

# MITOSIS pitch

## MITOSIS: distributed autonomous management Of Service compositions

Giovanni Toffetti



# Foreword

- IaaS cloud research impact
  - Industry (e.g., J. Wilkes, A. Cockcroft)
  - Academia
- My strategic direction: focus on application providers rather than infrastructure providers
- Cloud-native applications: new development paradigms, best practices, open challenges

# Problem: cloud-native applications

- Cloud-native apps/services: much more than deploying VMs
- Three sources of uncertainty:
  - Varying demand/load
  - Unreliable infrastructure
  - Unreliable/varying 3<sup>rd</sup> party services
- Scale of systems and need of immediate reaction require service management automation

# Management functionalities

- Monitoring (e.g., ELK stack) → TF9
- Health-management (e.g., fleet, kubernetes)
- Auto-scaling → QoS model-based → TF2+8
- Dynamic service (re)composition (e.g., ribbon) → TF3
- Dynamic placement → Optimization → TF4
- Dynamic traffic routing (zuul, dyn dns)

# Current state of the art

- IaaS providers offer:
  - Generic monitoring (infra + RTs)
  - Generic auto-scaling (rule-based)
  - Drawbacks: vendor lock-in, generic one-size-fits-all, costly VAS \$\$\$
- 3<sup>rd</sup> party offerings (Rightscale, Scalr, NewRelics)
  - Component-specific monitoring collection
  - Drawbacks: non-compliant with data privacy, not application specific, costly \$\$\$ (svc + data transfer)

# Proj Goals

- Keep management functionalities within the application
  - Avoid vendor lock-in (change or use more than 1 provider)
  - Save \$\$\$
  - Make management functionalities resilient and scalable with the service (eat your own dog food)
- Release an OSS framework for self-managing cloud applications
  - Allow researchers to focus on their specific area of expertise
  - Provide common use cases deployable anywhere
  - Foster scientific collaboration / community work

# Main ideas

- Resilient distributed management based on distributed configuration
  - Consensus algorithm for leader election: leader is responsible of mgmt functionality
  - “Stateless” mgmt can be restarted upon failure of any component → use shared state
- Apply same idea hierarchically and use service orchestration concepts to manage compositions and life-cycle

# etcd

- Distributed key value store
- Designed for: shared configuration & service discovery
- Implements *Raft consensus algorithm*
- Handles machine failures, master election etc.
- Actions: read, write, listen
- Data structure
  - /folder
  - /folder/key
- REST-API
- easy to use client: *etcdctl*





# etcd - example

## read/write a value

- > etcdctl get /folder/key
- > etcdctl set /folder/key

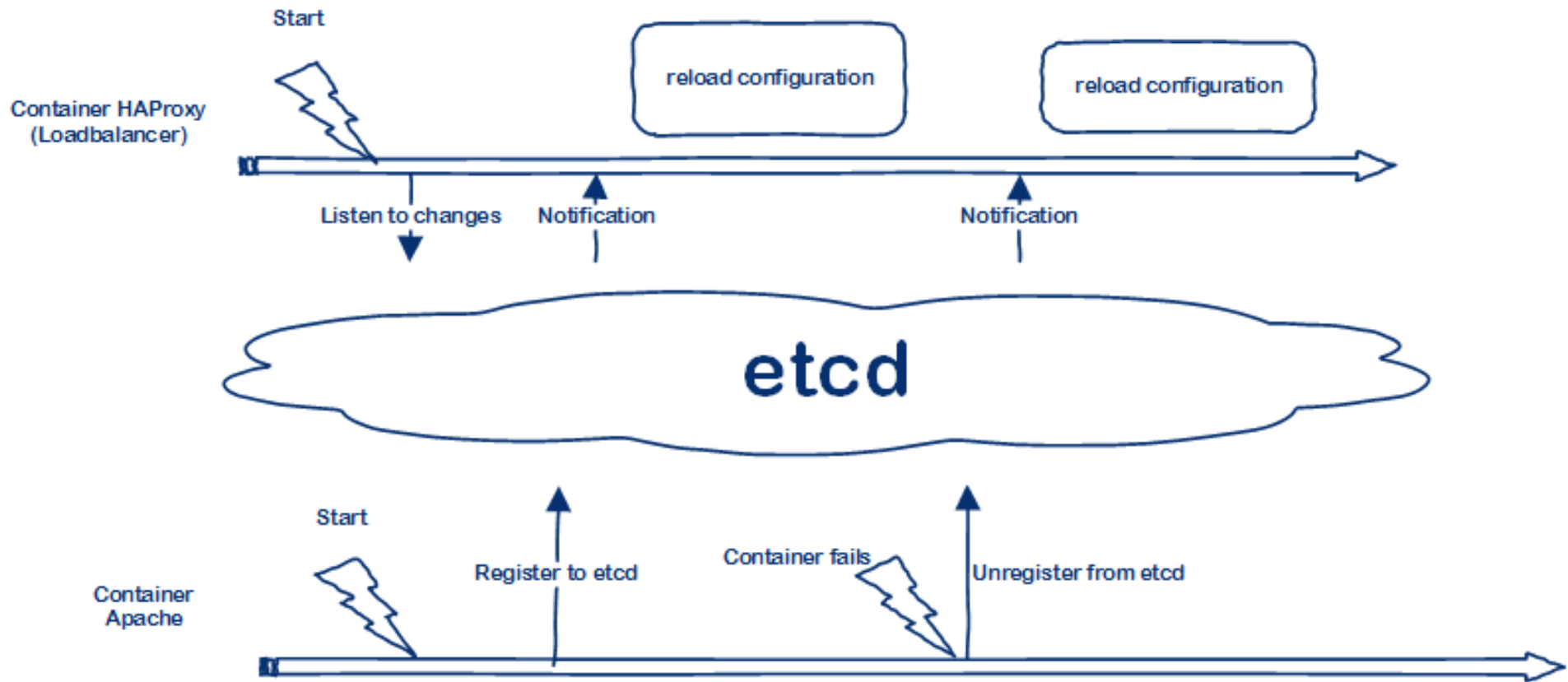
## read/create directory

- > etcdctl mkdir /folder
- > etcdctl ls /folder

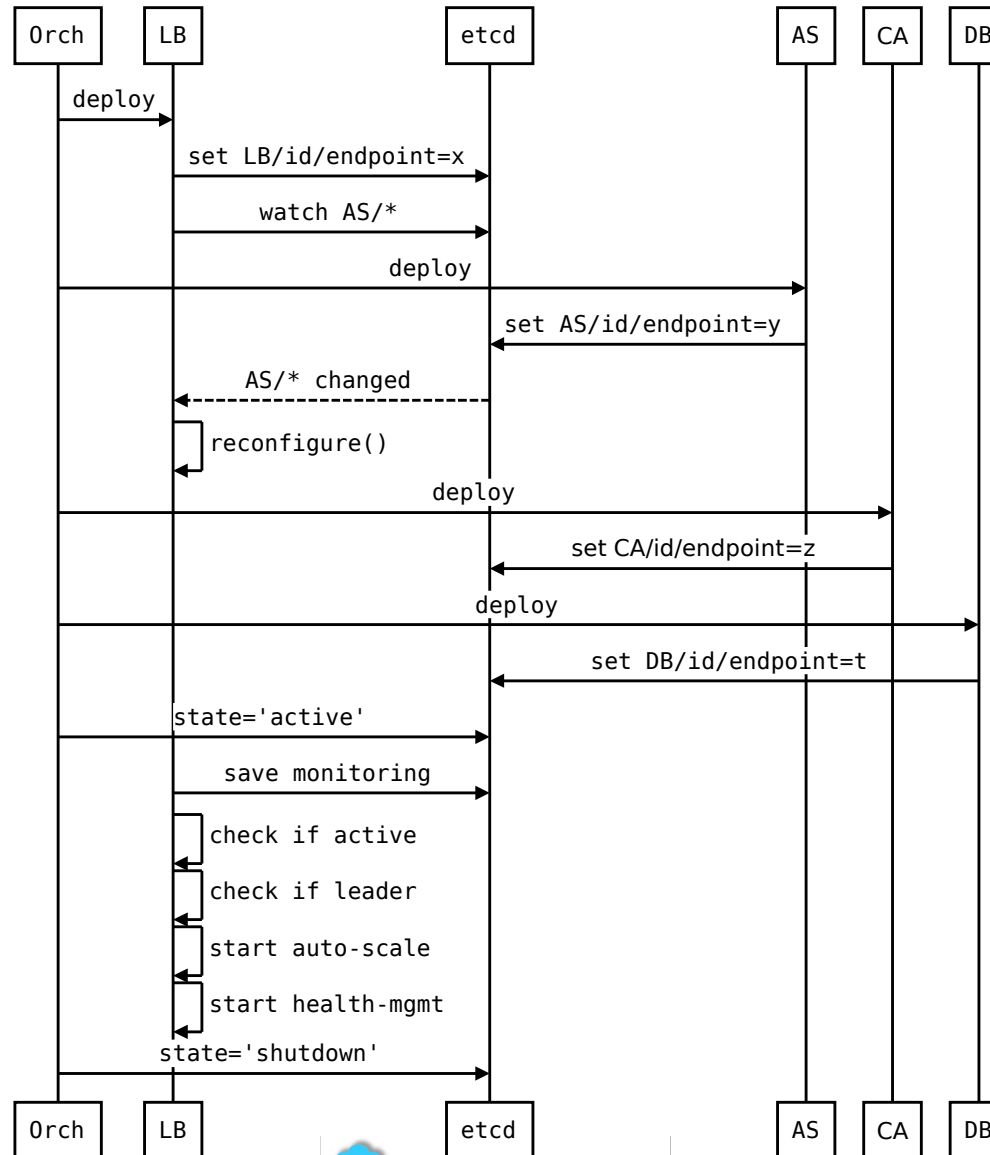
## listen to changes

- > etcdctl watch /folder/key
- > etcdctl exec-watch /folder/key -- /bin/bash -c "touch /tmp/test"

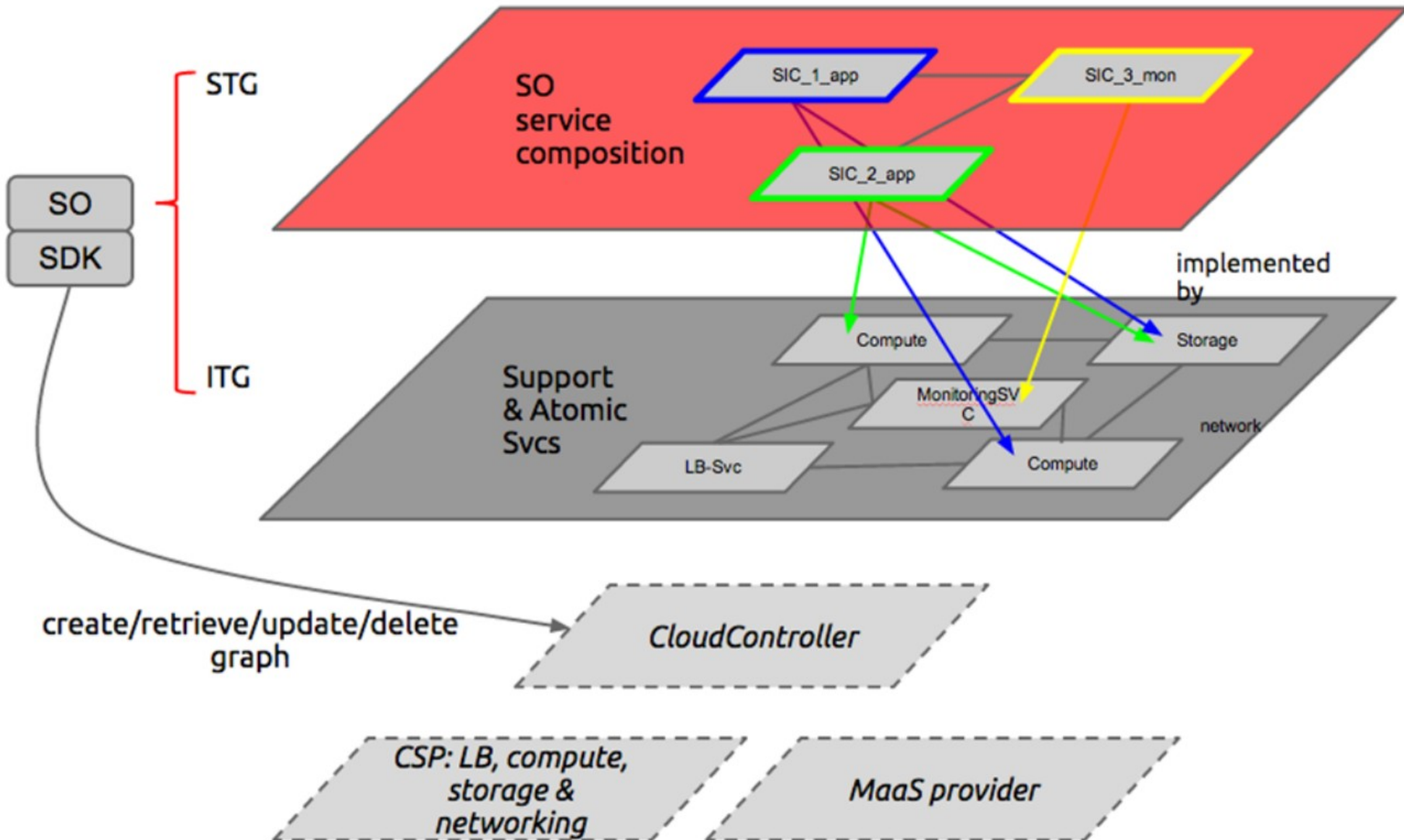
# etcd - service discovery



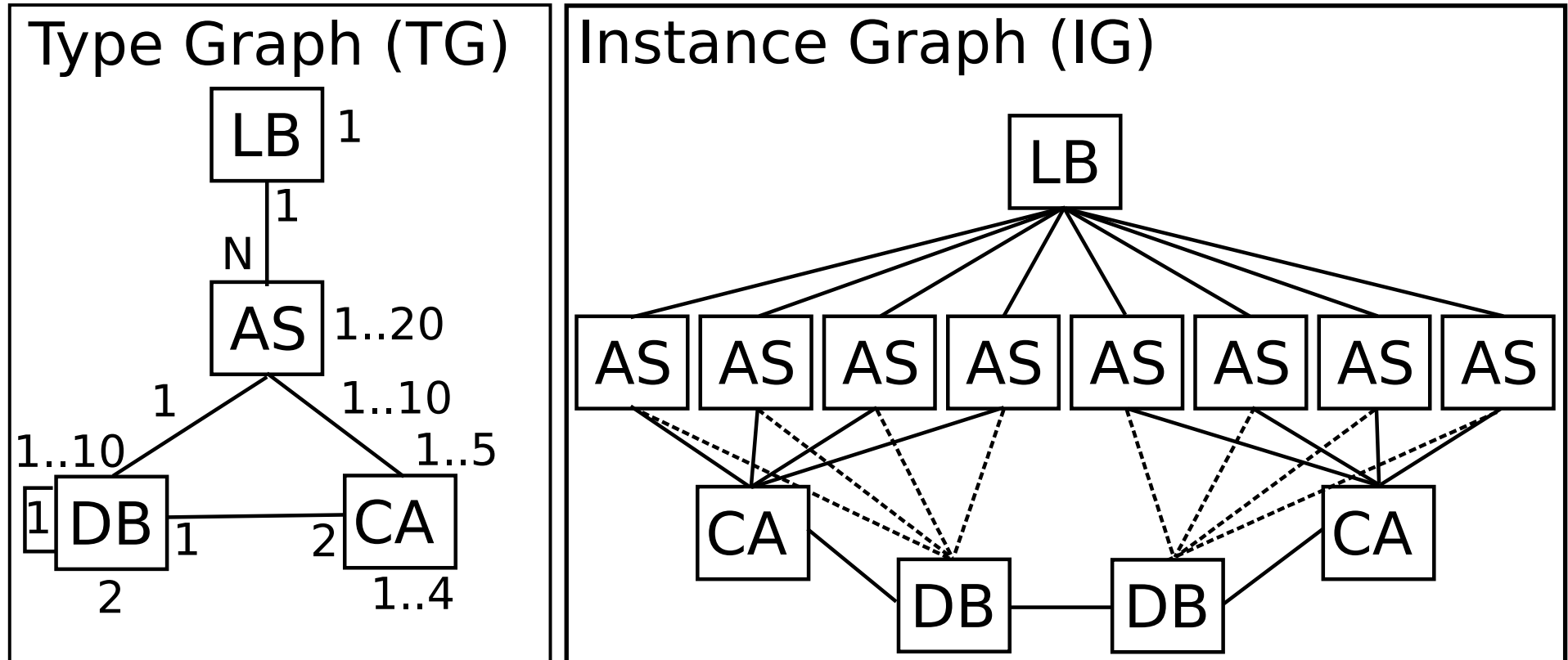
# lifecycle



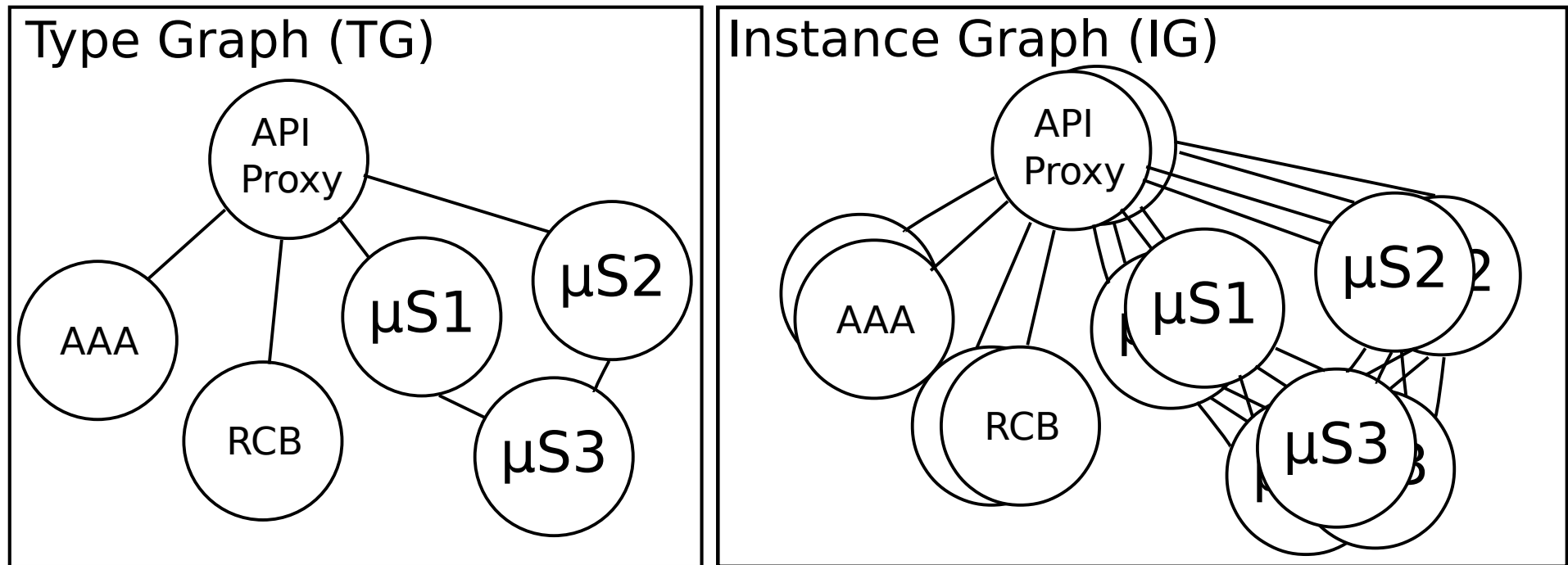
# Service Orchestration (the MCN way)



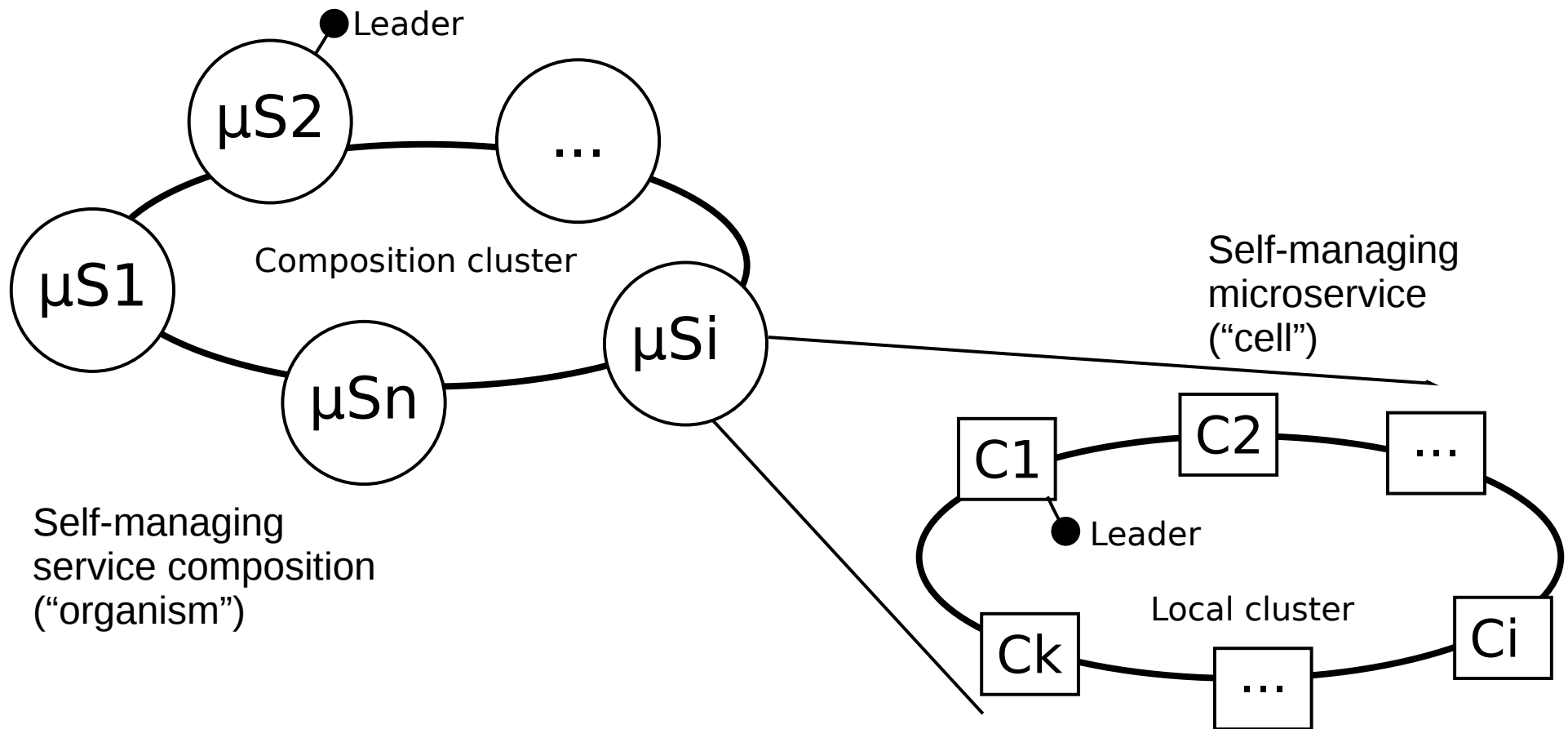
# Atomic (micro)service graphs



# Service composition graphs



# hierarchical etcd clusters



# Take home message

- Proj goal:
  - Vendor independent self-managing services
  - Managing functionalities deployed within the service (monitoring, health-mgmt, autoscaling, svc recomposition, placement, routing)
  - Strive for technology independent OSS fwk for self-managing svcs (etcd + REST + actuator wrappers)
- Looking for:
  - Experts in any of the mgmt functs willing to contribute their requirements / design inputs, approaches as fwk plugins, use cases
- Offer:
  - The base fwk, extensive cloud experience, good laughs :)



# That's all folks

## Any questions?

If interested just drop me a line at: [toff@zhaw.ch](mailto:toff@zhaw.ch)

- ICCLab: <http://blog.zhaw.ch/icclab>



- Cloud-Native Applications Initiative:  
<http://blog.zhaw.ch/icclab/category/research-approach/themes/cloud-native-applications/>

# Hic sunt leones

- Backup slides from here...

