Why Microservices need a Full Stack

- Building the Right Thing, *Right*
- The Platonic Ideal of Waterfall never worked well…
- Agility is much more like Stoicism & Seneca
- Struggles of Software Development
  - The Elephant in the Standup
- A ‘Good’ system should be beyond robust, into antifragile
  - A great example is Netflix
- Granularity is key
  - Avoid pushing Boulders, try pushing Pebbles forward
- Even with better granularity, it was not a perfect world
  - It was hard work to extract services
    - People resisted until they *had* to externalise the service
    - Lots of questions to answer
      - Style of service
      - HTTP?
      - Messaging?
      - Patterns?
        - Circuit Breaker et al?
  - Speed of Testing, with Java, was terribly slow now there was a greater emphasis on ‘integration’ testing with the service apis
  - JSON Processing was just a pain in Java
- Should we switch everything to a new platform?!
  - Hell no!
  - Been here before
    - Let’s not create yet another monoculture
- Lunchtime Spikes showed the way
  - Each service could be migrated to the simplest solution for it.
  - Simplicity was an objective measure that the team could decide upon
  - The architects governed using principles, rather than onerous decisions
  - Human Comprehension was King
  - Mechanical Sympathy was Queen
    - And the queen often got her way
- Reduced resistance to service extraction by simplifying service exposure and consumption
  - Muon
  - A library for microservice concepts
  - Open Source, releasing soon
  - Polyglot System
- The BIG Payback?
  - Choice and Competition
  - *Speed!!*