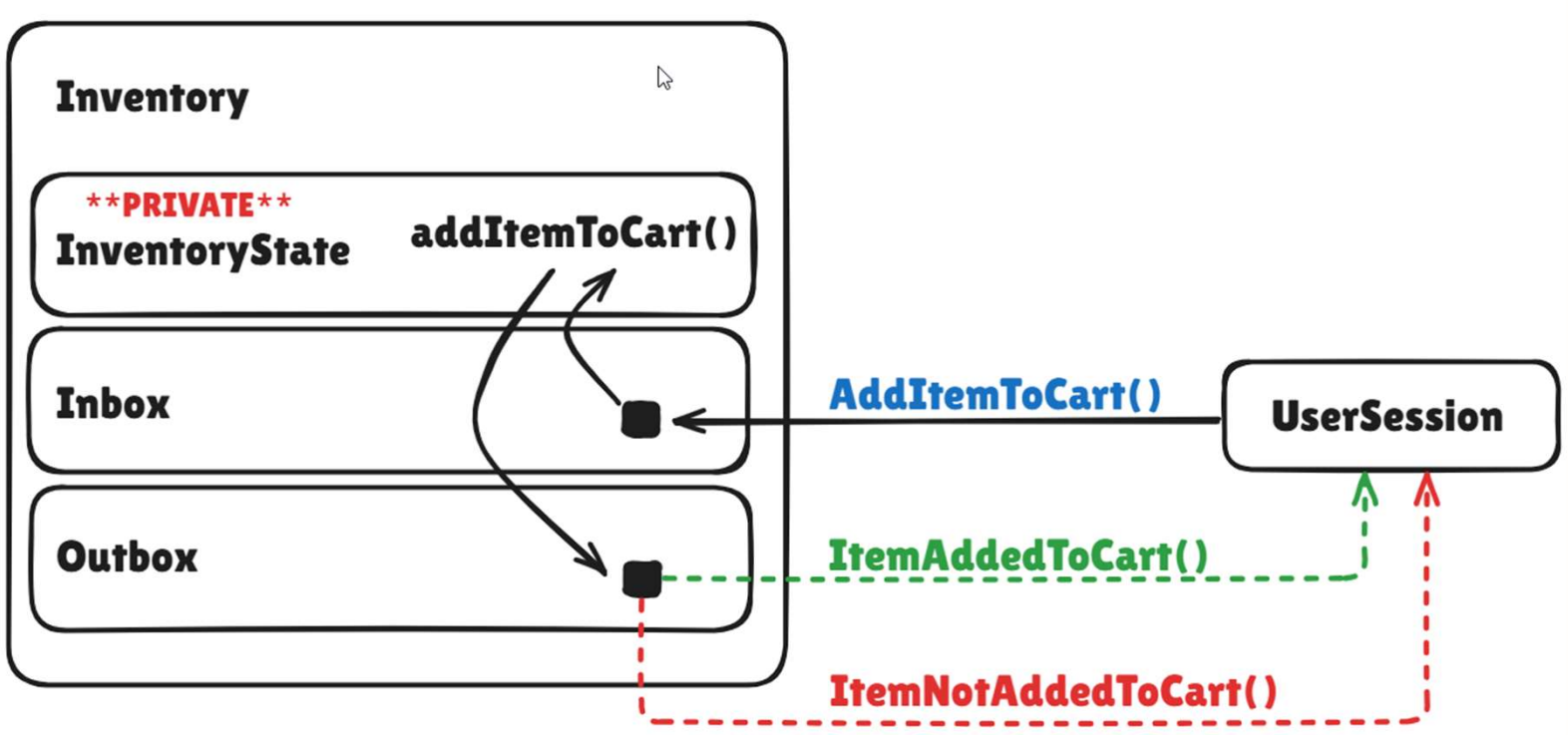
EXAMPLE – INVENTORY APP

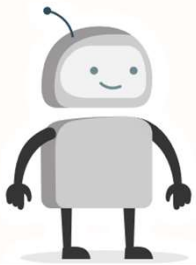




ACTOR EXAMPLE

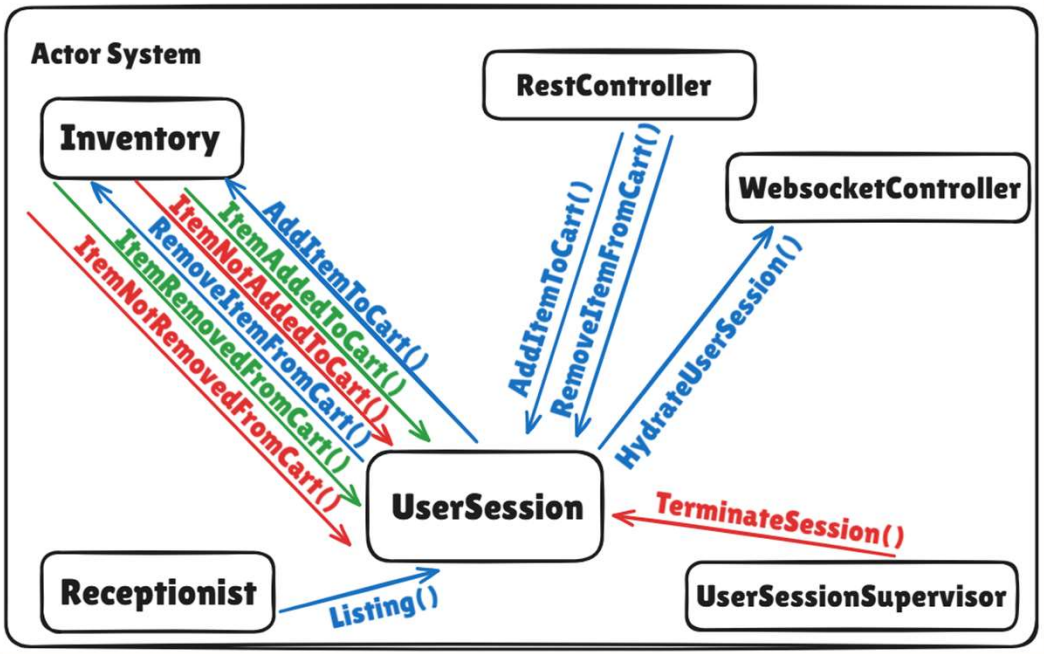
Adding An Item To User's Cart

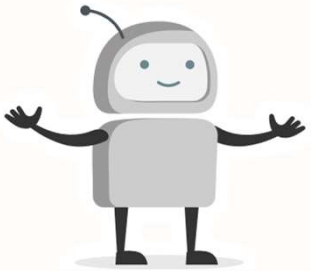




MESSAGES

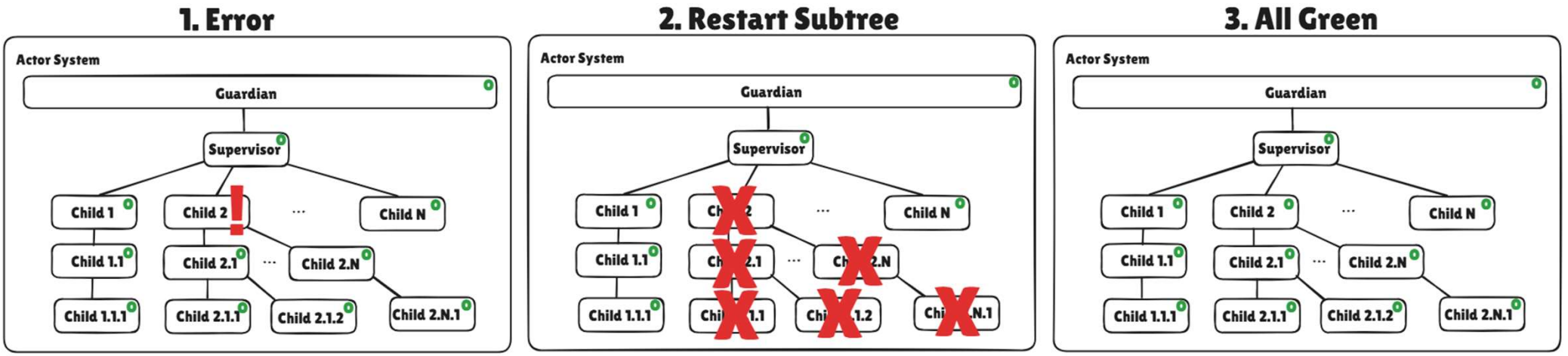
What about the messages themselves?

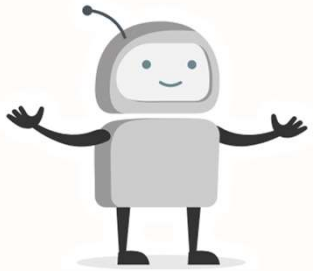




FAULT TOLERANCE

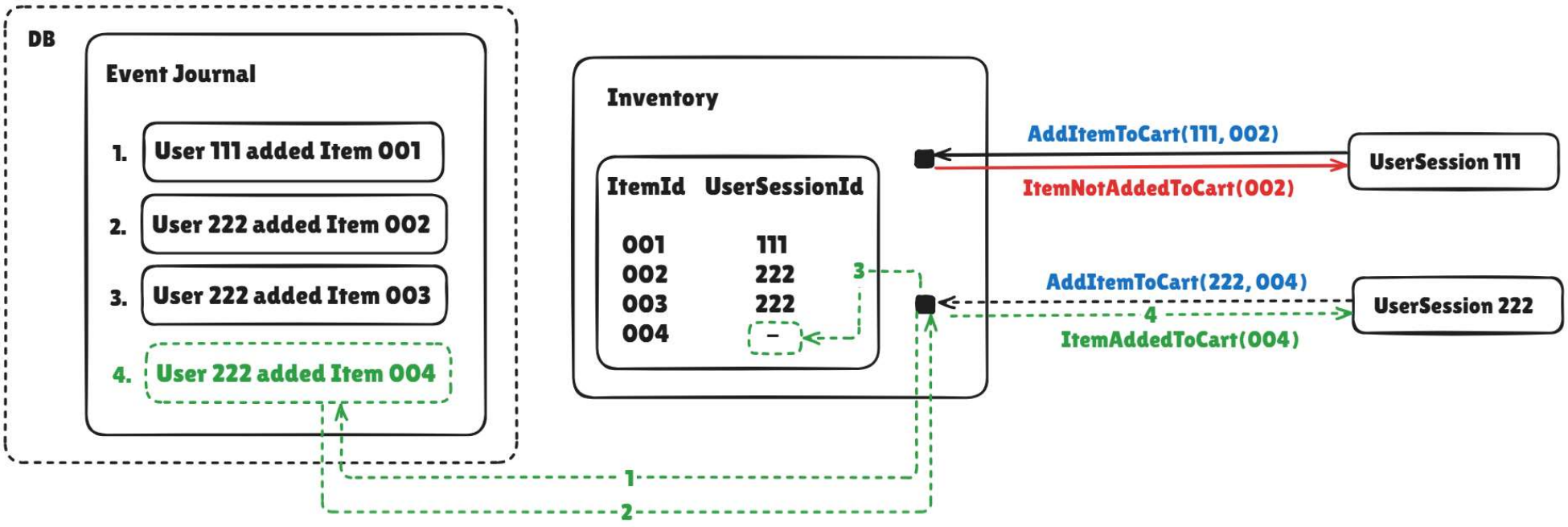
What happens when code fails?

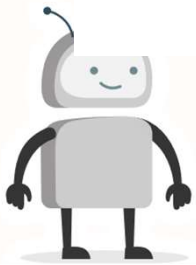




FAULT TOLERANCE

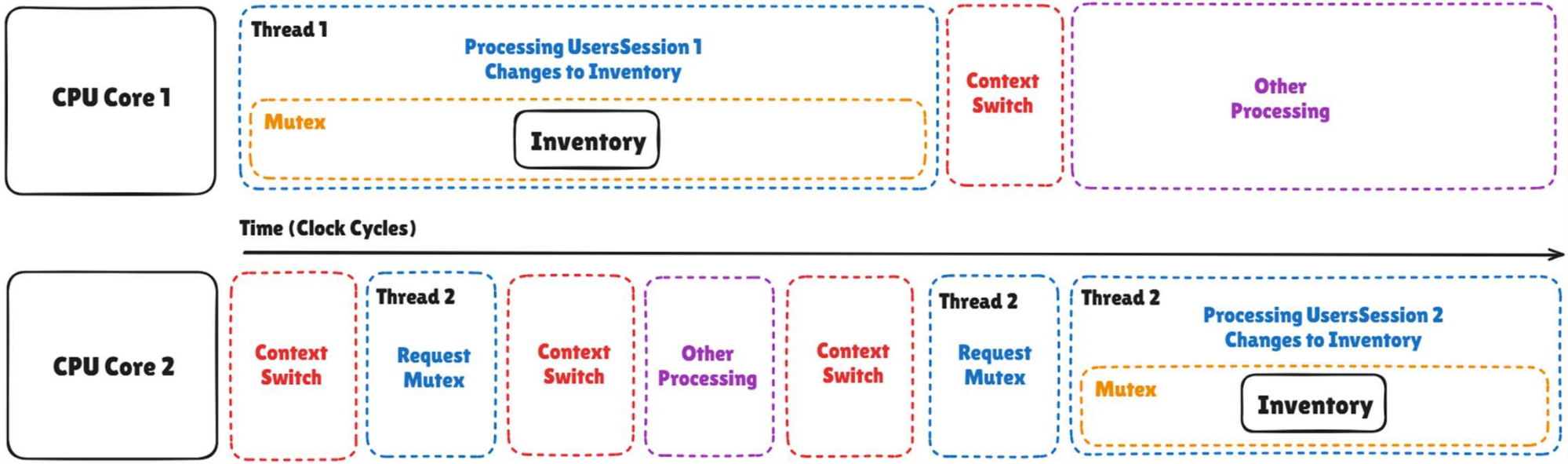
Persistent State Built From Messages

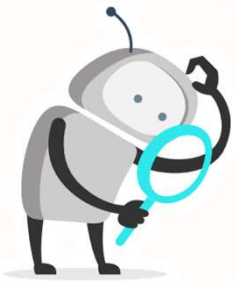




MASSIVE MULTITHREADING

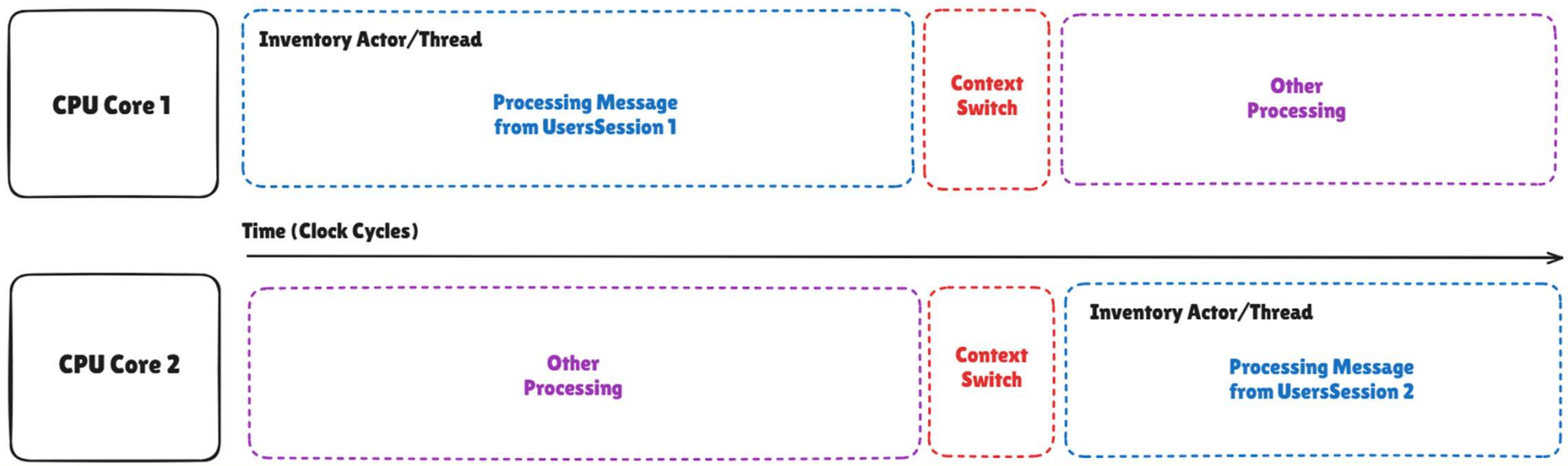
The Trouble With Locking **(Non Actor Model)**

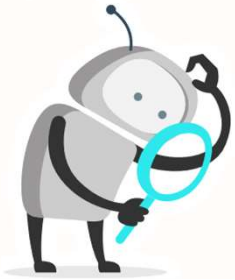




MASSIVE MULTITHREADING

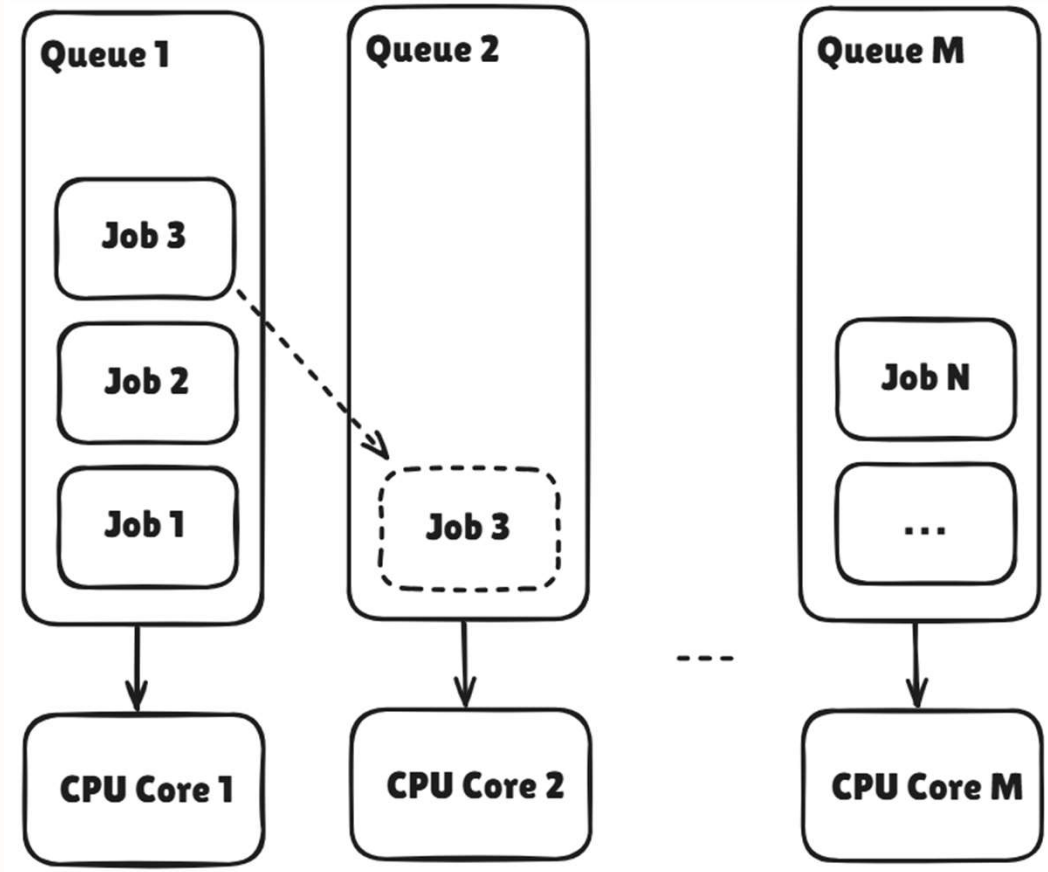
No Unnecessary Context Switching

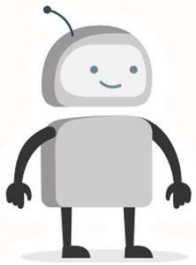




MASSIVE MULTITHREADING

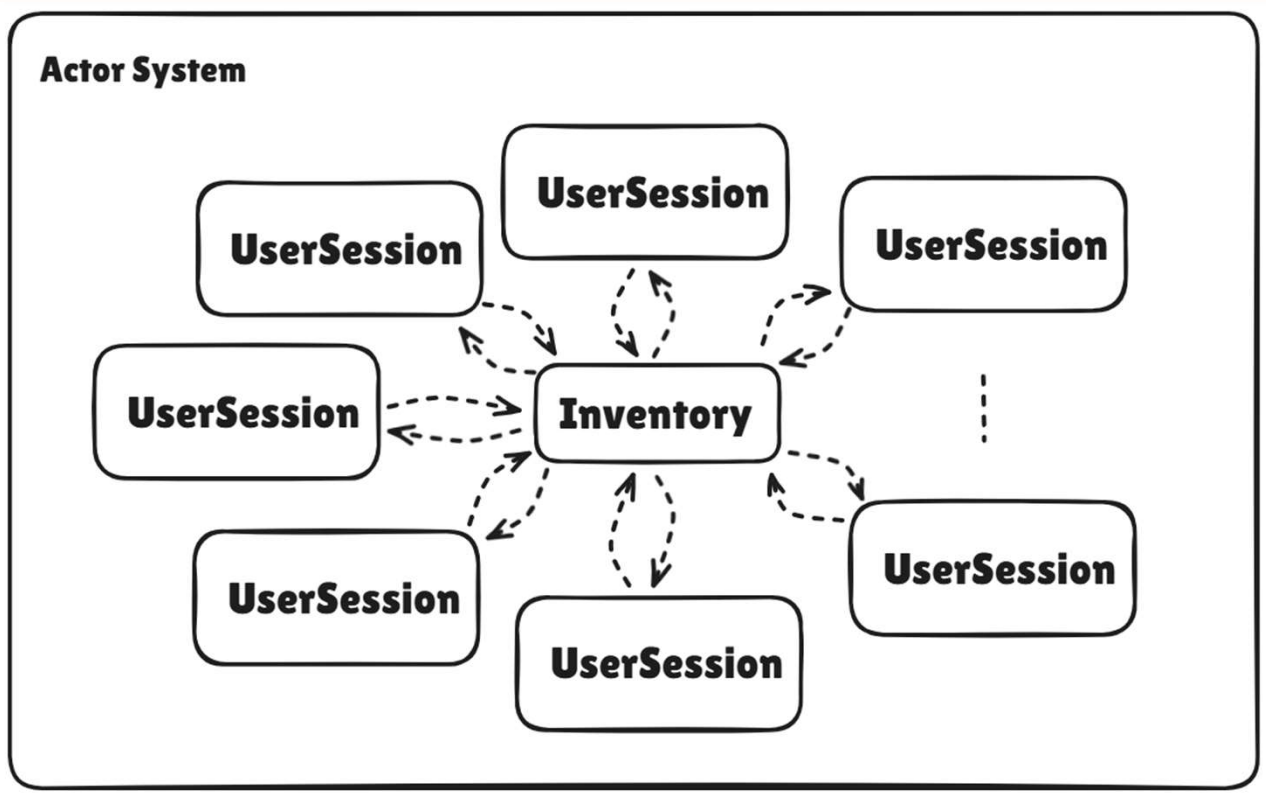
Lock Free Is The Way To Be

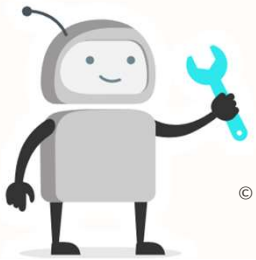




MASSIVE MULTITHREADING

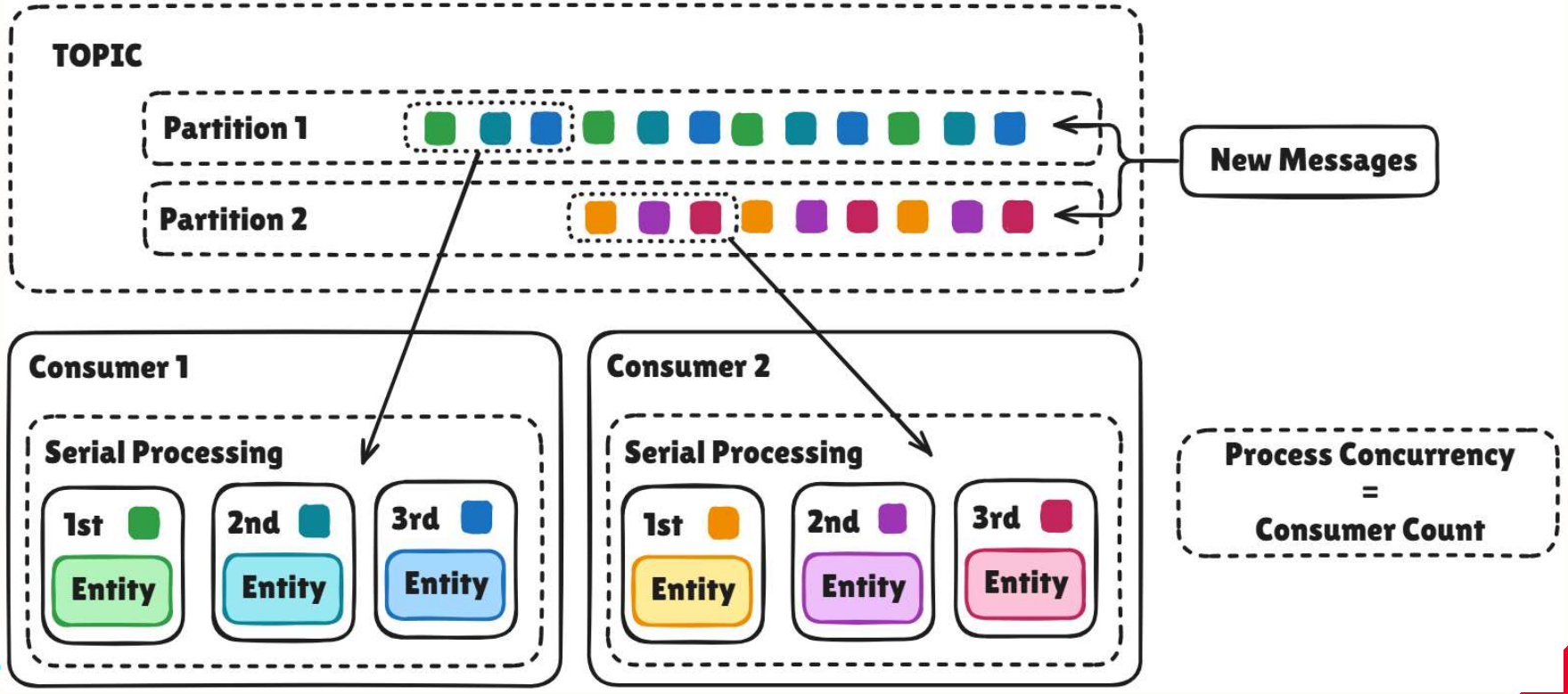
Wait, How Many Actors?





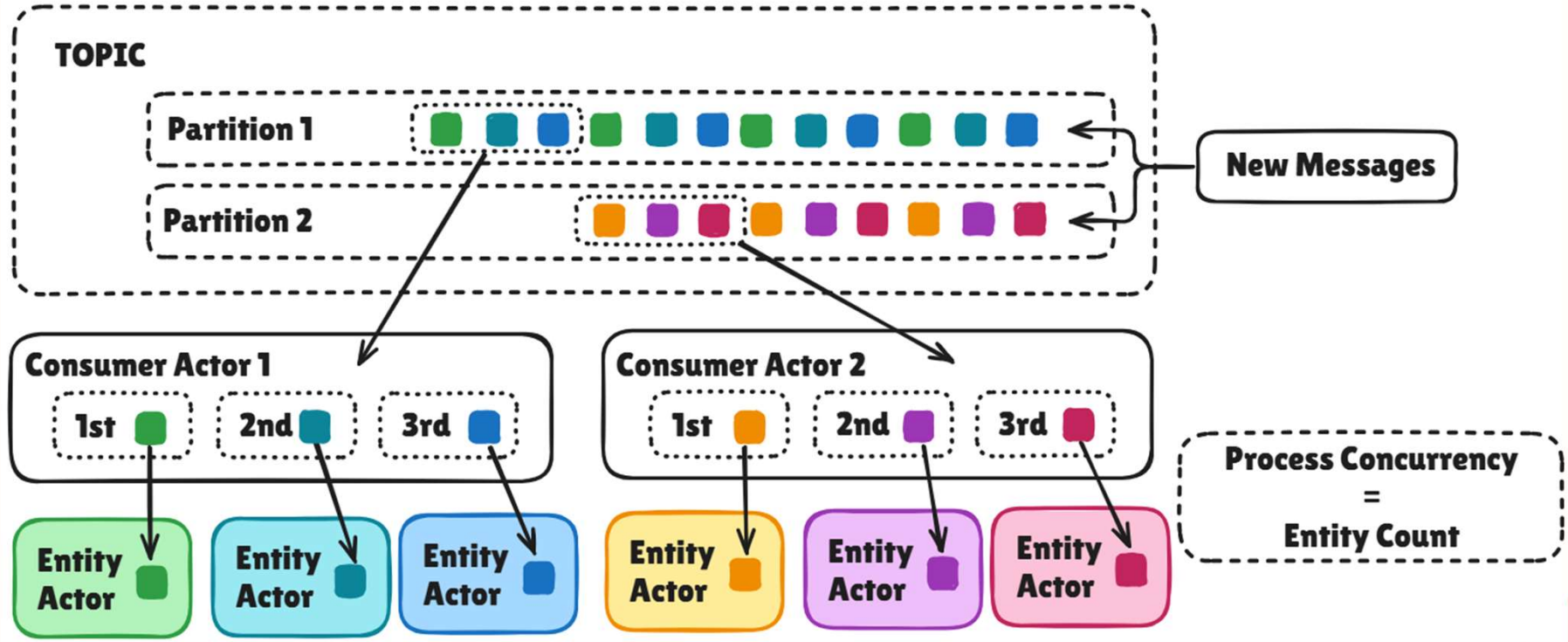
MASSIVE MULTITHREADING

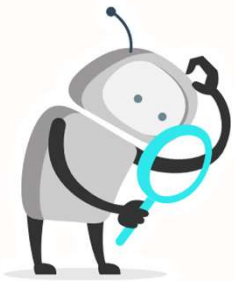
Limitations of Kafka + Spring



MASSIVE MULTITHREADING

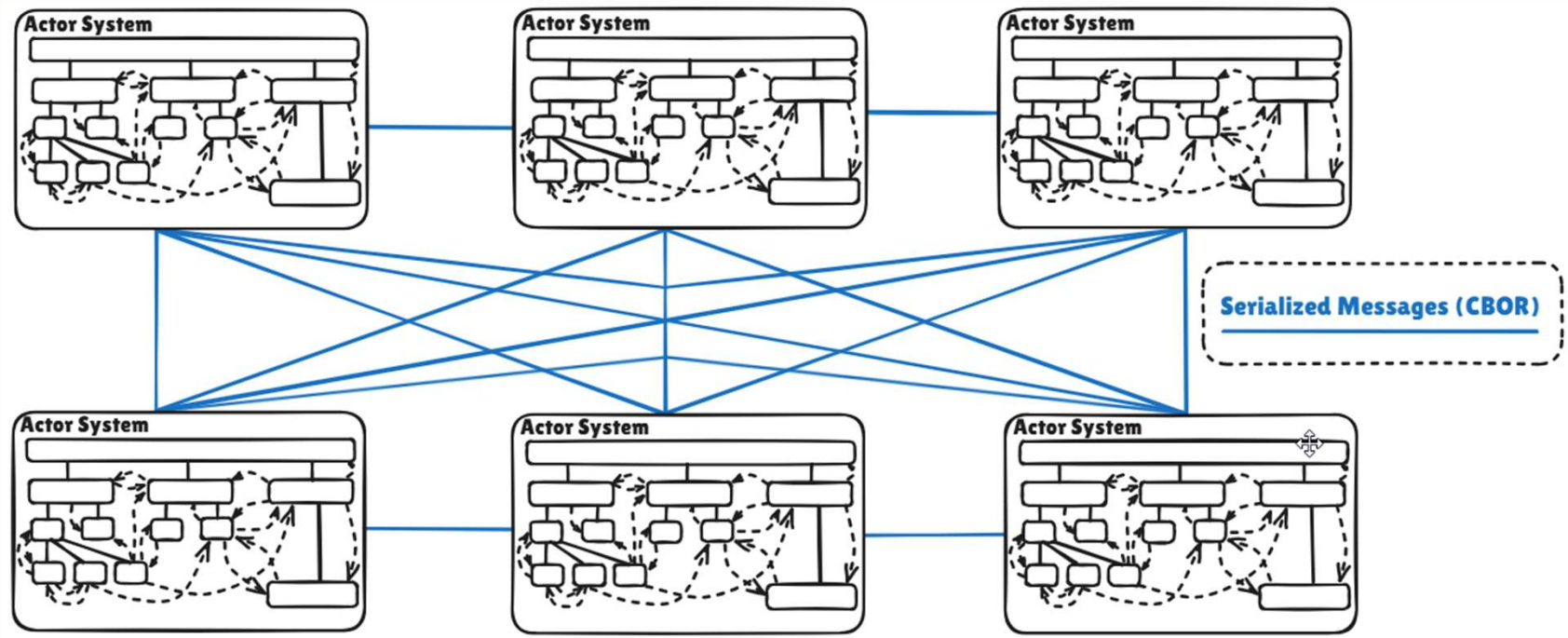
High Concurrency Processing With Kafka + Akka (Alpakka)

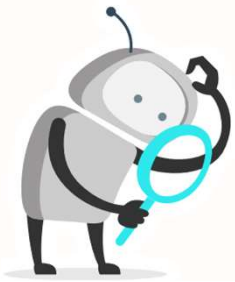




DISTRIBUTED APPLICATIONS

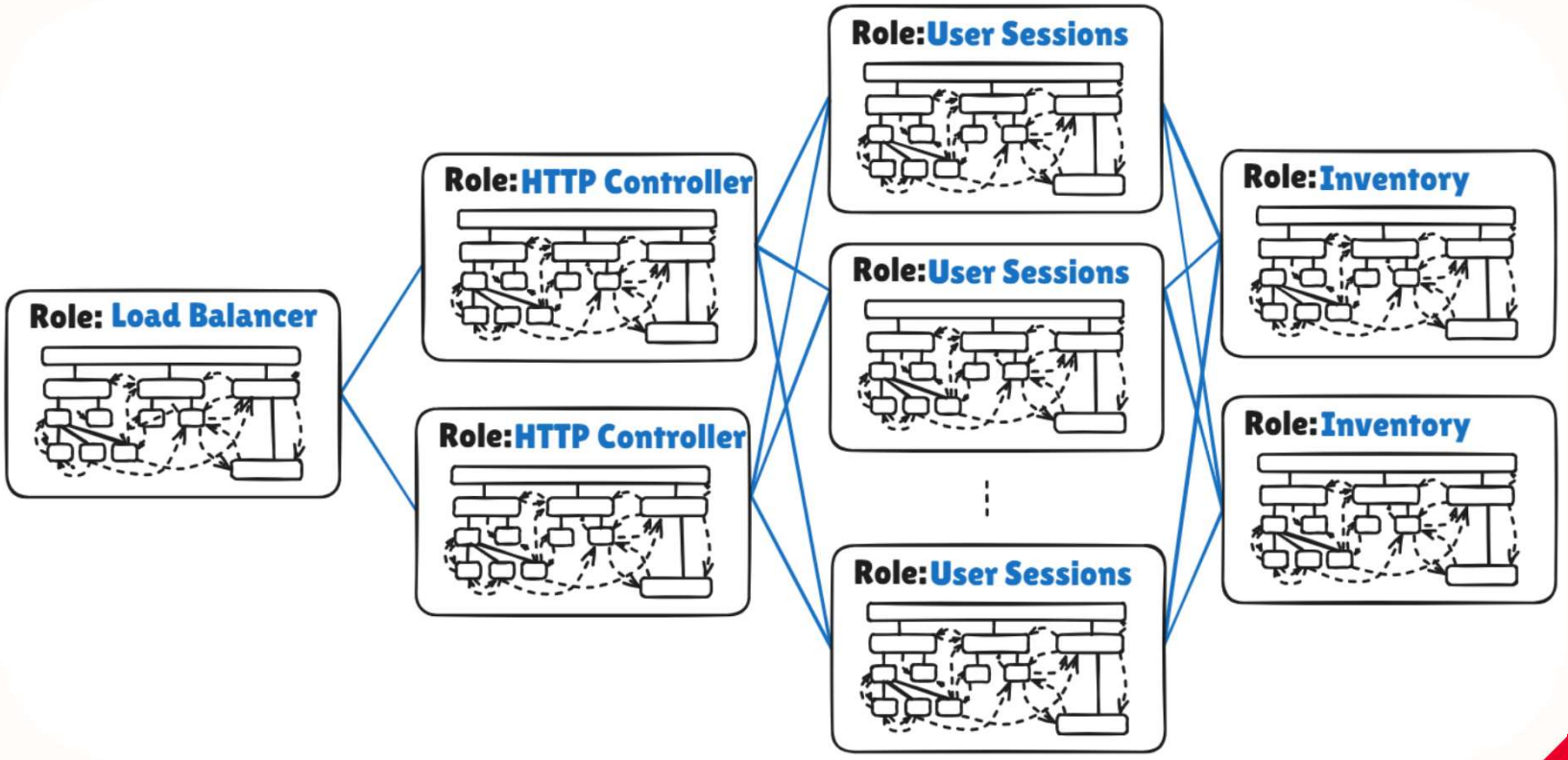
Multiple Actor Systems Working as One with Cluster Sharding

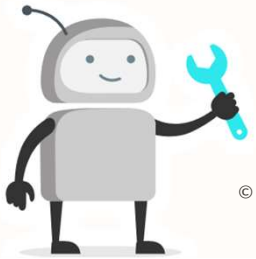




DISTRIBUTED APPLICATIONS

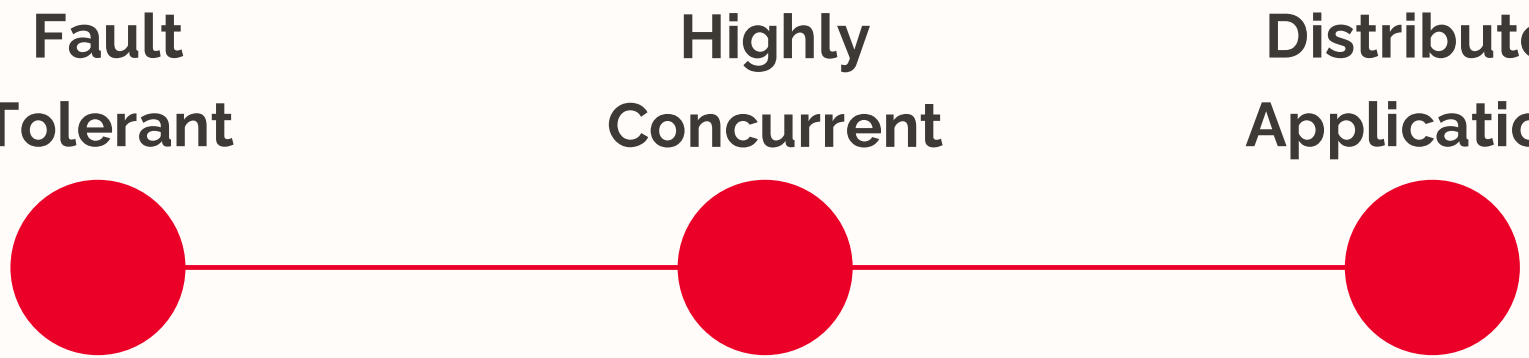
Roles and Fault Tolerance





SUMMARY

The Actor Model

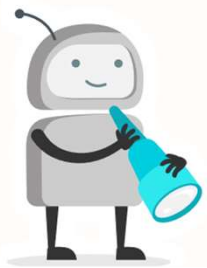


- Supervisor Pattern
- Compartmentalized Errors
- Persistence (Event Sourcing)

- Threading W/O Locking
- Efficient CPU Scheduling
- Processing Thread Per Entity

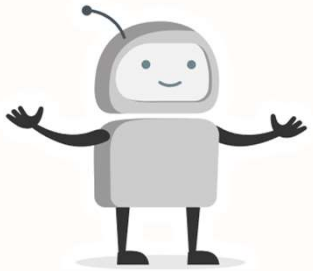
- Cluster Sharding
- Actor Migration
- Node Roles

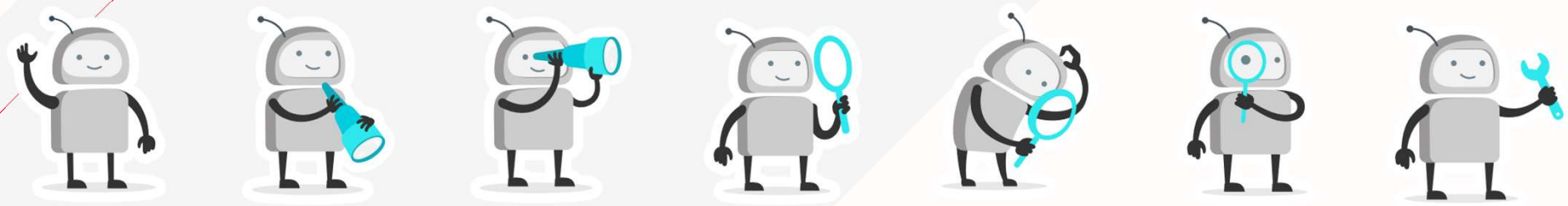
THANK YOU!



ADDITIONAL RESOURCES

1. **Akka Doc:** doc.akka.io
2. **Pekko Doc:**
   pekko.apache.org/docs/pekko/current/typed/guide/introduction.html
3. **Example Project:** gh/nathanielbellamy/aboutactors
4. **Diagrams w/ Excalidraw:** excalidraw.com





exp.oCon 2025