

# ADDO<sup>®</sup>

ALL DAY DEVOPS

NOVEMBER 6, 2019



# DevOps in a containerized world

Martin Alfke - example42 GmbH



MARTIN ALFKE



DevOps in a  
Containerized World



example<sup>42</sup>

# Classical DevOps

Agile Development -> Faster Release Cycles

Collaboration and automation required

Everybody is a trusted and honest customer



Image: tatlin



# Classical DevOps

Product is customer for Dev

Dev is customer for Ops

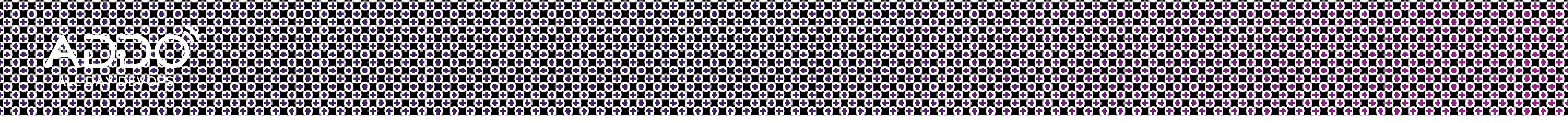
Everybody is customer for Sec

ProdDevSecOps or DevOps



Image: tatlin





# Classical DevOps

DevOps KATA

- (K)Culture



Image: tatlin



# Classical DevOps

DevOps KATA

- (K)Culture
- Automation



Image: tatlin



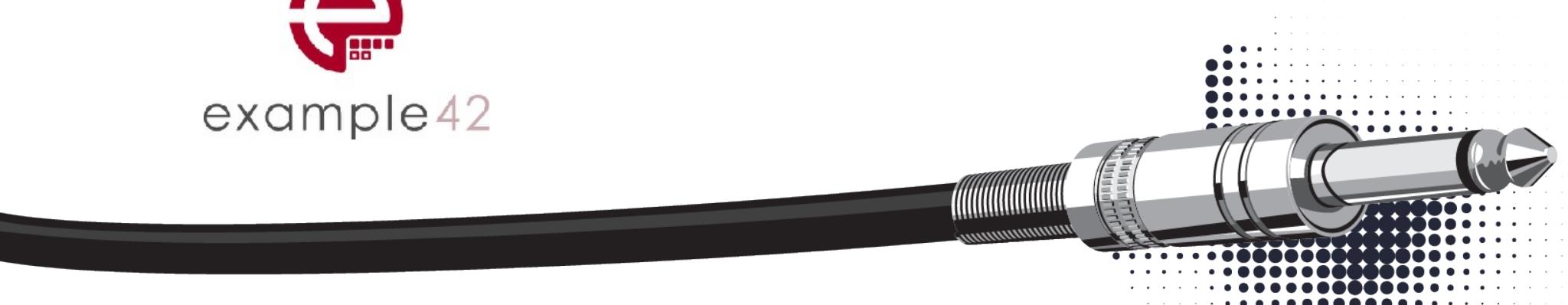
# Classical DevOps

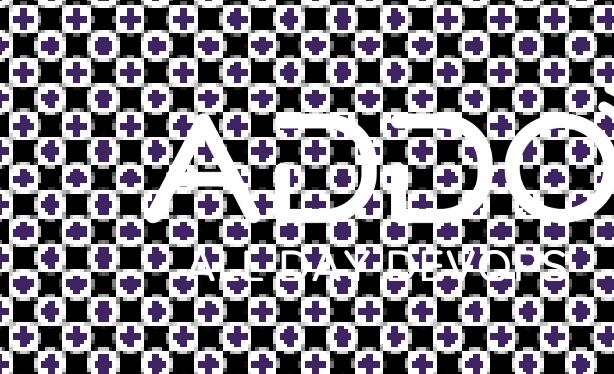
## DevOps KATA

- (K)Culture
- Automation
- Transparency



Image: tatlin





# Classical DevOps

## DevOps KATA

- (K)Culture
- Automation
- Transparency
- Agility



Image: tatlin



# Classical DevOps

Shared tooling (where useful and possible)

- Version Control
- Configuration Management
- Secrets Management



Image: tatlin



# Classical DevOps

Shared tooling (where useful and possible)

- VM Management
- Metrics
- CI/CD/CD System



Image: tatlin

# Classical DevOps

Shared responsibility

- Hardware
- OS
- Application



Image: tatlin

# Classical DevOps

Hard learning curve:

- GIT (rebase, squash, merge)
- API driven infrastructure
- Change fast and early
- Paradigm Change



Image: tatlin



# Cloud DevOps

Cloud adds Finance!

- Budget
- Spending overview and forecast
- Invoicing

Tooling remains the same

ProdDevSecFinOps



Image: tatlin



# Classical and Cloud DevOps

Standardized Systems Setup

On-premise or off-premise

Private or public cloud

Collaborative setup, management and maintenance



Image: tatlin



# DevOps and Containers

Dev learned complexity of systems and application

Separation of concerns:

- Ops manages OS and DC
- Dev manages App (incl. deployment, monitoring, metrics, alerting)



Image: tatlin



# DevOps and Containers

Container infrastructure:

- SDN is absolute must
- Containers need orchestration
- Monitoring on services, not systems



Image: tatlin



# DevOps and Containers

- New thinking on infrastructure and applications required (Dev, Sec, Net and Ops)
- Time to learn new concepts, technology and automation



Image: tatlin



# DevOps and Containers

## Why containers?

- Cloud compatible - more easy to migrate
- Dev can isolate issues within applications
- Ops can isolate issues within infrastructure



Image: tatlin



# DevOps and Containers

- Dev only needs CI/CD/CD, Registry and CR or CO API
- Dev responsible for staging and reverting via API calls/health checks
- Dev responsible for performance and availability (of applications)



Image: tatlin



# DevOps and Containers

- Ops responsible for sizing and storage
- Ops responsible for access using tokens to namespaces with hardware limits set
- Fin responsible for budget



Image: tatlin



# DevOps and Containers

- Sec provides policies on containers (cgroups, Kernel capabilities)
- Net builds interconnect between Layer 1 and SDN
- Sec supports all security aspects: Network, Servers, Application



Image: tatlin



# DevOps and Containers

How about:

- Tooling decisions
- Secrets
- Infrastructure decisions



Image: tatlin



# DevOps and Containers

DevSecOps:

- nothing is stand alone
- security brings everybody together
- KATA



Image: tatlin



# DevOps and Containers

How much Ops would you still like to do?

Maybe private cloud with "opsless" and "serverless" is an option (start-up mentality).

What about heritage platform?



Image: tatlin

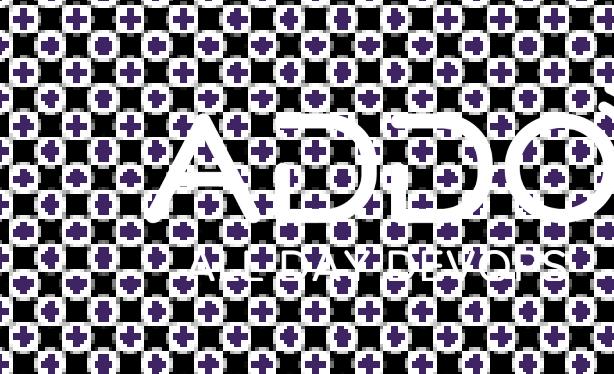


# Summary

“Simple can be harder than complex:  
You have to work hard to get your thinking clean to make it  
simple.  
But it's worth it in the end because once you get there, you can  
move mountains.”

Steve Jobs





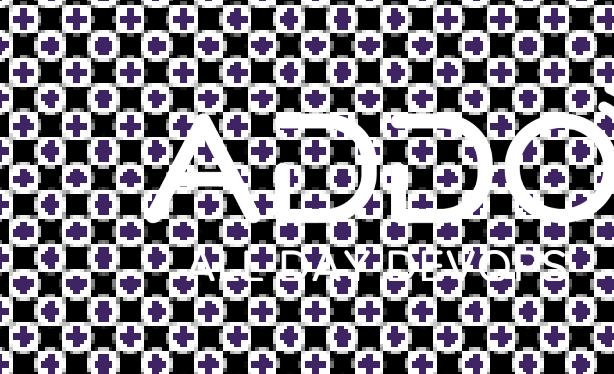
# Summary

- DevOps in a containerized world is not dead.
- It is even more required compared to heritage systems.
- Steep learning curve for everybody (Dev, Sec, Ops, Net, Fin, Mgmt).



Image: tatlin





# Conclusion

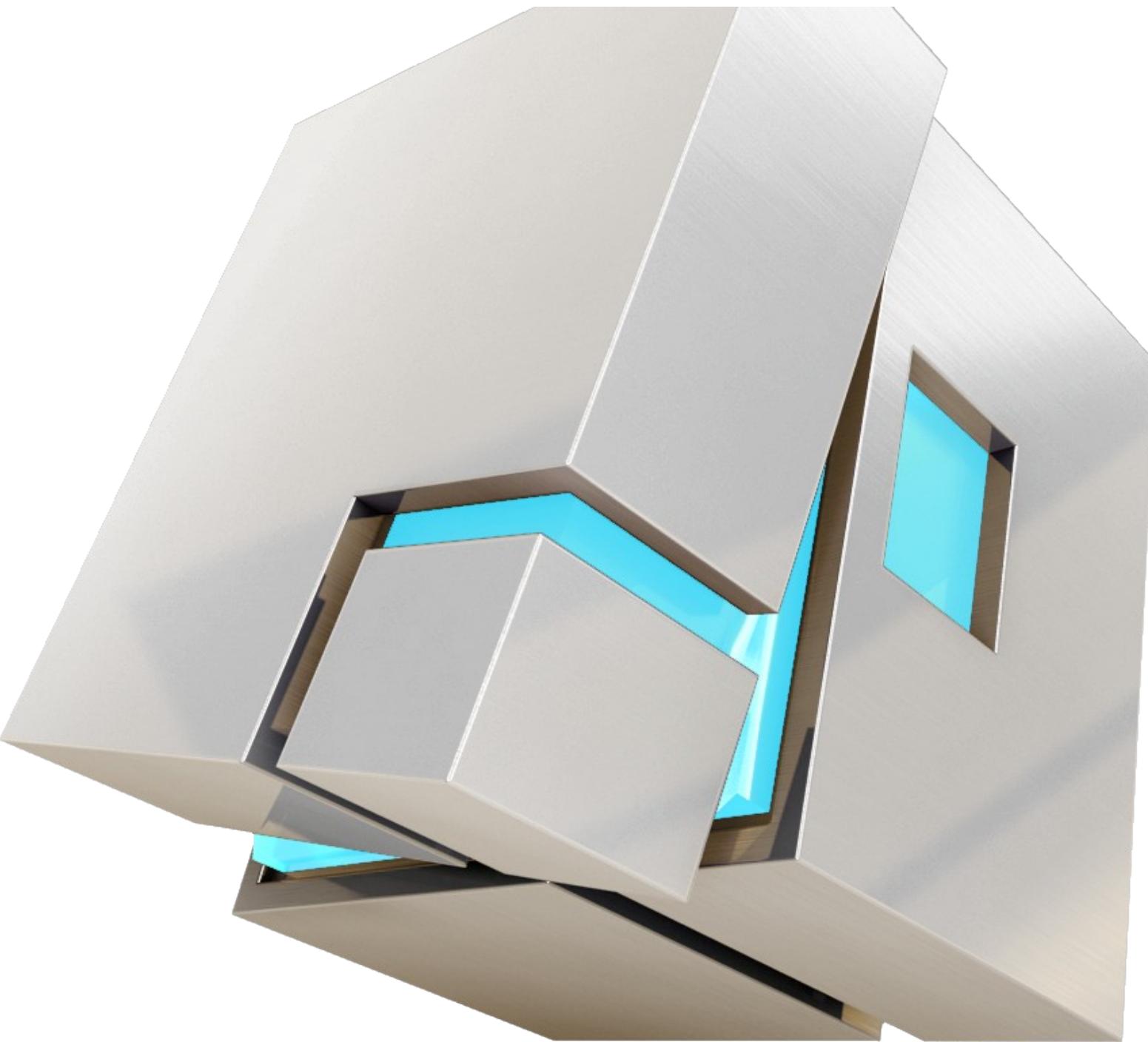
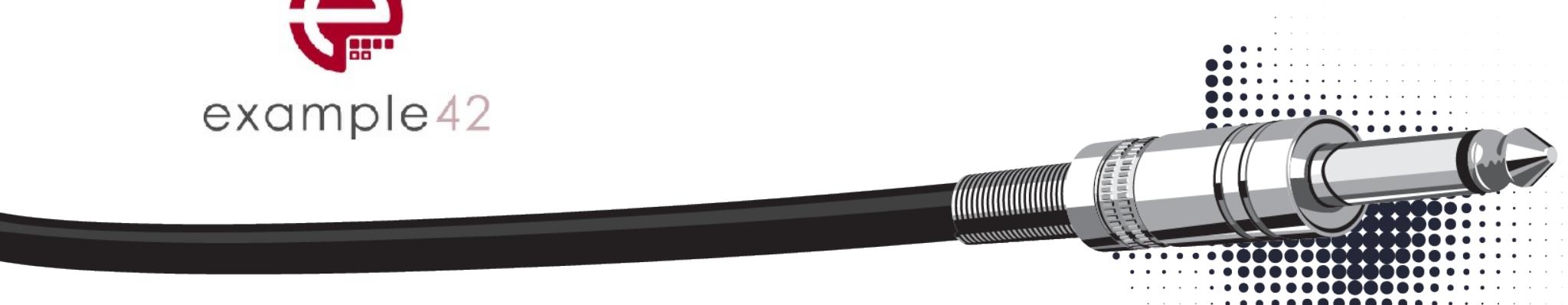
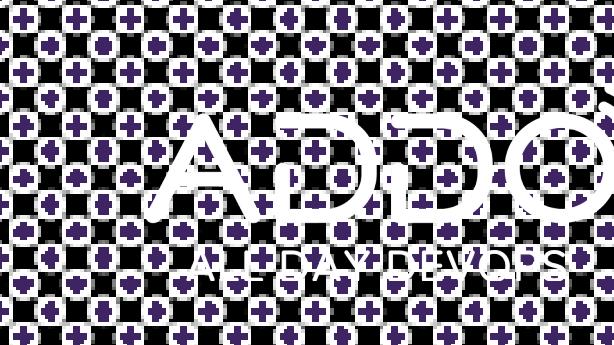


Image: tatlin





# Conclusion

- Find tools which integrate properly (REST API)
- Prevent NIHS (not invented here syndrome)

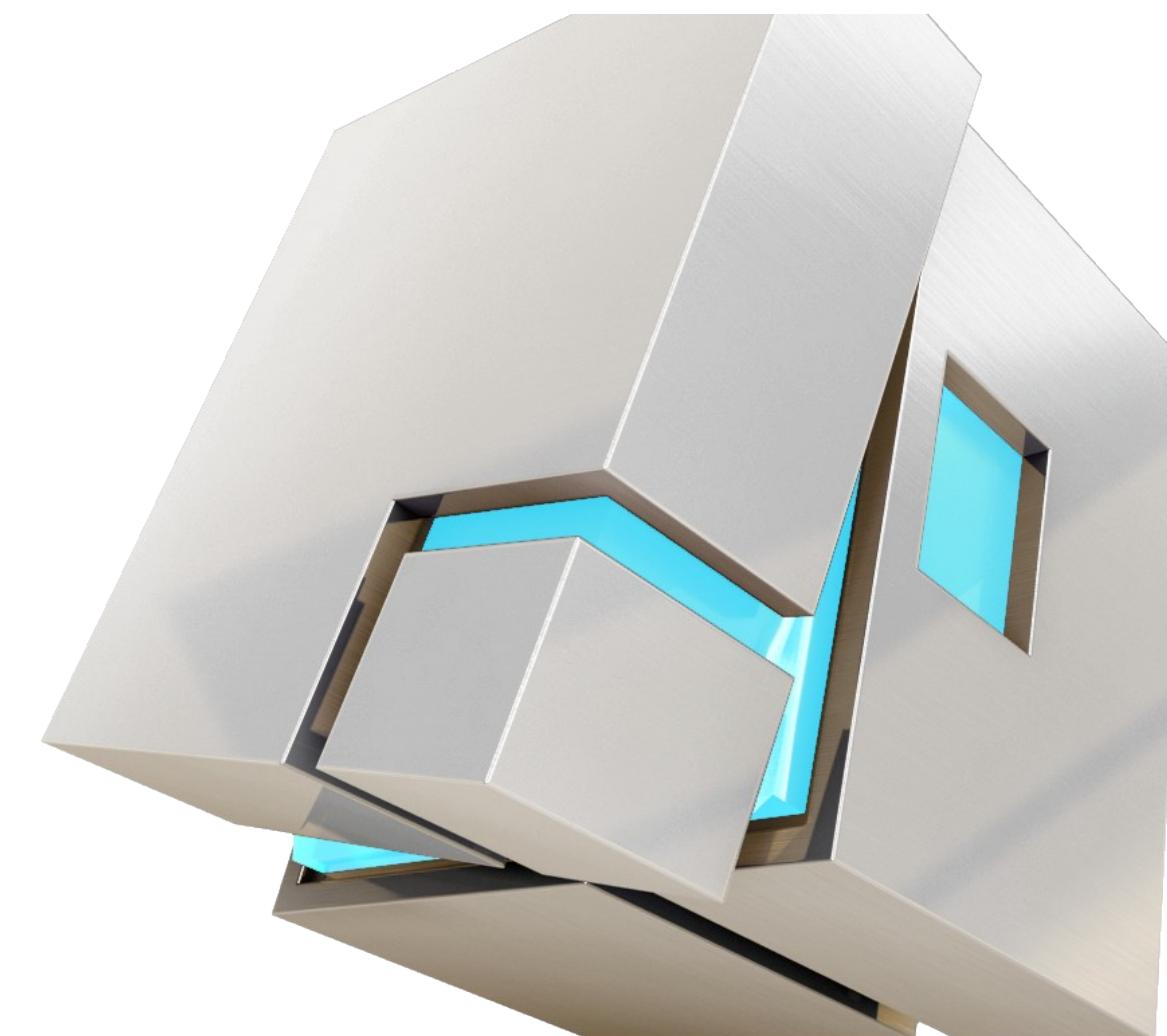
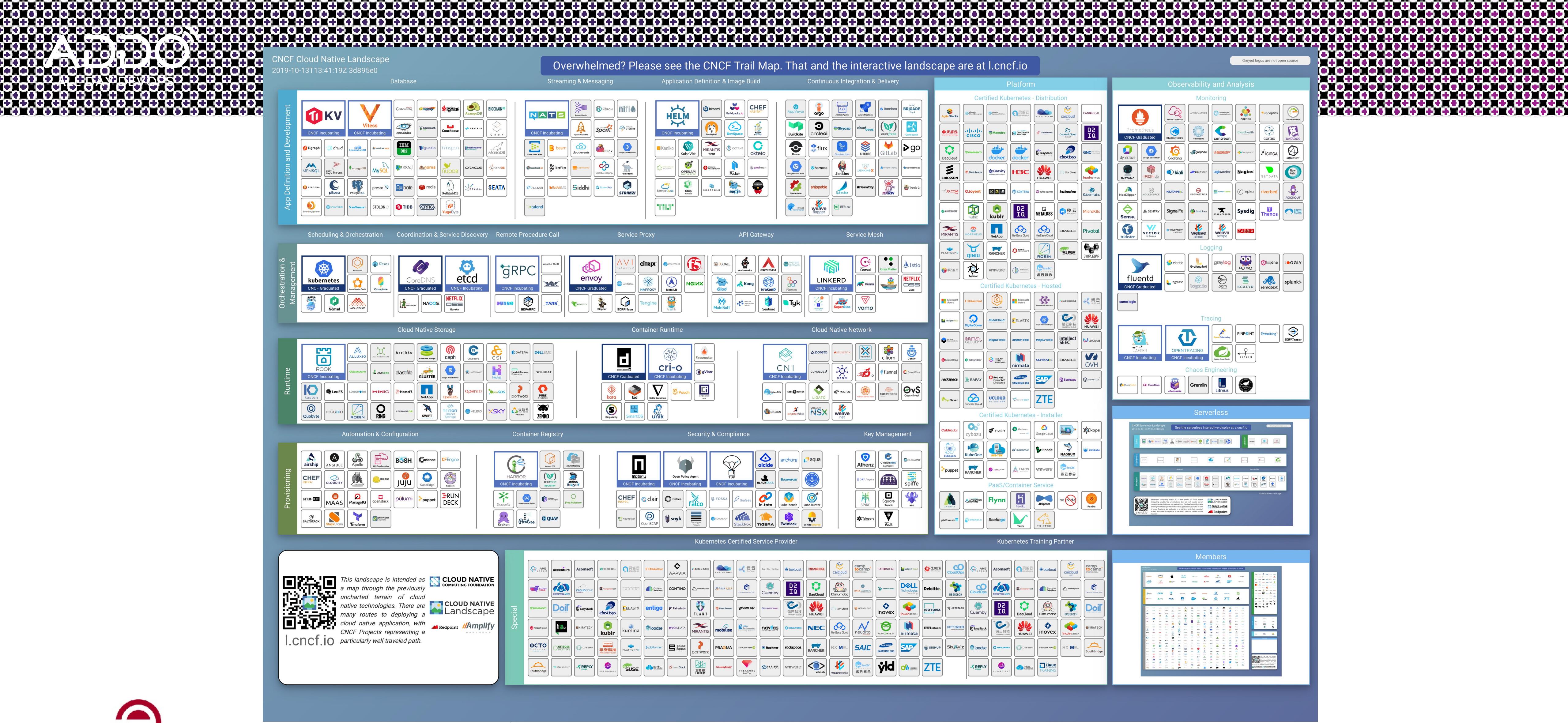


Image: tatlin

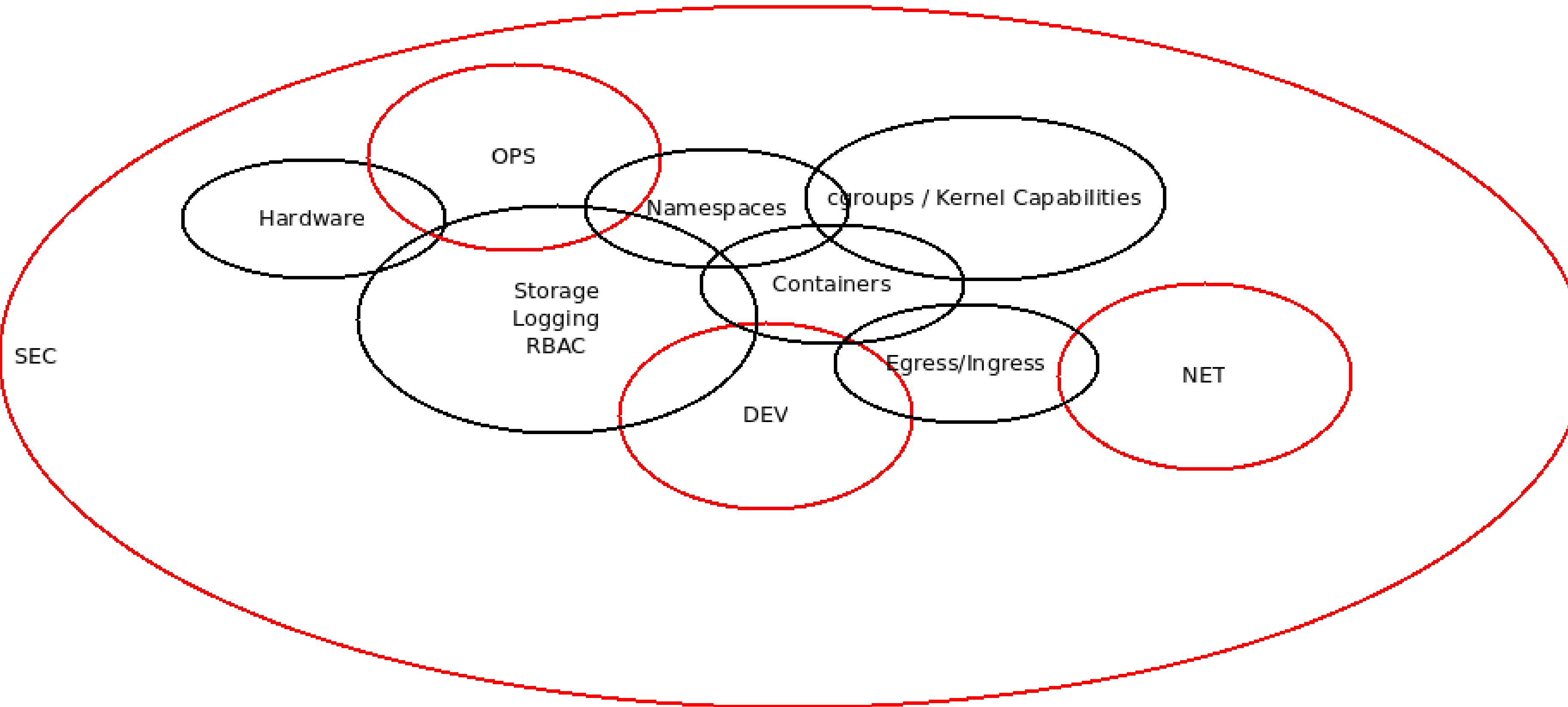


example<sup>42</sup>



## example 42

# Conclusion



example42



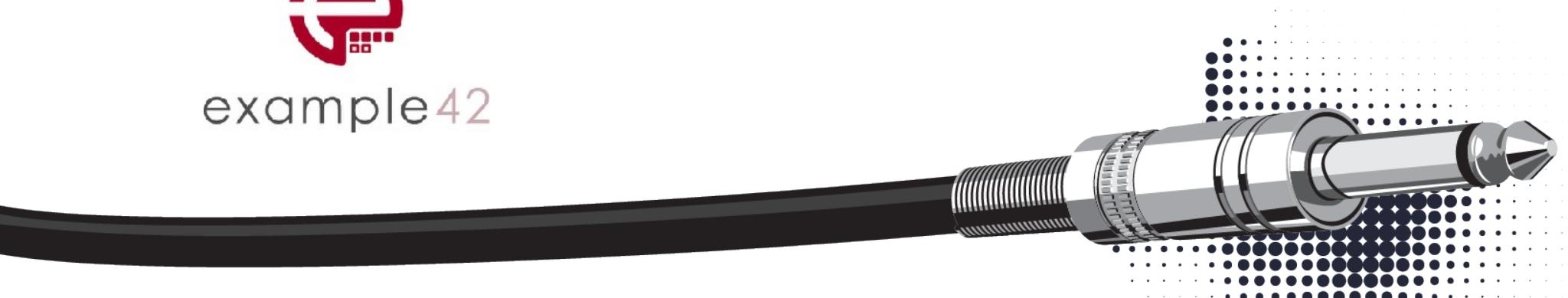
## Conclusion

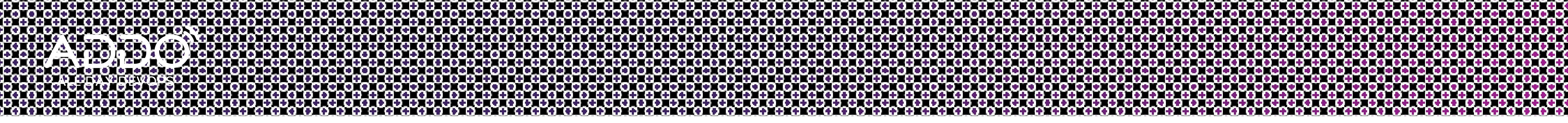
"I used to think that top environmental problems were biodiversity loss, ecosystem collapse and climate change. I thought that thirty years of good science could address these problems. I was wrong.

The top environmental problems are selfishness, greed and apathy, and to deal with these **we need a cultural and spiritual transformation**.

And we scientists don't know how to do that."

Gus Speth, March 2016





# DevOps in a containerized world

Martin Alfke - example42 GmbH



example42