Microsoft

ITARC Stockholm 2021

# How to architect something really big?

Clemens Vasters
Cloud Platform Architect
Azure Messaging
Microsoft

@clemensv

# Do not.

I’m absolutely serious, and here’s why …


Before we go there ....

# Architecture is Opinionated

# Architecture is Engineering

Dear Barry,

“How to architect something really big?” is a trap of a session title and you knew this when you asked me to do this talk.

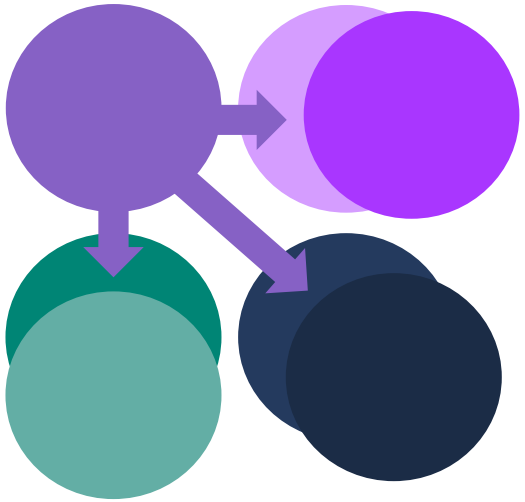
So here we are. There is no good, universal answer to a question that broad.

Yet we still have some 40 minutes to fill…

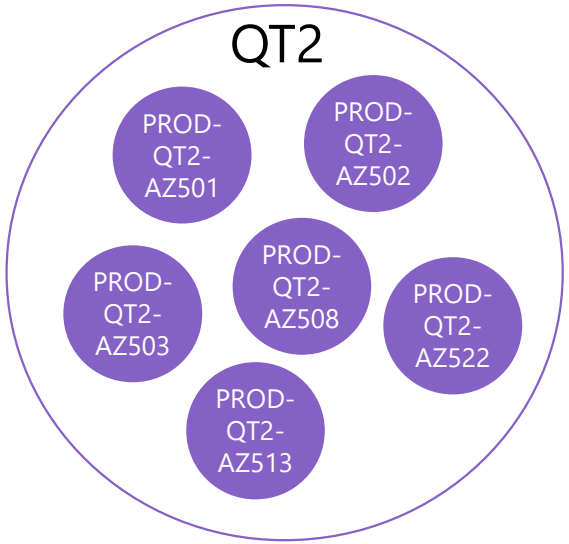
# Azure Messaging. Arguably Big.

| | | | |
|---|---|---|---|
| >140 Trillion<br>Monthly Requests across Messaging | 500 GB<br>Per minute at peak Written to storage by Event Hubs Capture | 99.9997%<br>Weekly success rate across Messaging | >168 PB<br>Monthly Data Volume |
| >60 regions<br>Running our services Ring 0 | 100%<br>Of the top 500 Azure customers use messaging services | >50,000+<br>VMs run our service | 4.5 Trillion<br>Daily Requests |

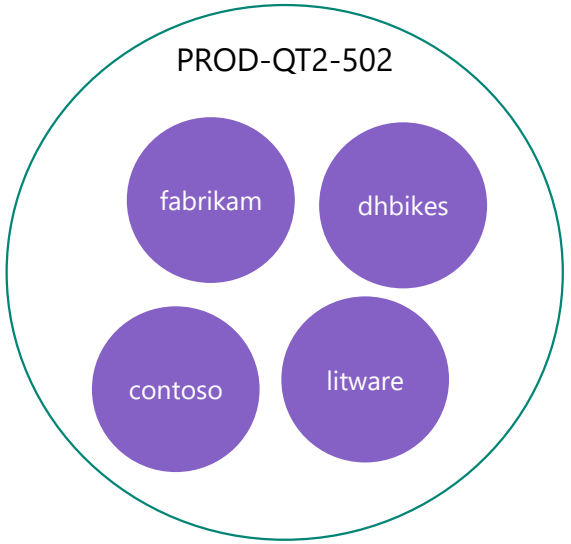
# What did we build?



7 operationally distinct service types.
Starting from one.




Regions.
Availability zones.
Clusters.




Cluster Tenants.
In-Cluster Isolation.

# Big things split things.

# Meet Your Adversaries

- Budget !!
- Business Model !!
- Capacity
- Latency
- Affinity
- Failure
- Thieves
- Idiots

# Capacity

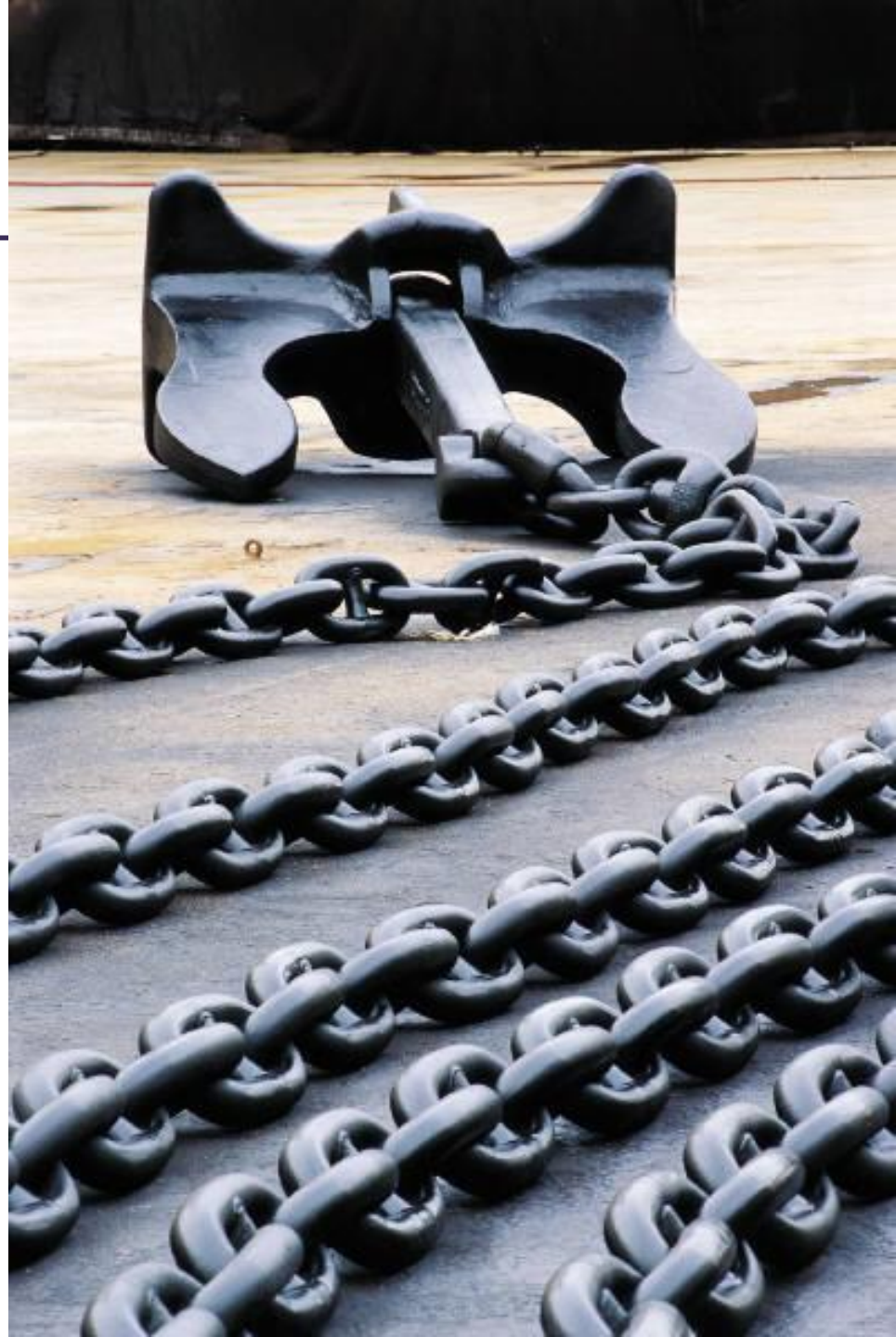
- ‘Capacity’ defines the limits of how much information can be held, transferred, or processed by a system during some period
  - Machine instructions that can be processed within the period to compute some output based on some input
  - Amount of information that can be held in the storage container at any given time
  - Amount of information that can be transferred into and out of a scope of location within the period
- If capacity were ‘unlimited’, all information would be ubiquitous

# Latency

- ‘Latency’ is the time it takes to accomplish a certain task
  - Compute an output based on some input using an algorithm
  - Store information into or retrieve information from some kind of container
  - Transfer information between different system scopes and/or physical locations
- If latency were ‘zero’, all information would be instantly available everywhere
- Latency is Software’s Friction

# Affinity

- ‘Affinity’ is the degree to which a particular set of information is bound to a scope or location
  - Affinity of original and intermediate state to a particular sequence of processing while producing a desired output
  - Affinity of information to a particular container due to size or transfer latency
  - Affinity of transfer to a particular communication route
  - Affinity to a particular physical device or location (including someone’s pocket)
- Affinity is the biggest inhibitor for scaling out



# Failure

- ‘Failure’ is any condition that interrupts or otherwise adversely affects the common, planned flow of information processing
  - Programming errors
  - Physical failure of equipment
  - Exhaustion of system capacity
  - Environment  conditions
- ‘Permanent Failure’: ‘Call the Doctor’
- ‘Intermittent Failure’: ‘Wait for it to go away’

# Thieves and Idiots

- "Thieves and/or idiots"
  - Security protects against thieves
    - Identify who you're dealing with (authentication)
    - Only allow them to do what they're entitled to do (authorization)
  - Safety protects against idiots
    - Make sure information is safe against inadvertent destruction ("Oops. I didn't mean to do that")
- Security and Safety are about undesired interactions with people.

# Considerations.

# Business

- What are you selling?
- How does a new feature impact what you are selling?
- What does the new feature do for your bottom line?
- Might that new feature hurt your bottom line?

# Blast radius

1

- How much impact can a bug have?
- How much impact can a dependency issue have?
- How much impact can an attack have?
- How much impact can an operator have?

# 2

# Tenancy and Partitions

- All "really big" systems have tenants.
  - Users, Accounts, Namespaces, etc.
- Natural partitioning boundary
- Consider workflows and objects as tenants
  - Short-lived tenants
  - Dynamic allocation

3

# Failures and Disasters

- Failures will happen. Big ones, too. Sorry.
  - (Hi Facebook!)
- How much are you willing to pay for what kind of risk mitigation?
  - Lots of disaster scenarios are the boogeyman.
  - No, you can't run platform better than a hyperscaler
  - Availability zones are GREAT tech
- Systems don't die clean deaths. Your disaster drills are a lie.

# 4

# Automation

- Radically automate (almost) everything
  - Every automated step is less of an opportunity for someone making a manual mistake
  - You can't scale globally with manual steps
- Do not hand the reliability predictability of your system to a load-adaptive algorithm unless you really understand what it does and why.

# 5

# Observability

- Instrument deeply: "weird" stuff will happen
- 0.005% reliability gains matter at scale
- The root cause is often elsewhere

# Truth and Lies

- You know the system reality. Does your user? Do they need to?


Microsoft
Thank you