# Controlled Chaos:<br>Taming Organic, Federated Growth of Microservices

# Southwest meltdown

July 20, 2016

- One of 2,000 routers fails
- All monitors green
- Billions of packets in, 0 out
- 30 min to discovery
- 12 hours rebooting
- 5 days to full recovery

1
Router

5
Days

$80M
Losses

-$3.4B
Mrkt Cap

Source: https://www.washingtonpost.com/lifestyle/travel/airline-computer-outages-like-deltas-are-bound-to-repeat-themselves-heres-what-to-know/2016/08/11/578a83cc-5d8d-11e6-9d2f-b1a3564181a1_story.html

Image: goodfreephotos.com

GLASNOSTIC

# Mission control

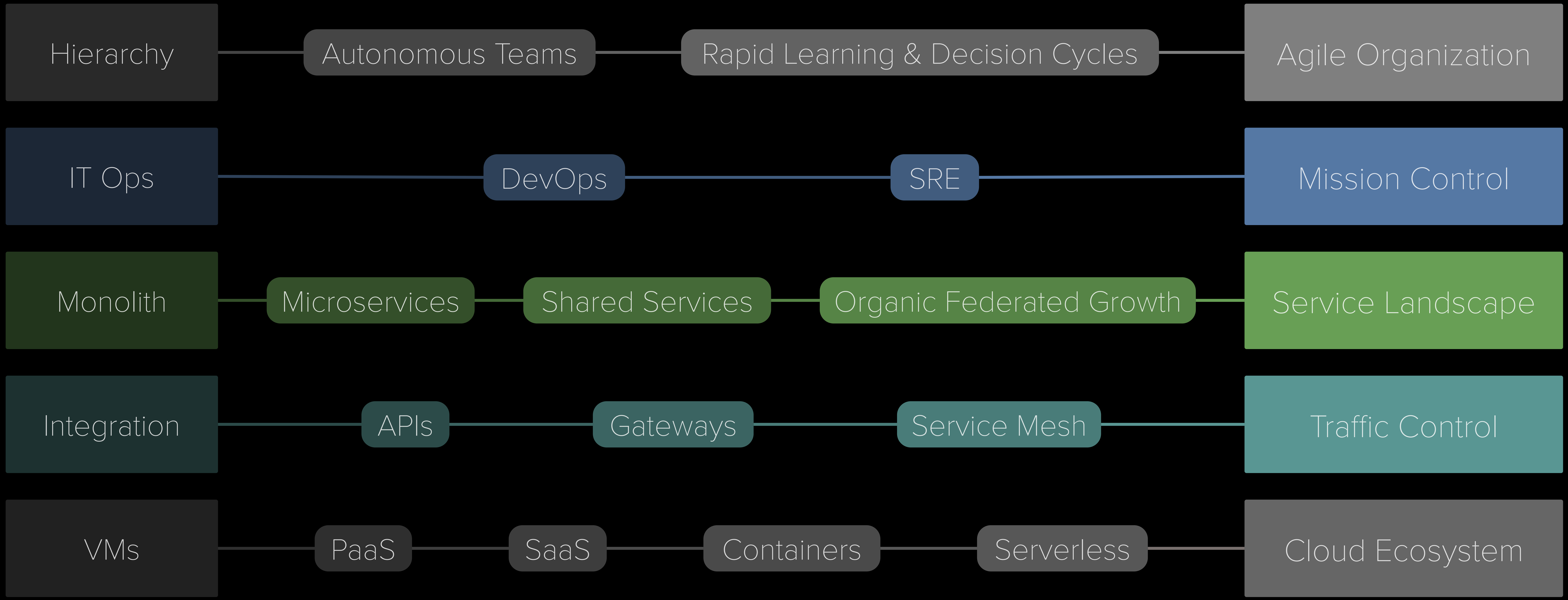
## Service landscape
## Behaviors
## Operational patterns

Tobias Kunze
Co-Founder & CEO

tobias@glasnostic.com
@tkunze

Image: NASA

# The agile operating model



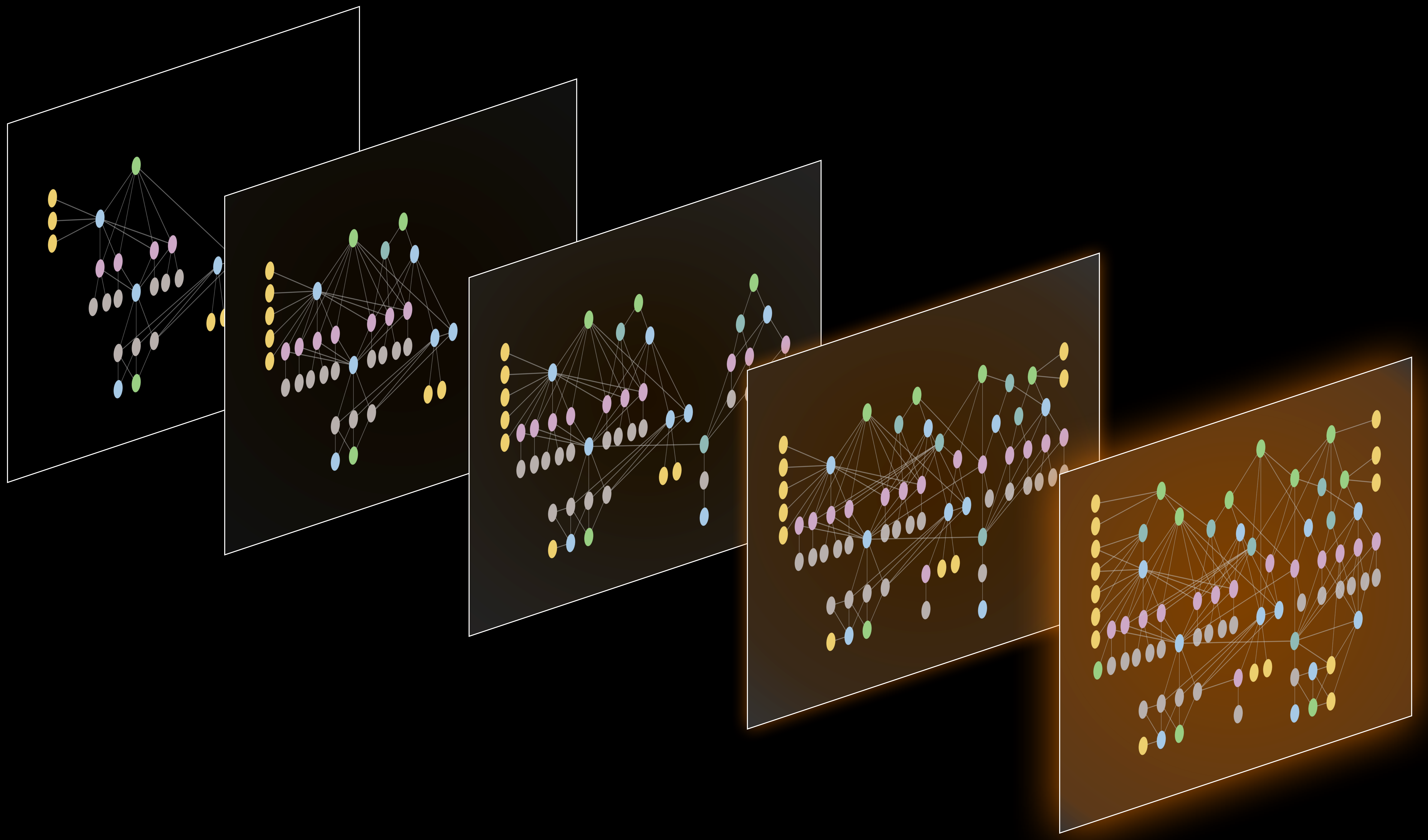
Hierarchy — Autonomous Teams — Rapid Learning & Decision Cycles — Agile Organization

IT Ops — DevOps — SRE — Mission Control

Monolith — Microservices — Shared Services — Organic Federated Growth — Service Landscape

Integration — APIs — Gateways — Service Mesh — Traffic Control

VMs — PaaS — SaaS — Containers — Serverless — Cloud Ecosystem



# Security: evolving topologies, ephemeral actors

Loss of perimeter
Loss of blueprint

Changing identities
Volatile behaviors

Fundamentally new challenge

Image: FIXME

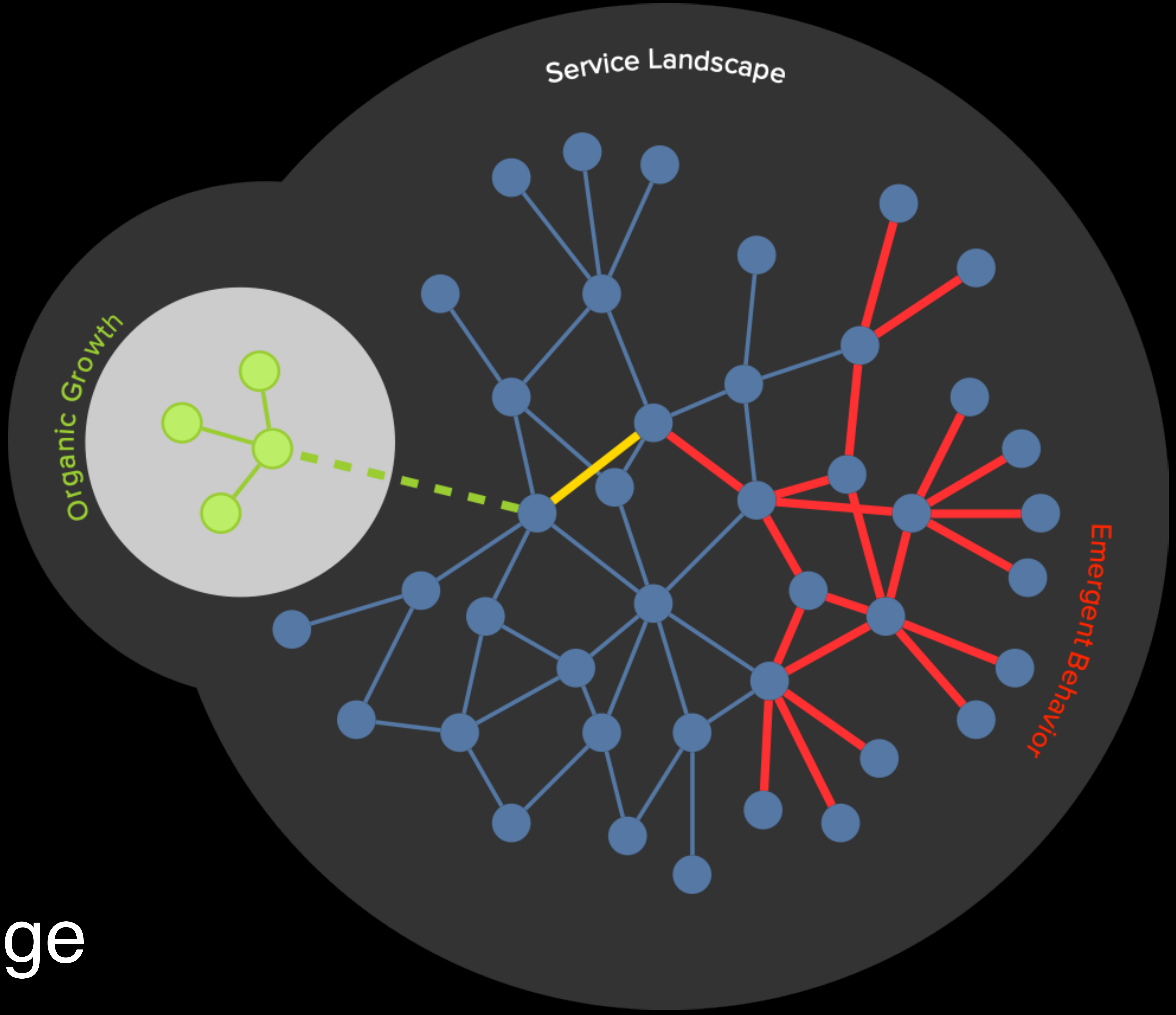
# Stability: complex emergent behaviors

Large-scale

Complex

Non-linear

Unpredictable



Fundamentally new challenge

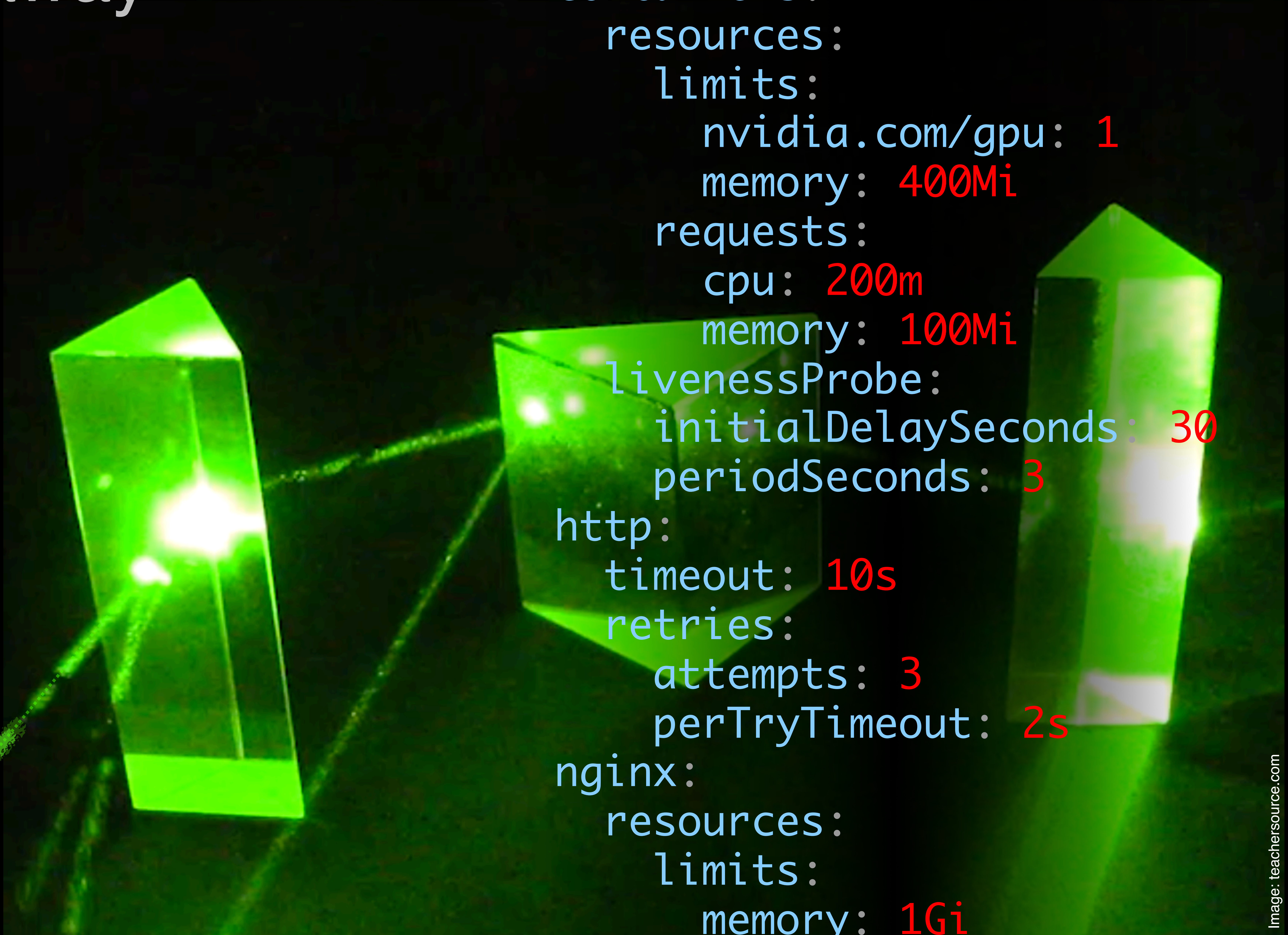
# Can’t engineer away

Resource limits

Scaling behaviors

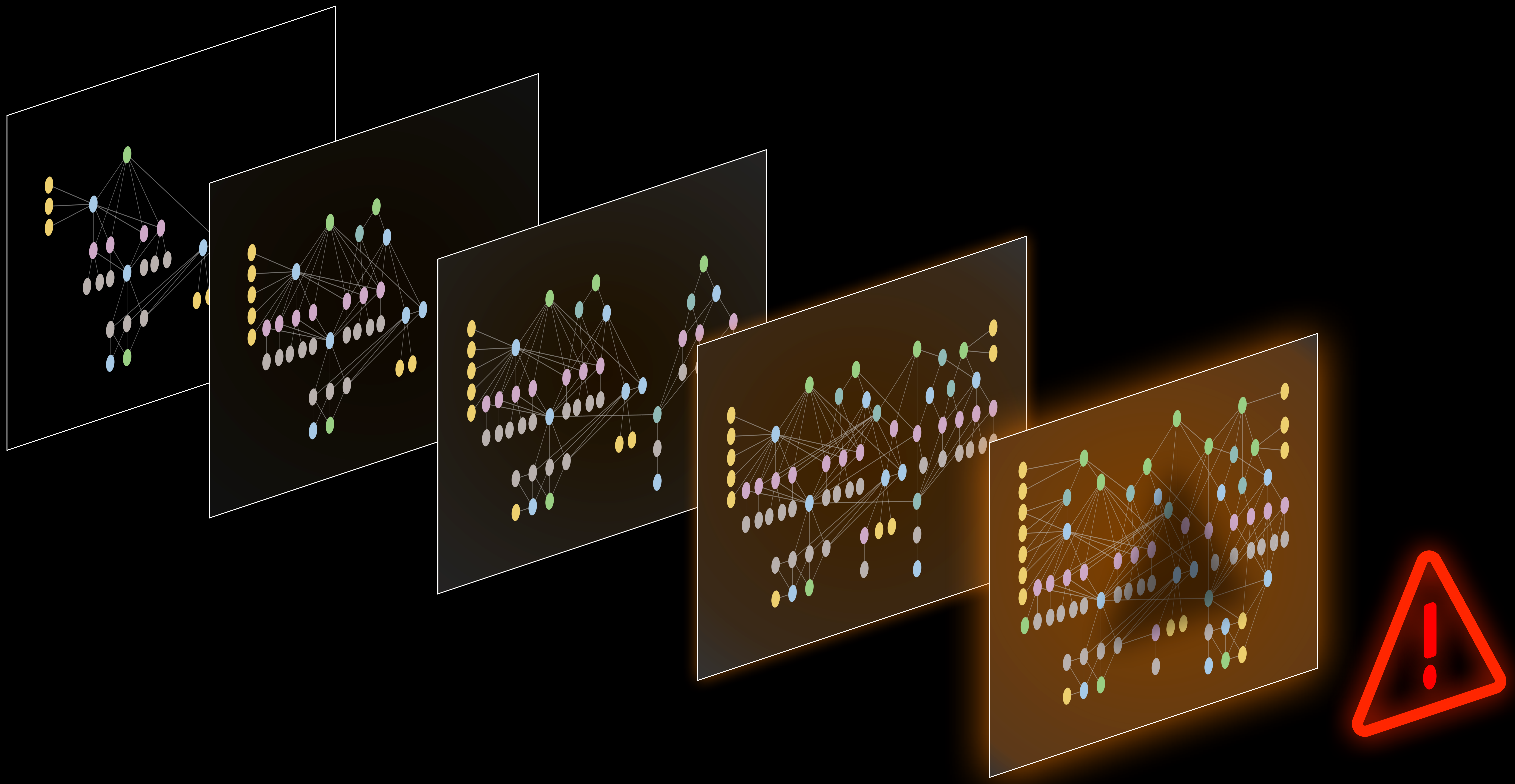
Request behaviors

Pool sizing



```
replicas: 3
spec:
  containers:
    resources:
      limits:
        nvidia.com/gpu: 1
        memory: 400Mi
      requests:
        cpu: 200m
        memory: 100Mi
    livenessProbe:
      initialDelaySeconds: 30
      periodSeconds: 3
  http:
    timeout: 10s
    retries:
      attempts: 3
      perTryTimeout: 2s
  nginx:
    resources:
      limits:
        memory: 1Gi
```

Image: teachersource.com



# Environment over code

Source: nats.aero

Successful missions are run.

Image: NASA

# Coping strategies

Do nothing

Monitor nodes

Trace requests

# Service mesh

Natural landscape evolves faster than baroque YAML

Source: gagliardiphotography.com

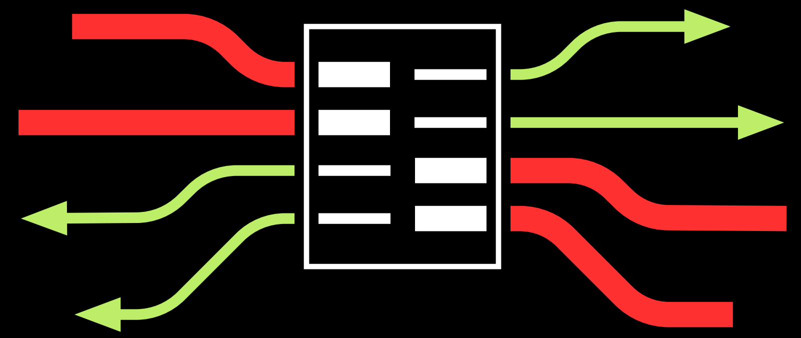
# Golden signals

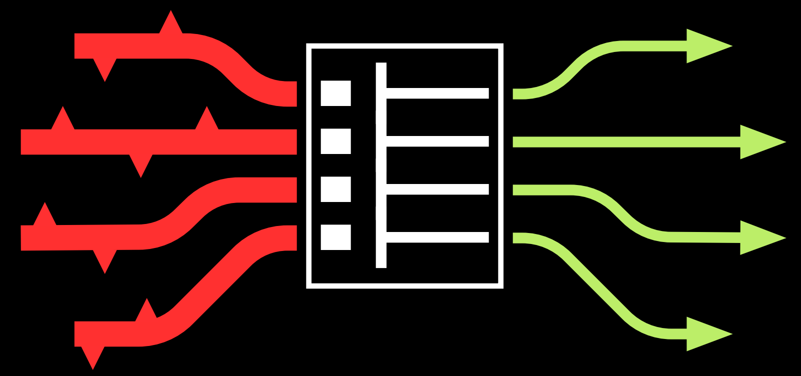
Requests
Latency
Concurrency
Bandwidth

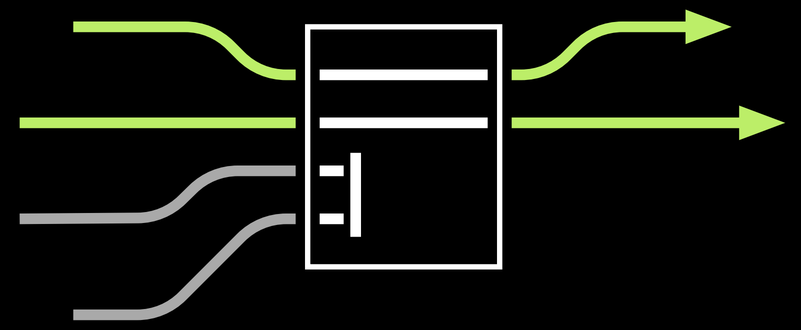
Image source: bbc.co.uk

# Operational patterns

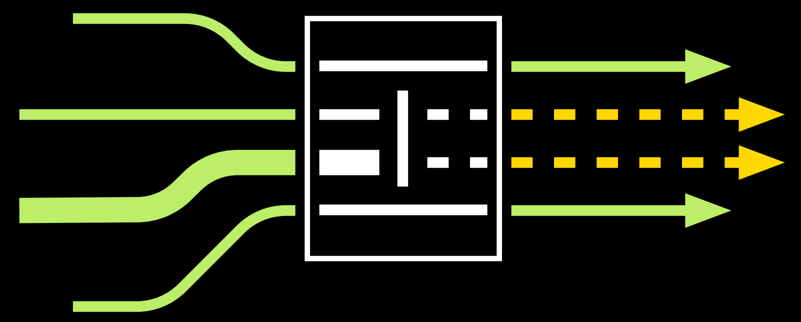
## Control Systemic Failures

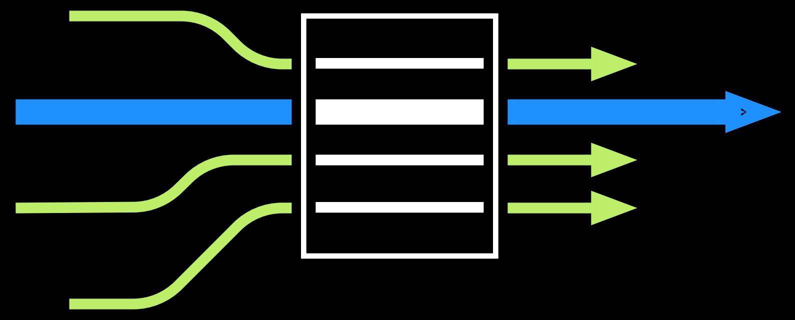

Bulkhead


Backpressure

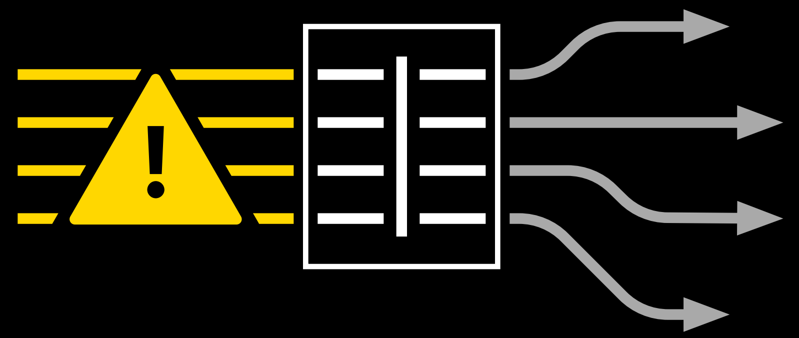

Segmentation

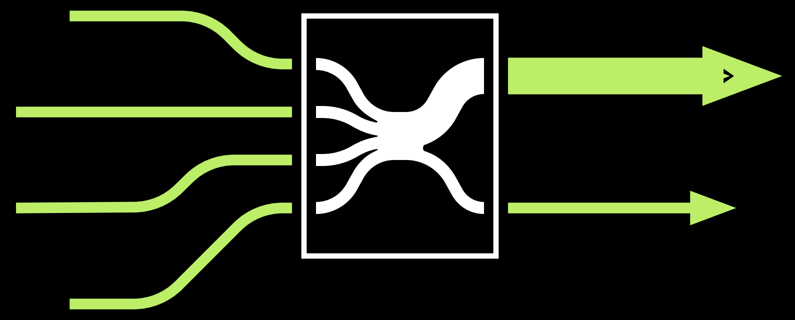
## Assure Performance


Circuit Breaker

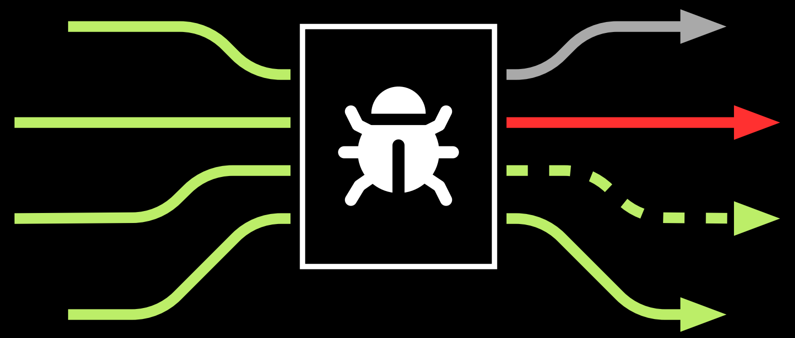

Quality of Service

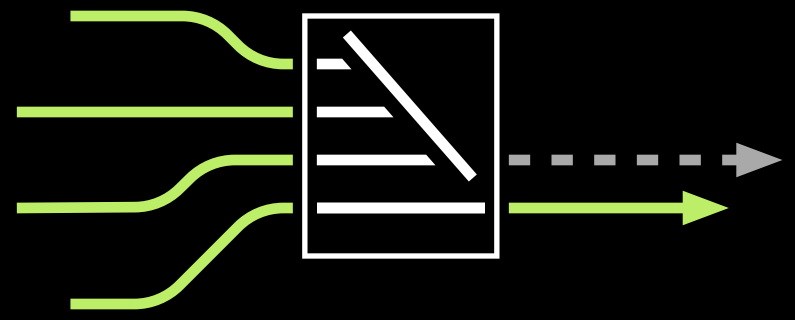
## Deploy with Confidence

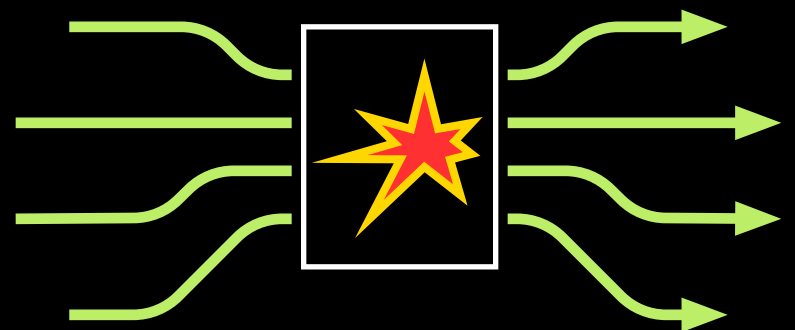

Quarantine


Canary

## Build Resilience


Fault Injection


Brownout


Blast Radius

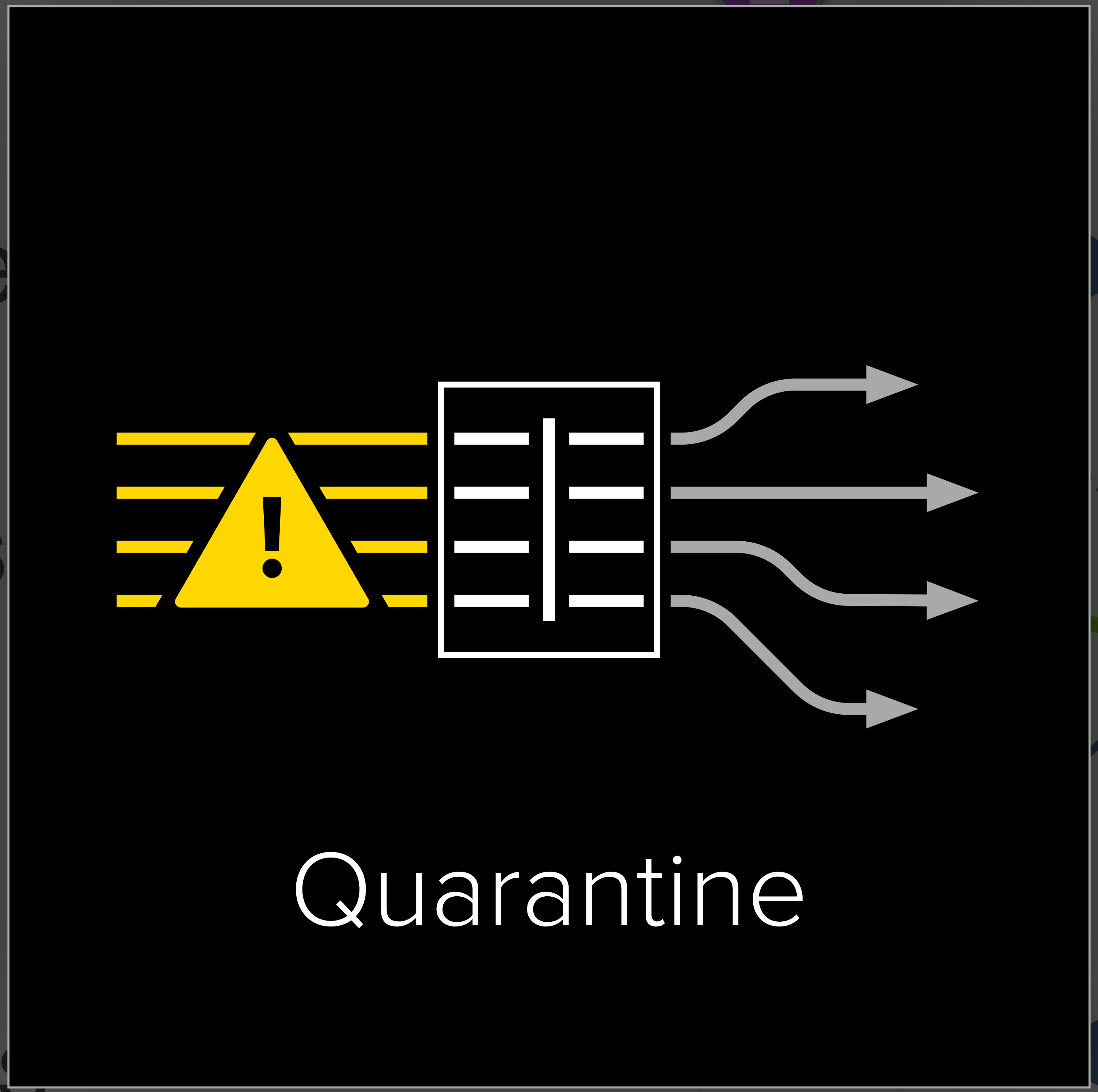
# Ex. 1: cache thrashing

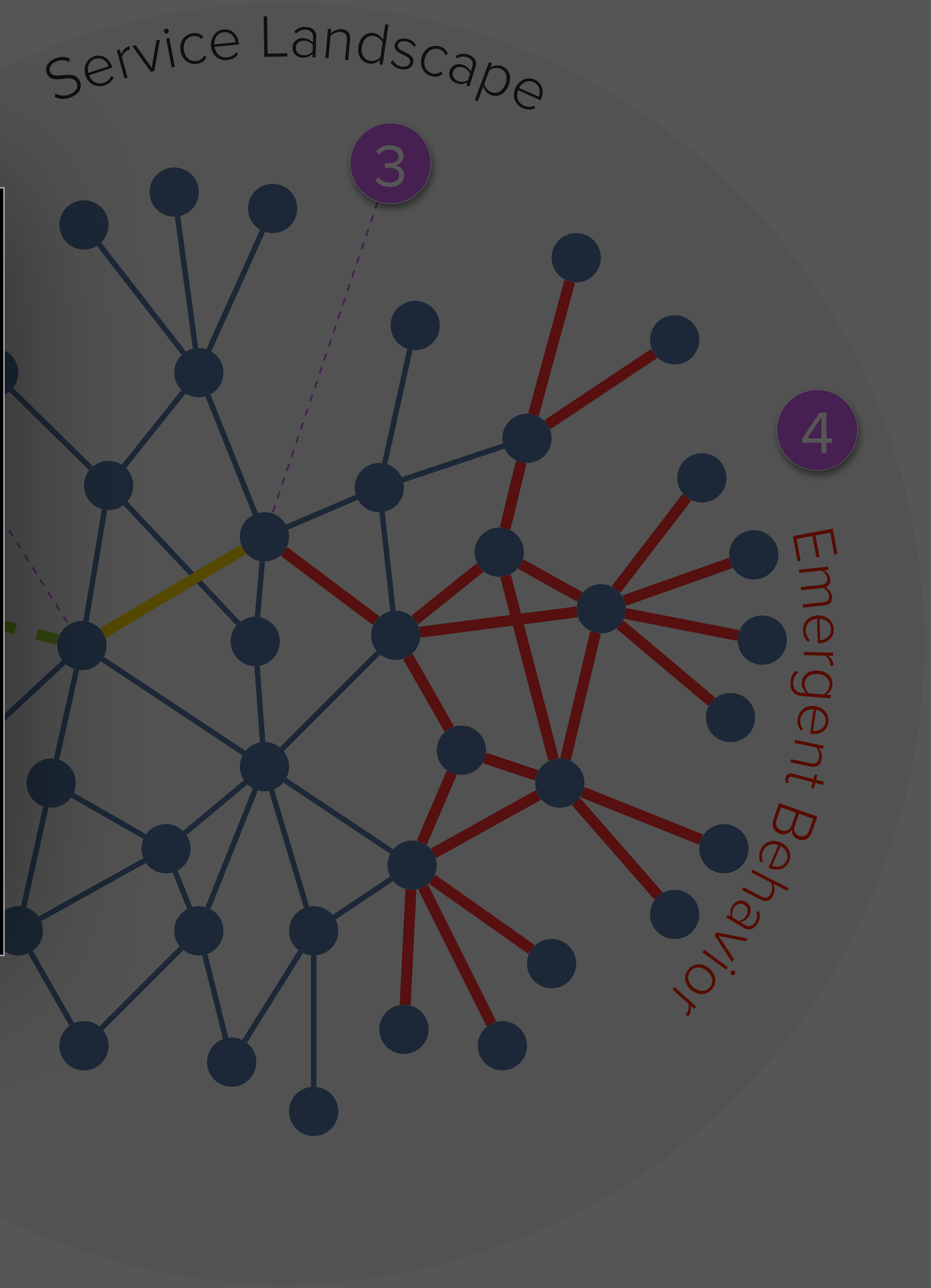
Behavior

1. New organic growth
2. Upstream fan-out change
3. Shared cache thrashes
4. Wide, unspecific slownes

Remediation

1. See widespread slowness
2. Identify bottleneck
3. Correlate with deployment
4. Quarantine deployment





# Ex. 2: cascading failure at Target

## Behavior

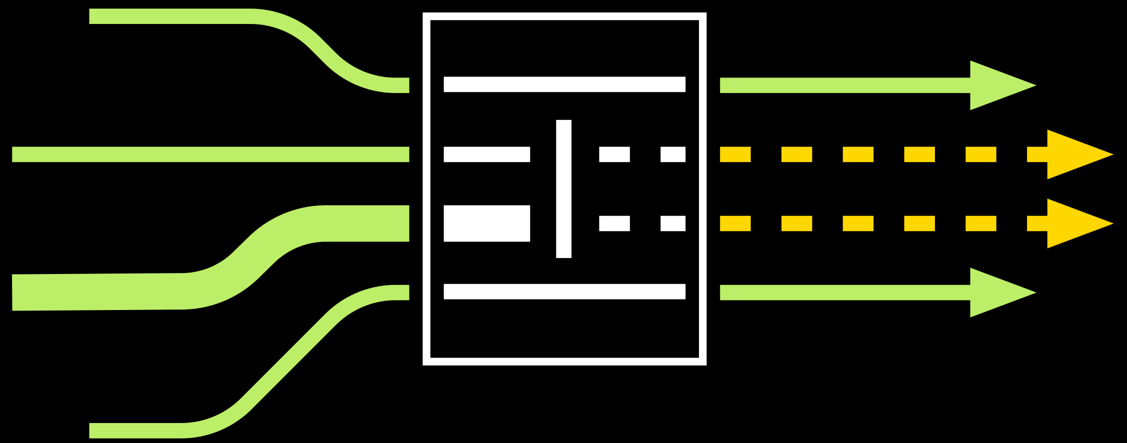
1. K8s flip-flo

## Remediation

1. S
2. B

eak

Backpressure



Circuit Breaker

Intermittently availa

Migrate

Service

Sidecar

8 CPU spike

OpenS

K8s

Do

9 Starvation

Network outage 2

Unhealthy 6

Node

10 Unhealthy

Maintenance 1

Cluster

Source: https://medium.com/@daniel.p.woods/on-infrastructure-at-scale-a-cascading-failure-of-distributed-systems-7cff2a3cd2df

# Ex 3: security breach

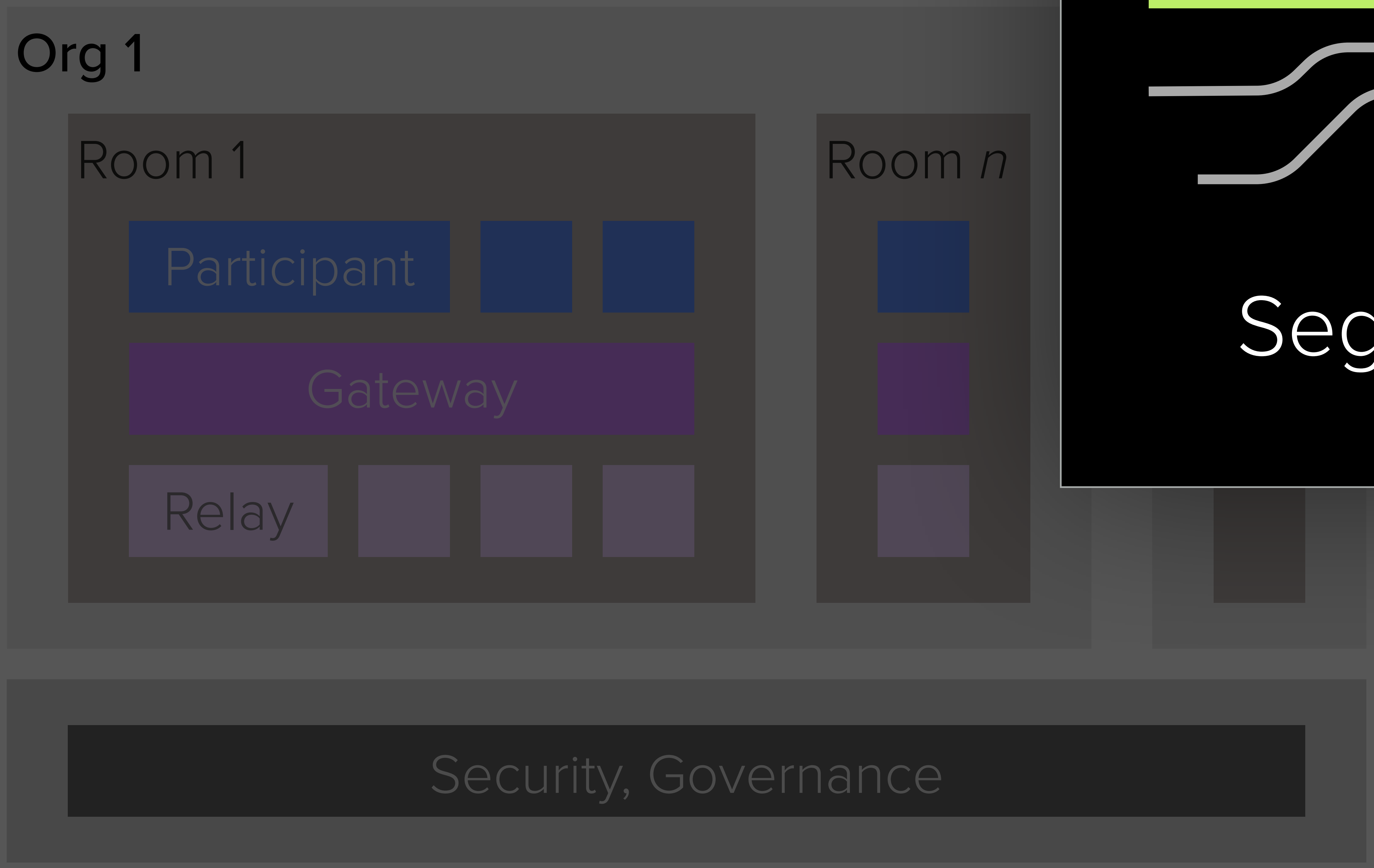
Behavior

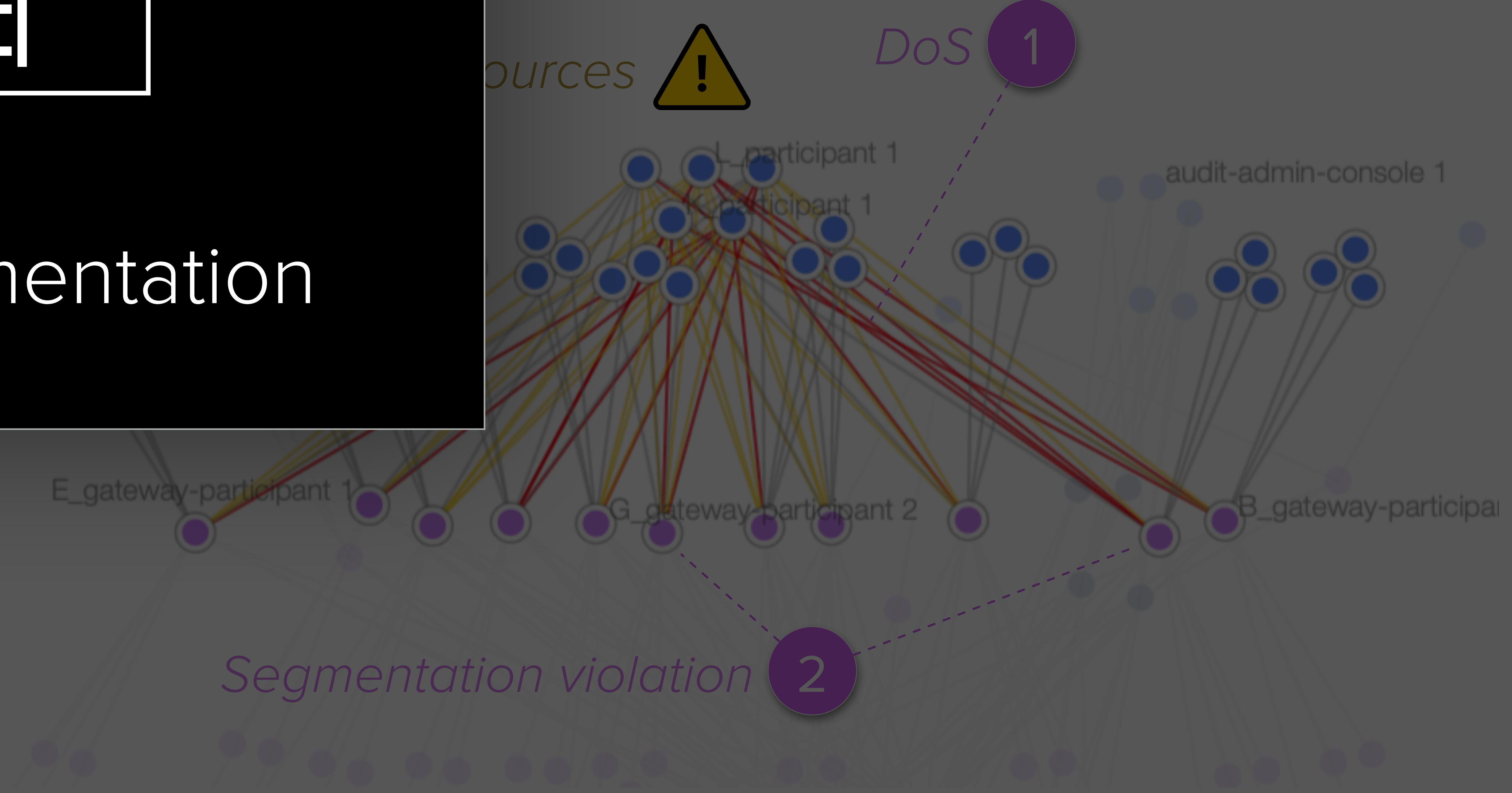
1. DoS
2. Segmentation violation

Remediation

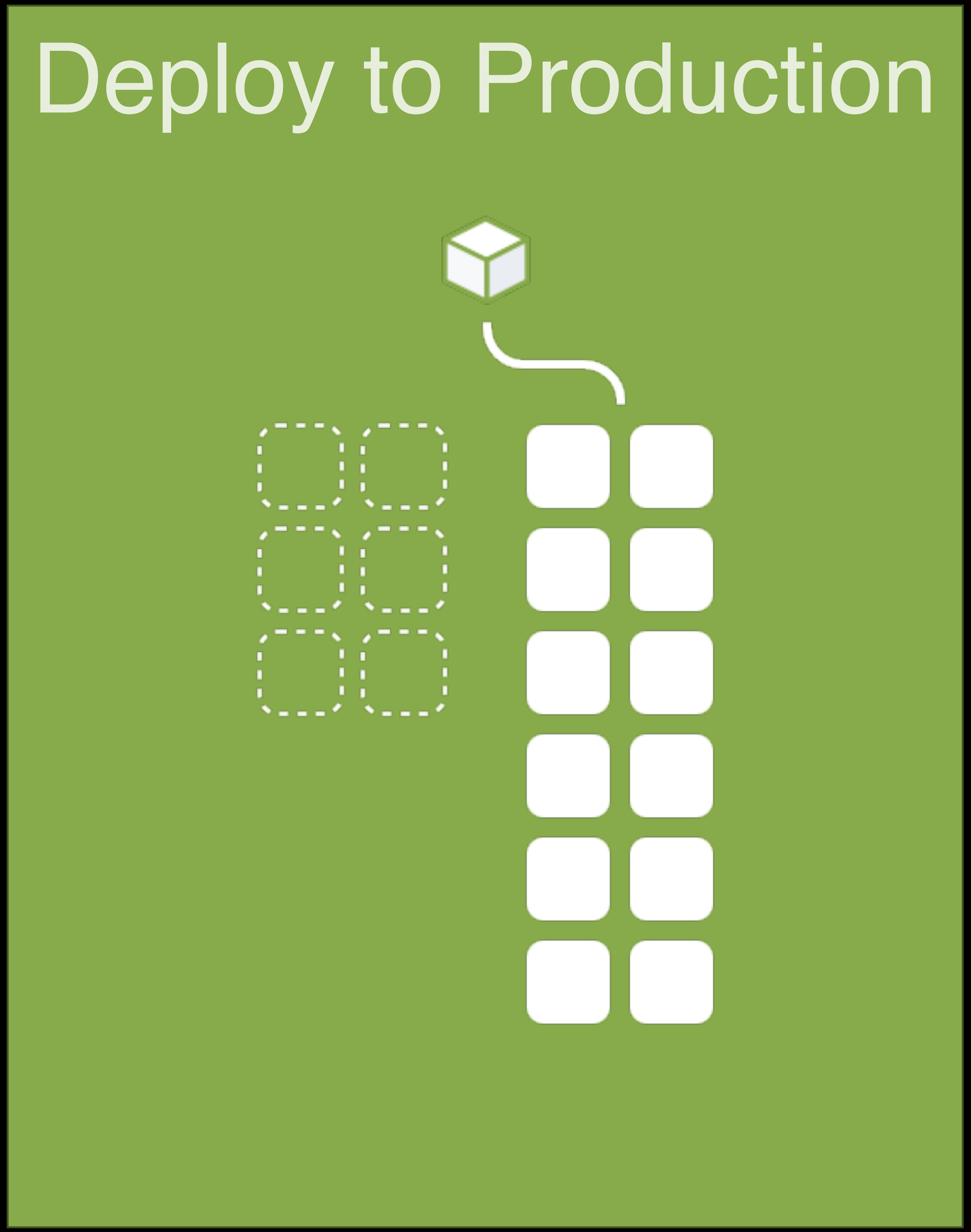
ources

Segmentation



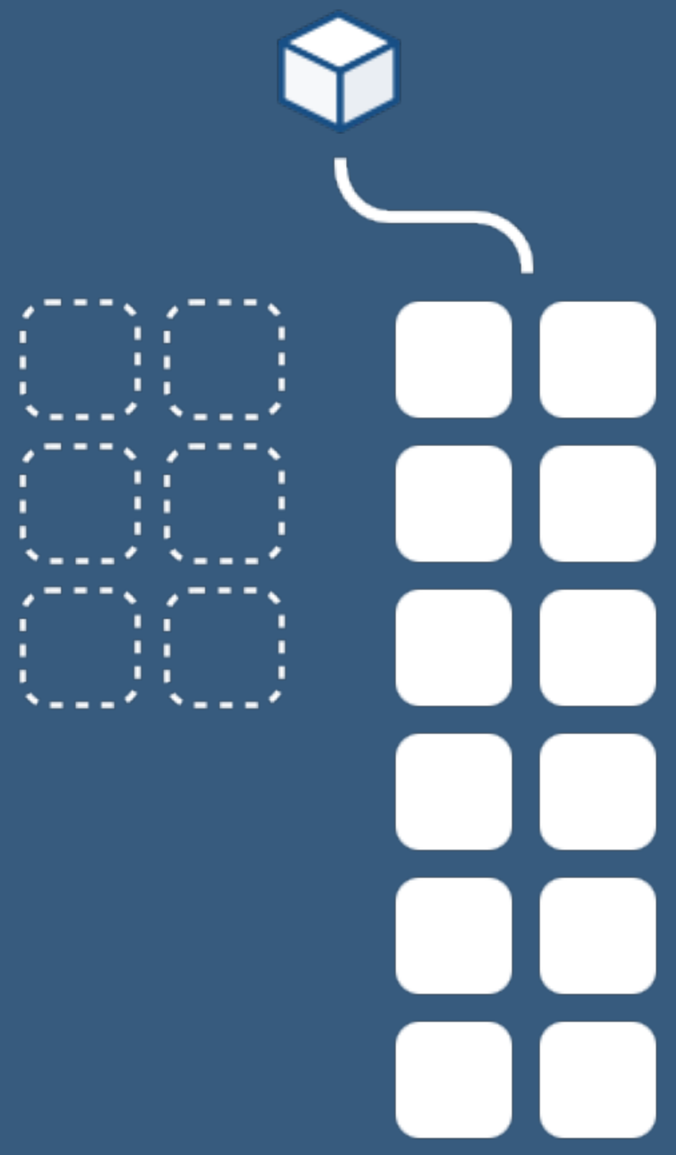


# Runtime control examples

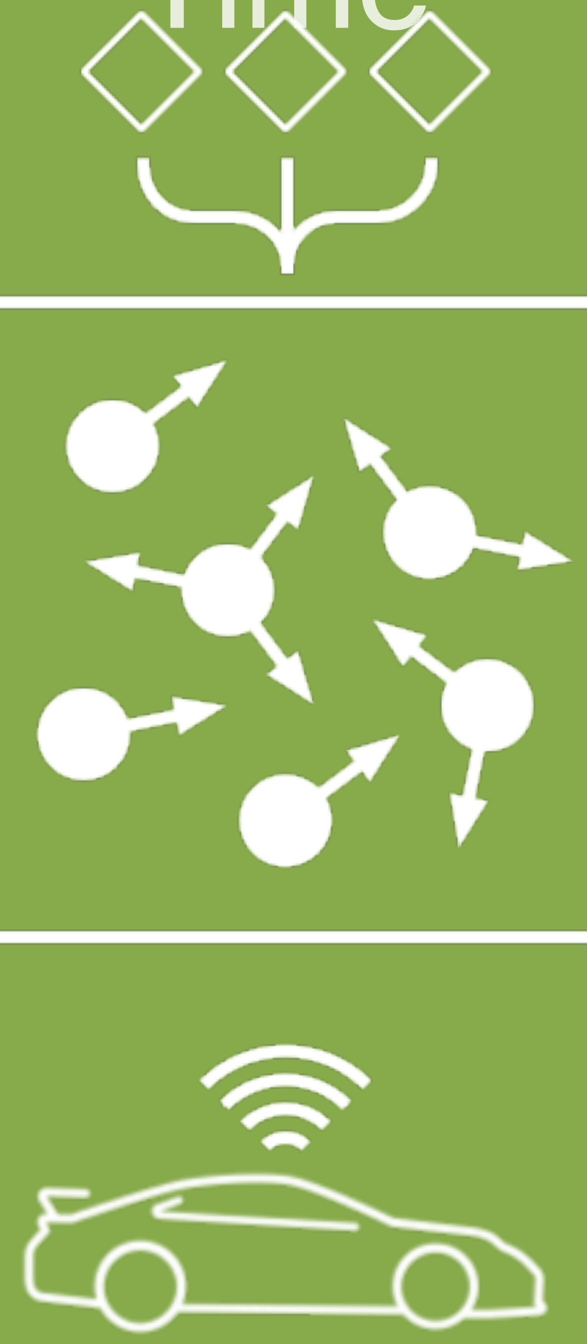


# Runtime control examples

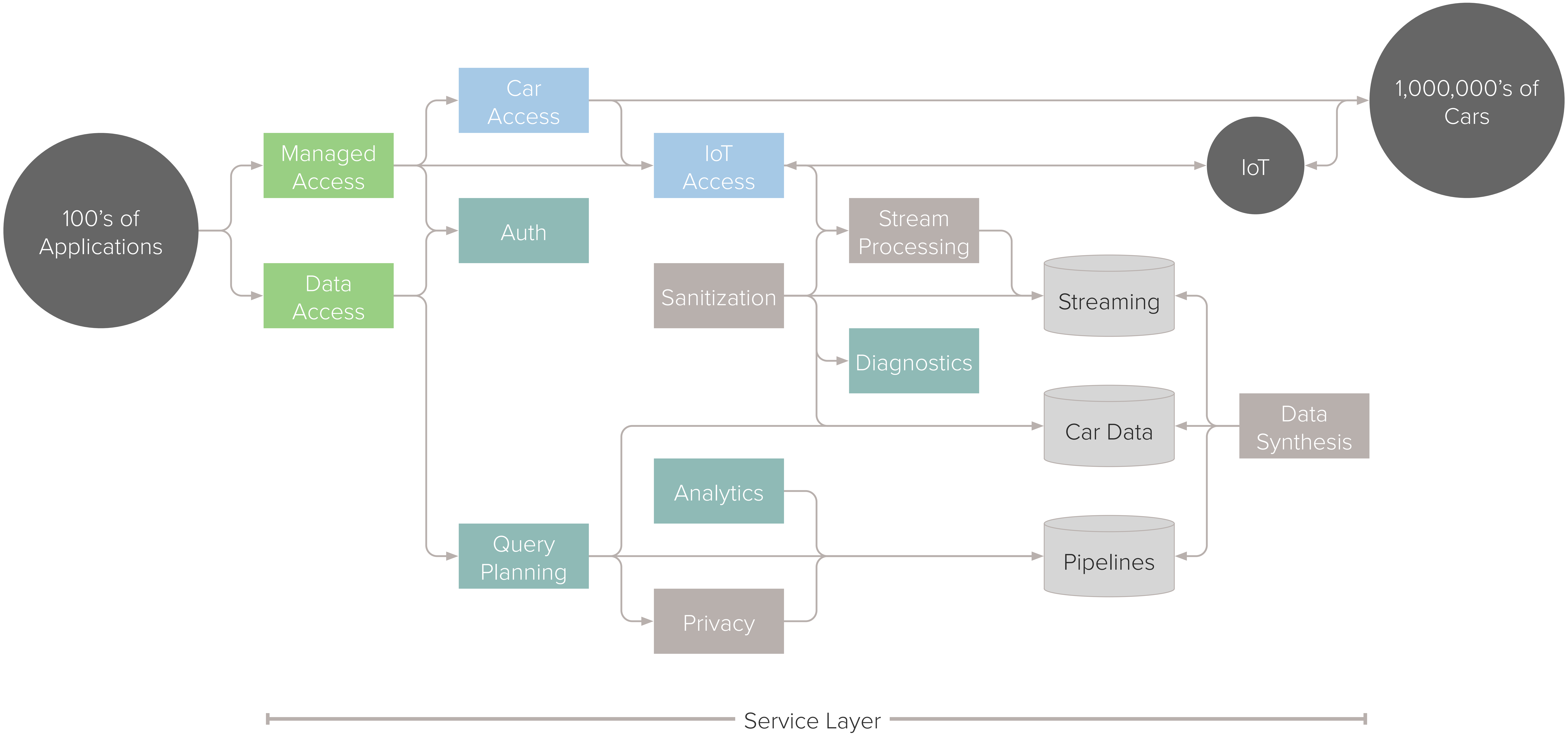
## Deploy to Production



## Architect in Real Time

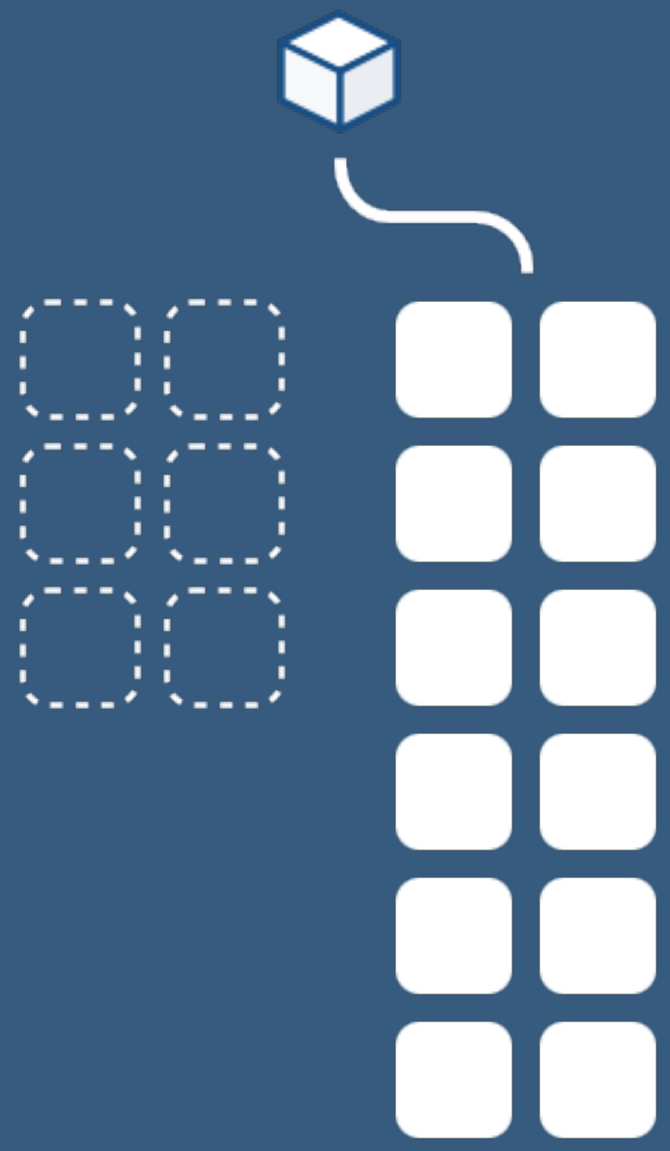


# Architect in real time

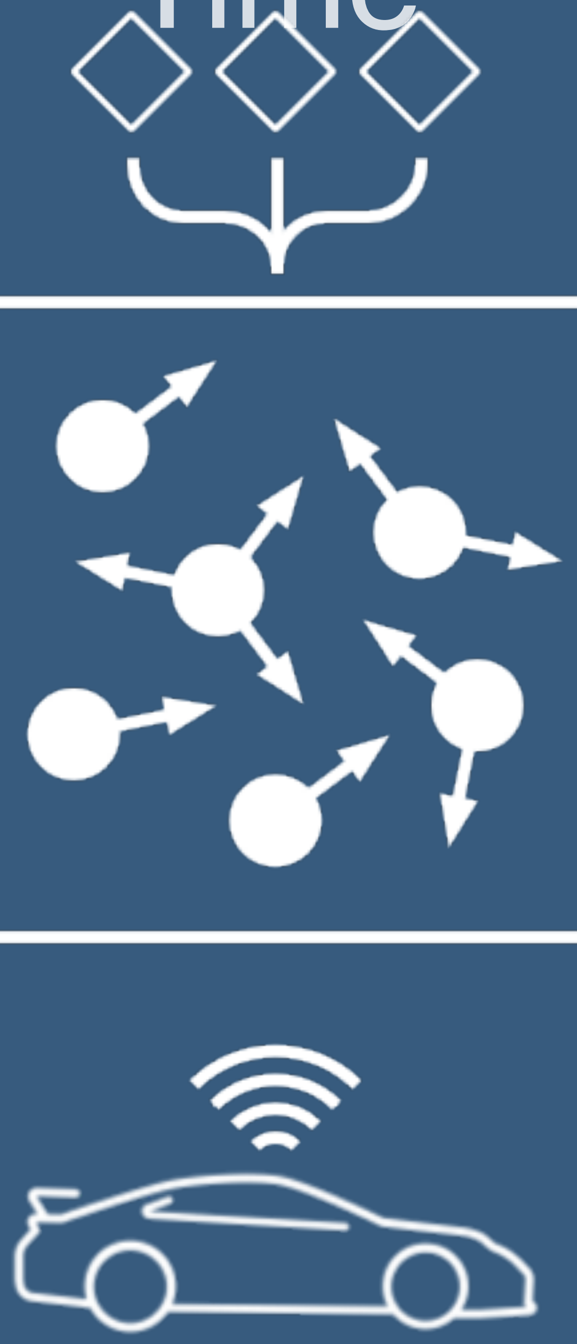


# Runtime control examples

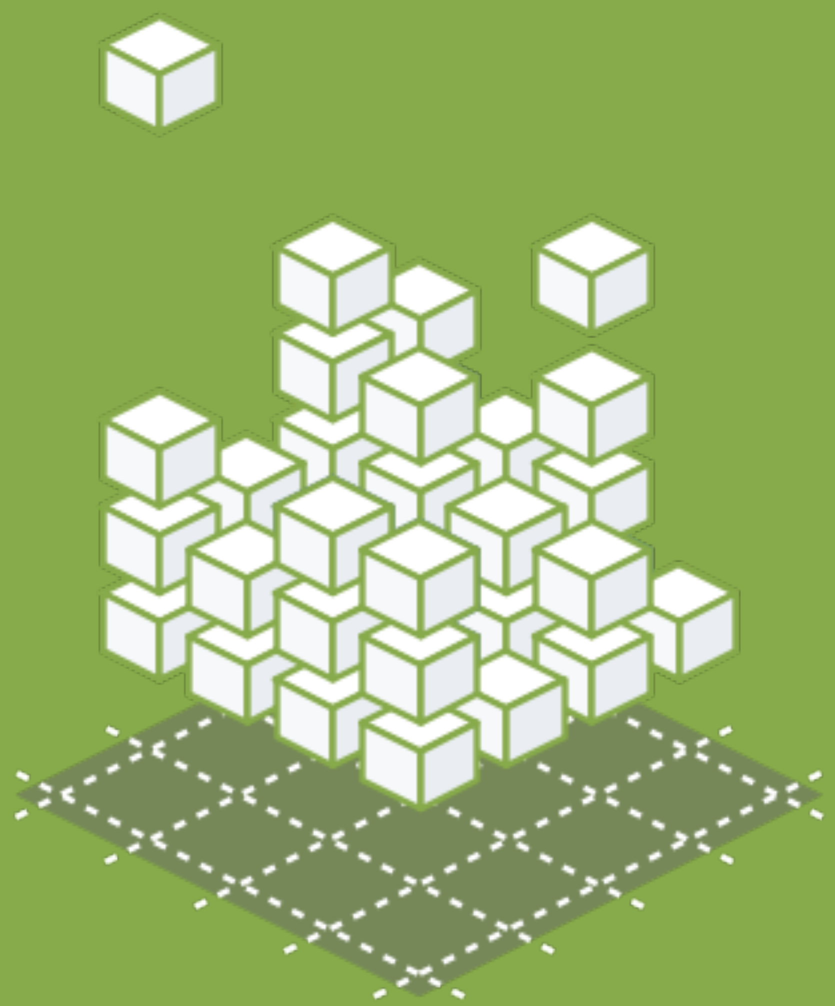
## Deploy to Production



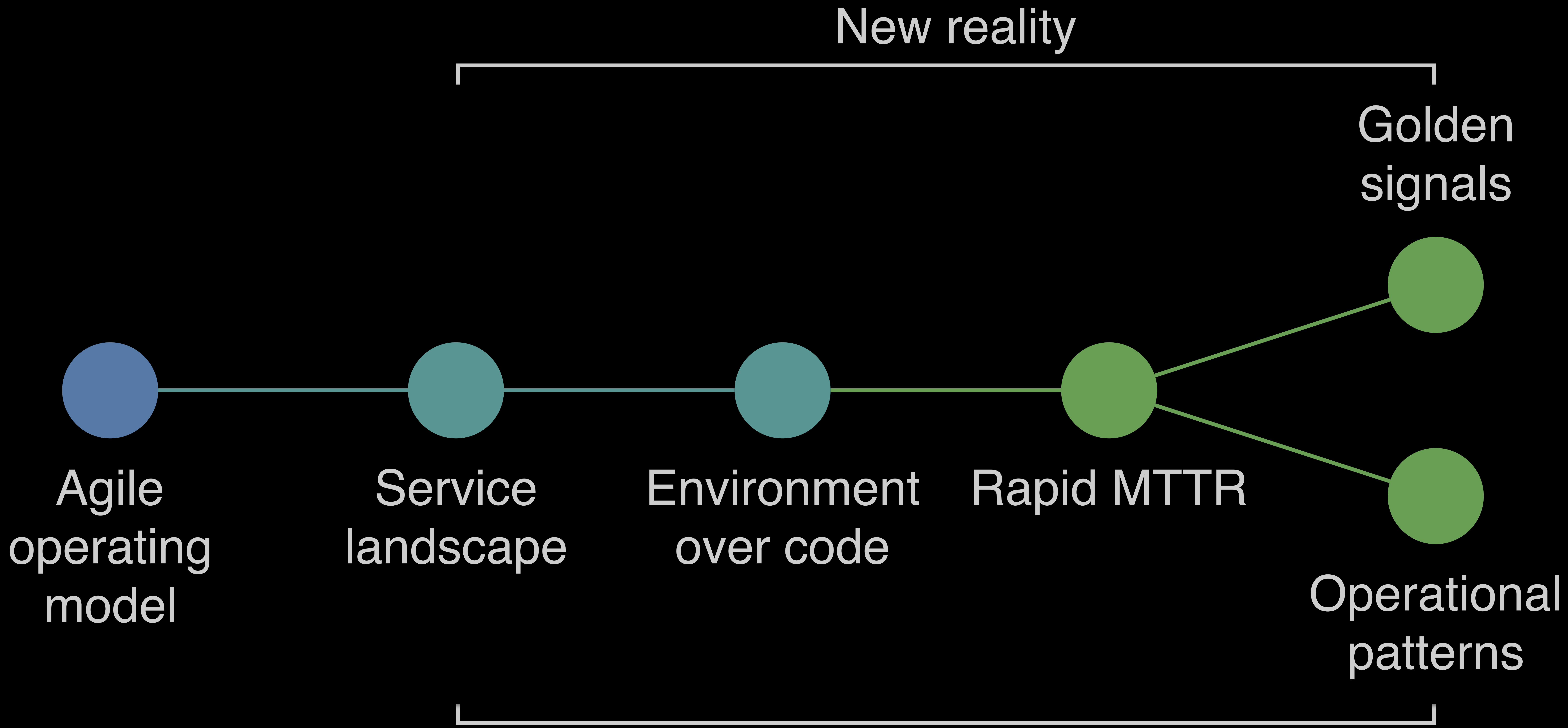
## Architect in Real Time



## Define Structure



# Summary



New reality
Golden signals
Agile operating model
Service landscape
Environment over code
Rapid MTTR
Operational patterns

# Takeaways

Developers

1. Avoid distributed systems
2. Resilient federations
3. Compensate
4. Redundancy, plausibility
5. Debug at unit level
6. Defer design to runtime

Operators

1. Environment over code
2. Rapid detect–react loops
3. Signals, patterns
4. No root causes: remediate
5. No process debugging
6. Architect at runtime

Applies to all decentralized architectures, not just microservices.

GLASNOSTIC

# Mission Control for Agile Architectures

glasnostic.com

Tobias Kunze
Co-Founder & CEO

tobias@glasnostic.com
@tkunze

Image: NASA