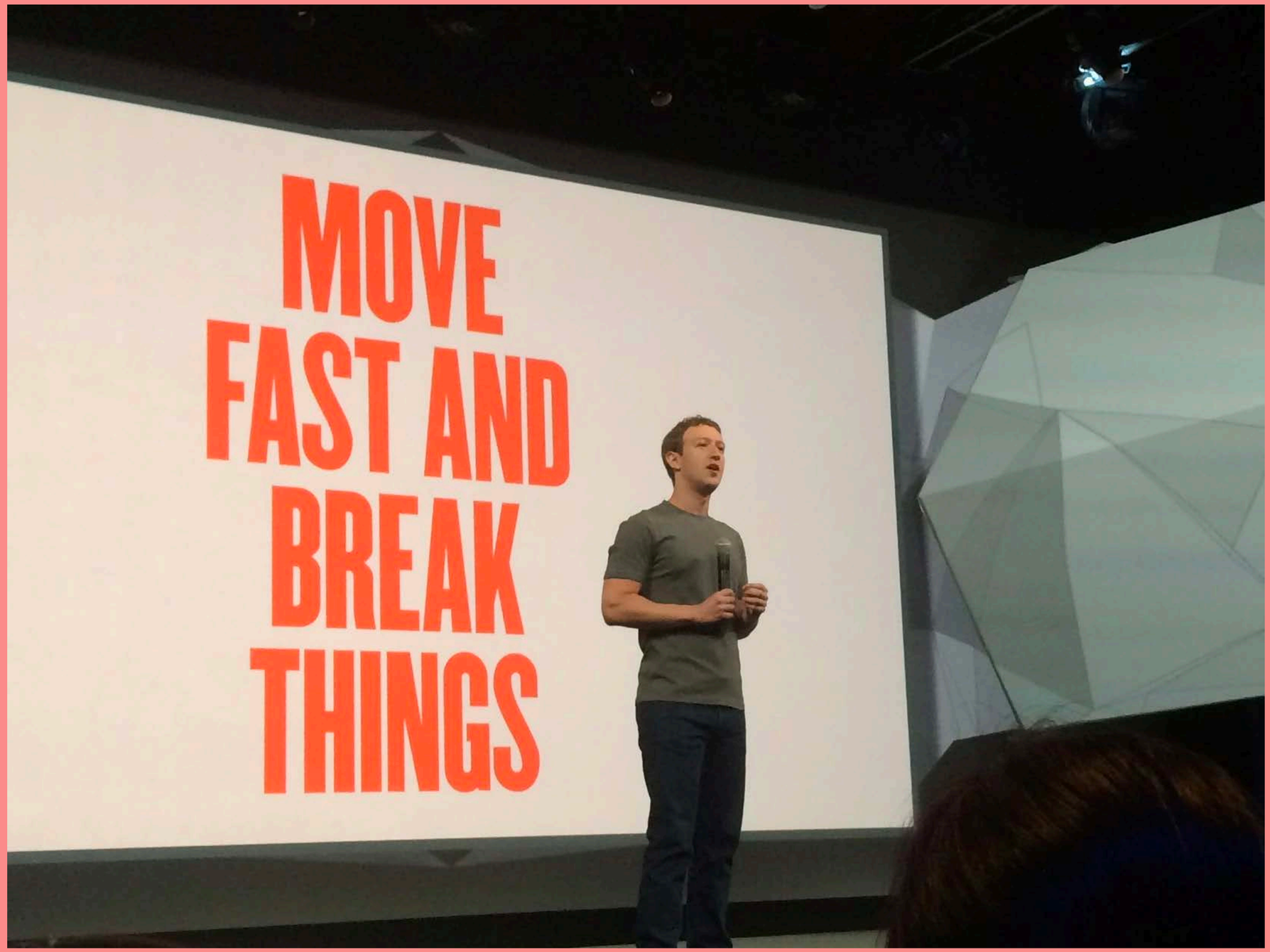
# Move fast without breaking things

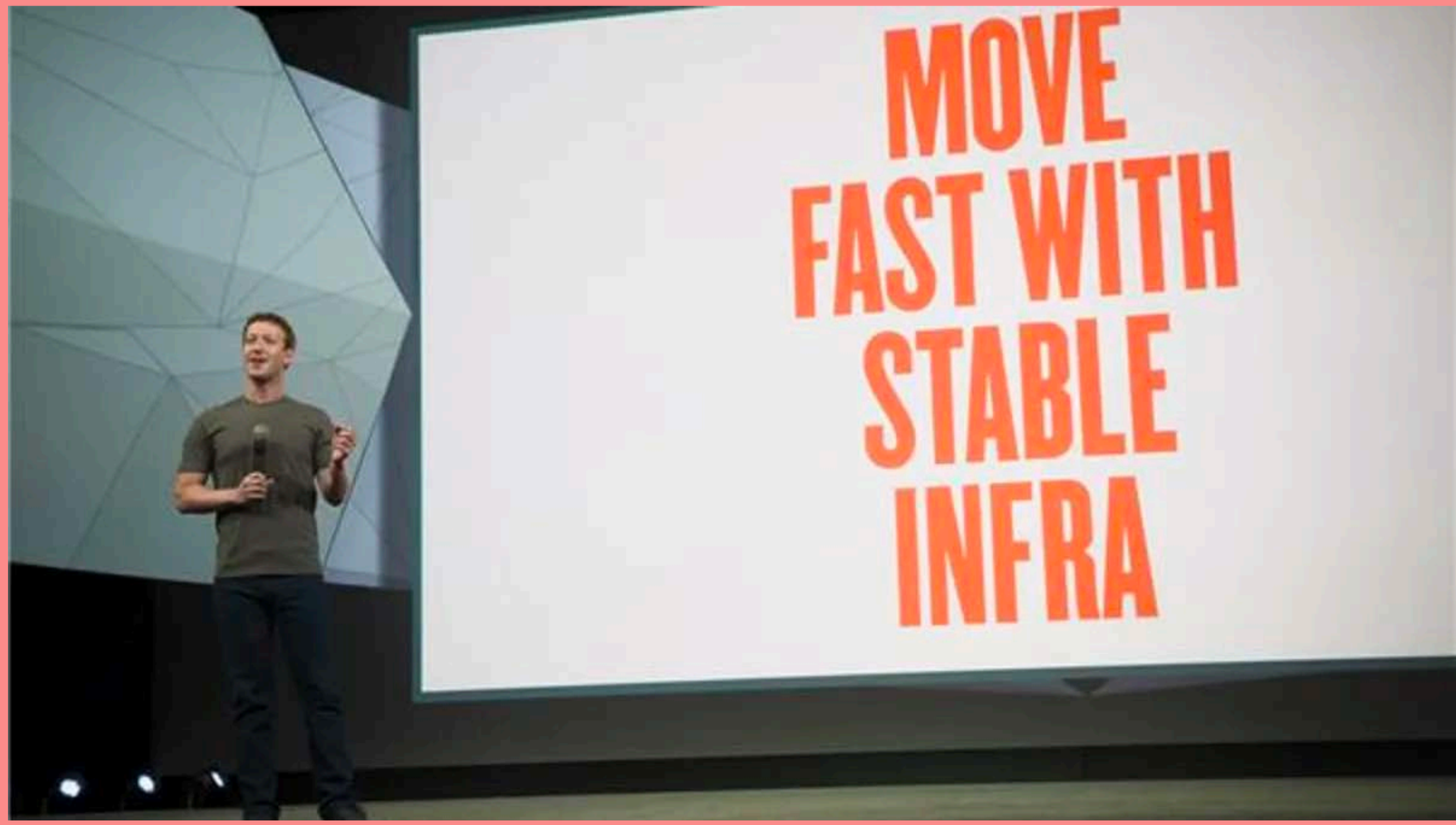
Sinan Kucukkoseler

shopify


MOVE
FAST AND
BREAK
THINGS

# Why?

- High quality threshold
- Experimentation
- Morale



**More than often, spending**
**%10 extra time will prevent %90 of**
**the issues you'll face later otherwise.**

# More about me:

Working in tech for 10 years.

Product minded engineer. Technical lead.

ThoughtWorks
New Relic
Shopify

Distributed Systems.

Complex Systems.

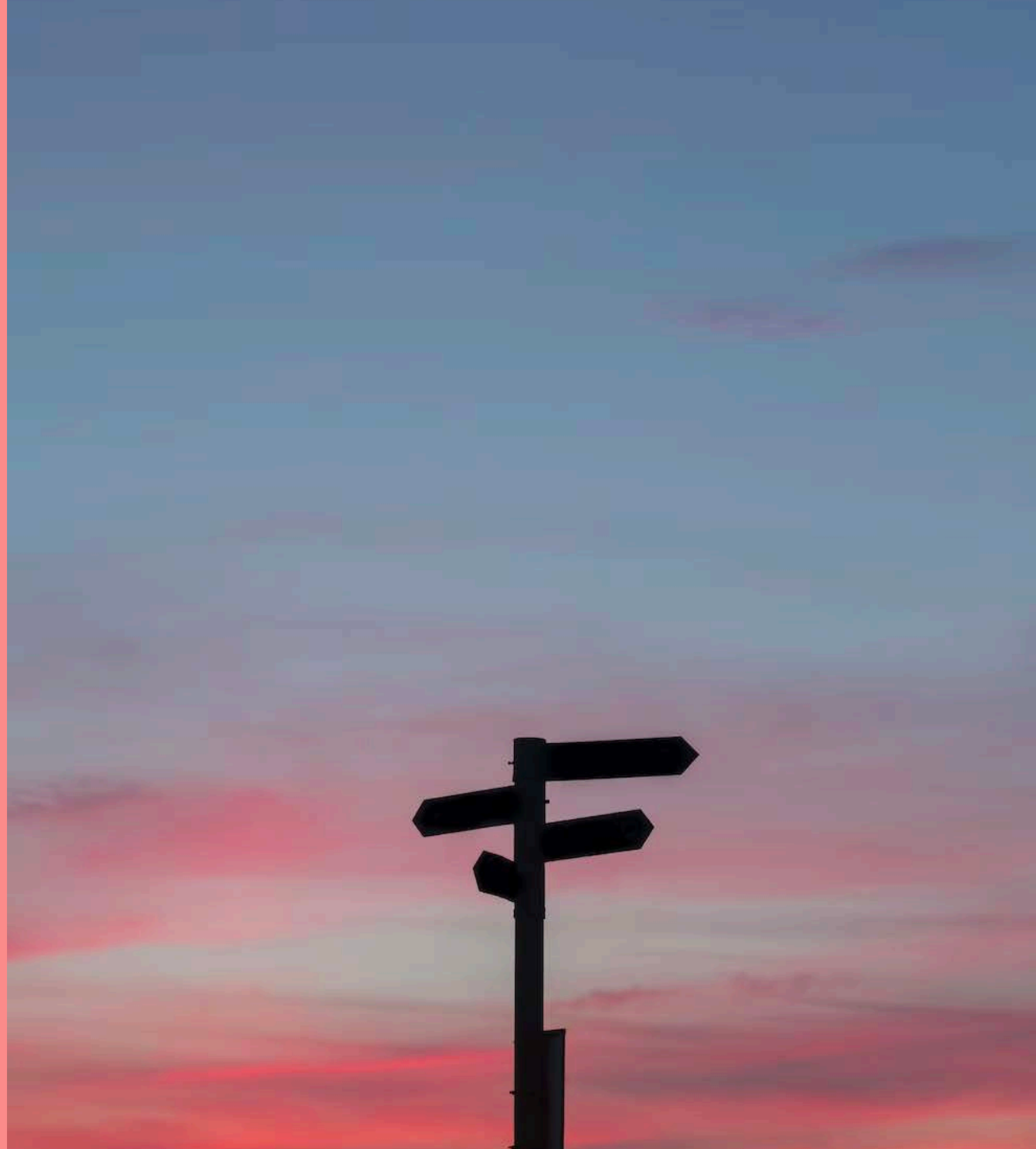
# Move fast without breaking things

Sinan Kucukkoseler

shopify

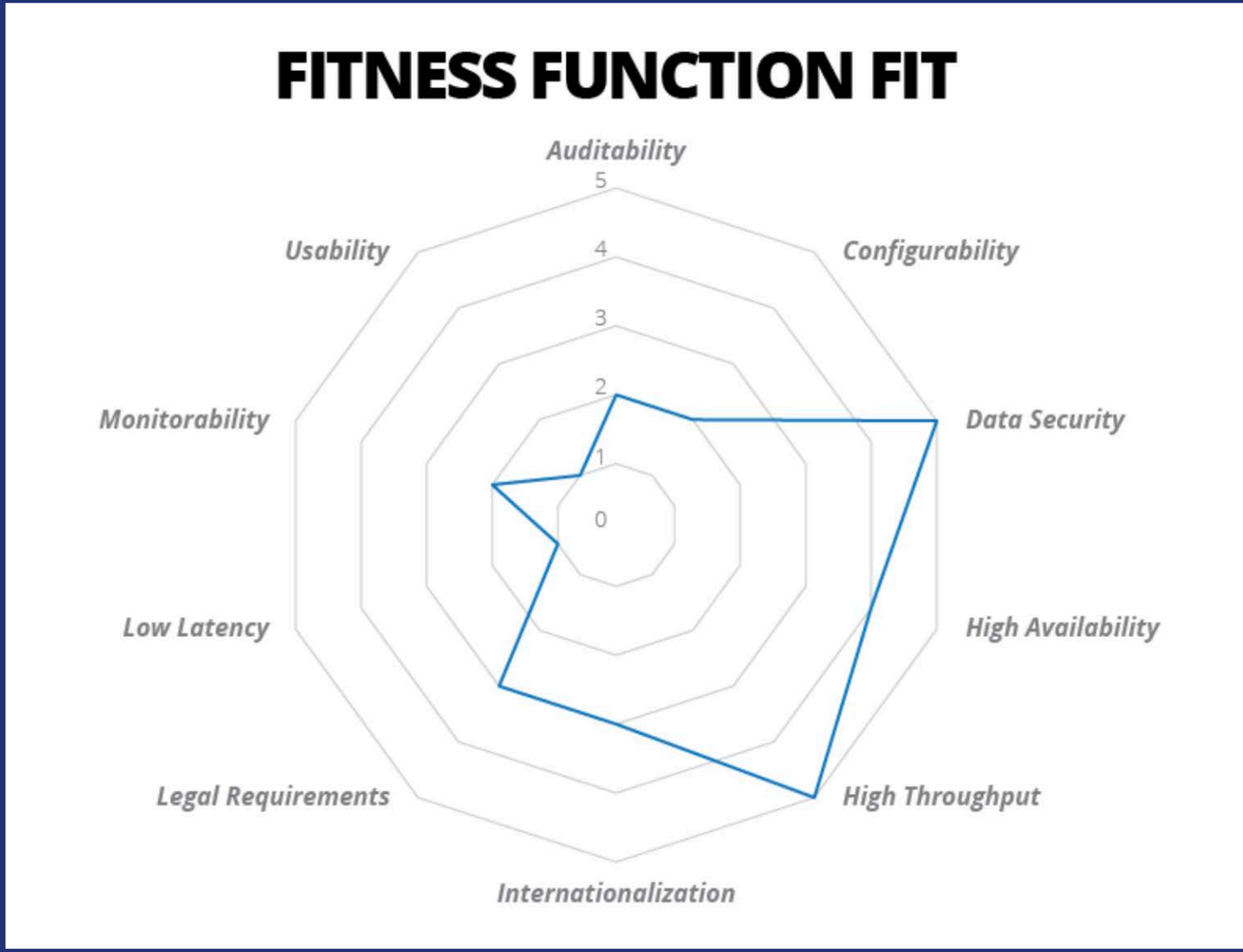
# How to plan

**A team should be doing the most important thing, at any given time.**



# System design + architecture

# Evolutionary Architecture




https://www.thoughtworks.com/en-es/insights/blog/microservices-evolutionary-architecture

# Keeping complexity in control

**More than often, spending %10 extra time will avoid %90 of the issues you'll face later otherwise.**

# Prioritisation

# High value + high complexity

Parts that:

- Connect the pipes and build the walking skeleton
- Are complex to build, risky
- Holds unknowns

# Integrations

- Downstream:

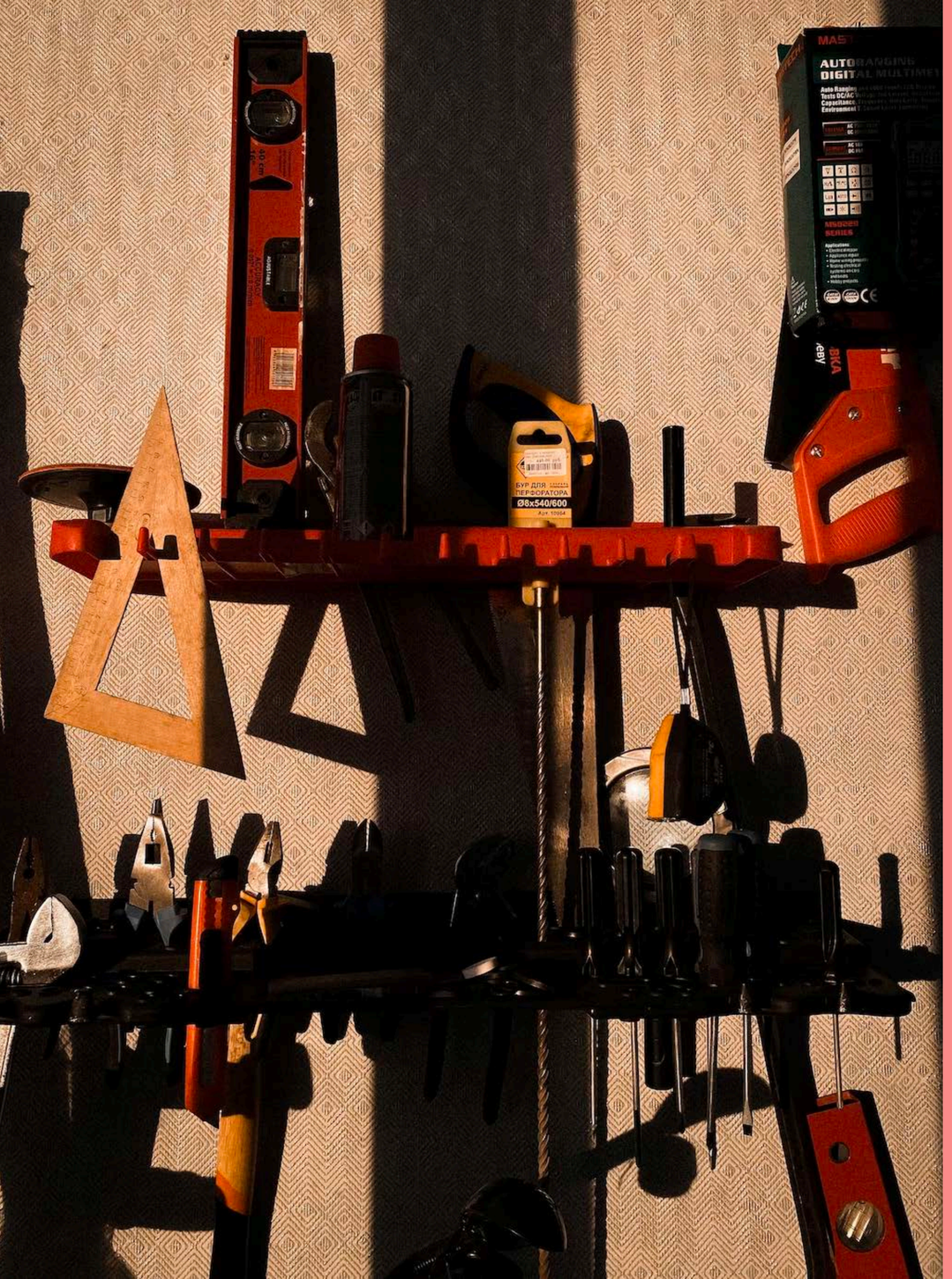
  timeouts, retries, back-off policies, circuit breaking.

- Upstream:

  bulkheads, load shedding, rate limiting.

# Testing

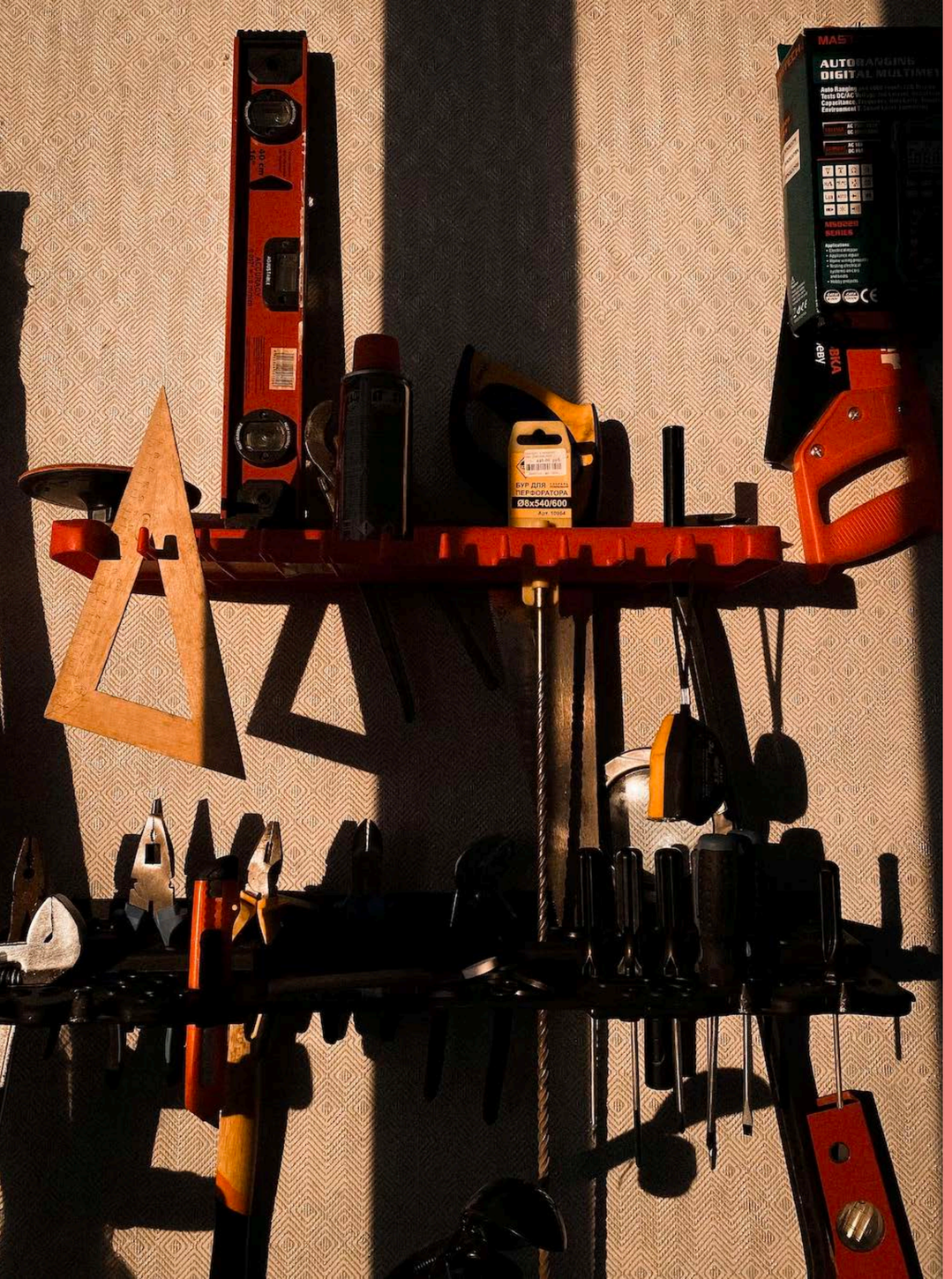
- Staging/test environments
- Load
- Diversity
- Shadow release




# Building for resilience

- Map possible problematic, error scenarios
  - Sudden increase in ingress load, db becomes bottleneck.
  - API calls gets throttled.
  - Caching cluster is unavailable.

- Decide how to react to these before they happen!
  - Have a run-book




# Building for resilience

- Auto-scaling + warm-up
- Immutability
  - Let us retry parts of our flow
- Compartmentalising
  - Lets us deprioritise less important, non-time sensitive tasks
  - Scale them separately
- Run-time configuration management

# Observability

- Add it while you build!!
- Start with the question:
  - *“How do we know if X’s working well?”*
    - Success rate of an API call / a process
    - Response time for a user request
    - # of requests served per second
- Start alerting from day 1!

# Adaptability

- Modes of behaviour
- Performance testing
  - If critical, start testing early. Use as a gateway
- Run game-days!
  - Manual testing or load simulation


# Re-cap

- Set your priorities clearly.
- Evolutionary architecture, optimise for less complexity.
- Walking skeleton. Tough tasks first.
- Secure integrations.
- Map out incident scenarios, create your run-book.
- Build optimising for resilience. Immutability + compartmentalise.
- Observability from day 1.
- Performance testing

# Thanks!

**sinan.kucukkoseler@gmail.com**
***linkedin.com/in/sinank***