DevOps in a containerized world

Martin Alfke - example42 GmbH

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

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

Image: tatlin

Cloud DevOps

Cloud adds Finance!

- Budget

- Spending overview and forecast

- Invoicing

Tooling remains the same

ProdDevSecFinOps

Image: tatlin

Classical and Cloud DevOps

Standarized 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

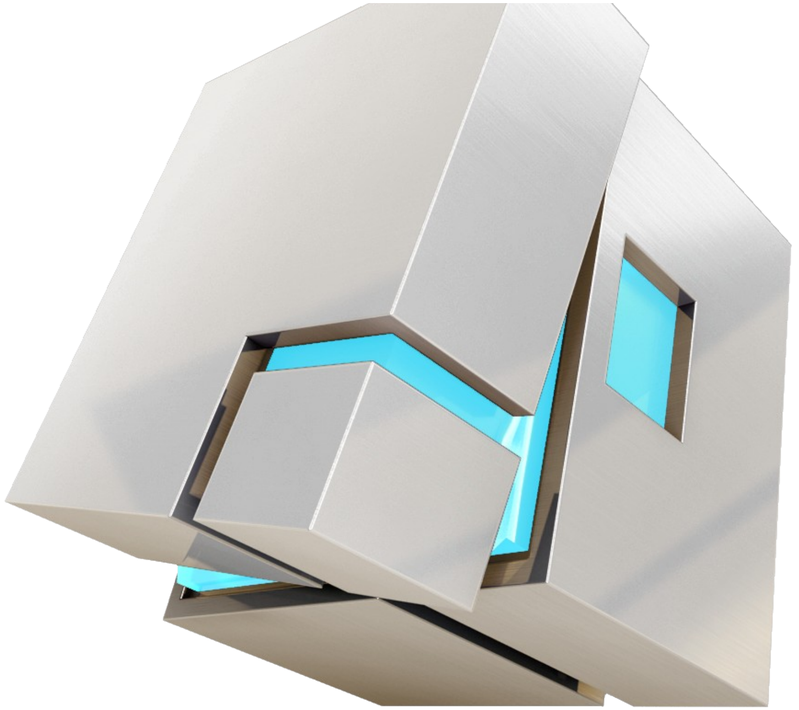


Image: tatlin

Conclusion

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

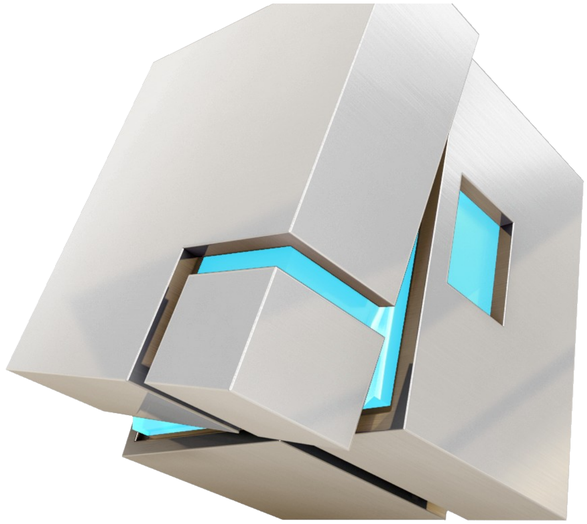
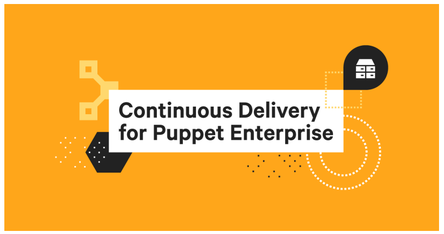


Image: tatlin

Conclusion



puppet Remediate™

Google Cloud

puppet Enterprise

bolt

project Nebula

aws

RUNNER

cri-o

vmware

CNCF Cloud Native Landscape

2019-10-13T13:41:19Z 3d895e0

Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

Greyed logos are not open source

App Definition and Development

Database

CNCF Incubating: TiKV, Vitess

Streaming & Messaging

CNCF Incubating: NATS

Application Definition & Image Build

CNCF Incubating: Helm

Continuous Integration & Delivery

Orchestration & Management

Scheduling & Orchestration

CNCF Graduated: Kubernetes

Coordination & Service Discovery

CNCF Graduated: CoreDNS

CNCF Incubating: etcd

Remote Procedure Call

CNCF Incubating: gRPC

Service Proxy

CNCF Graduated: envoy

API Gateway

Service Mesh

CNCF Incubating: Linkerd

Runtime

Cloud Native Storage

CNCF Incubating: Rook

Container Runtime

CNCF Graduated: containerd

CNCF Incubating: cri-o

Cloud Native Network

CNCF Incubating: CNI

Provisioning

Automation & Configuration

Container Registry

CNCF Incubating: Harbor

Security & Compliance

CNCF Incubating: Notary, Open Policy Agent, TUF

Key Management

Kubernetes Certified Service Provider

Special

Platform

Certified Kubernetes - Distribution

Certified Kubernetes - Hosted

Certified Kubernetes - Installer

PaaS/Container Service

Kubernetes Training Partner

Observability and Analysis

Monitoring

CNCF Graduated: Prometheus

Logging

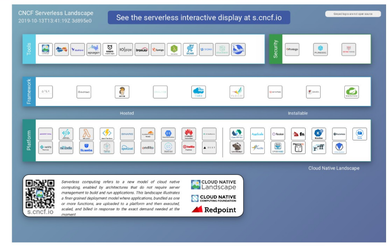
CNCF Graduated: fluentd

Tracing

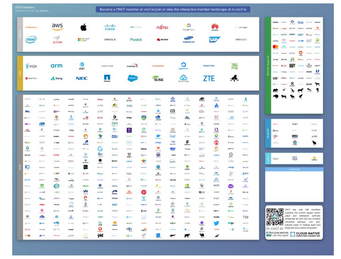
CNCF Incubating: Jaeger, OpenTracing

Chaos Engineering

Serverless



Members



This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

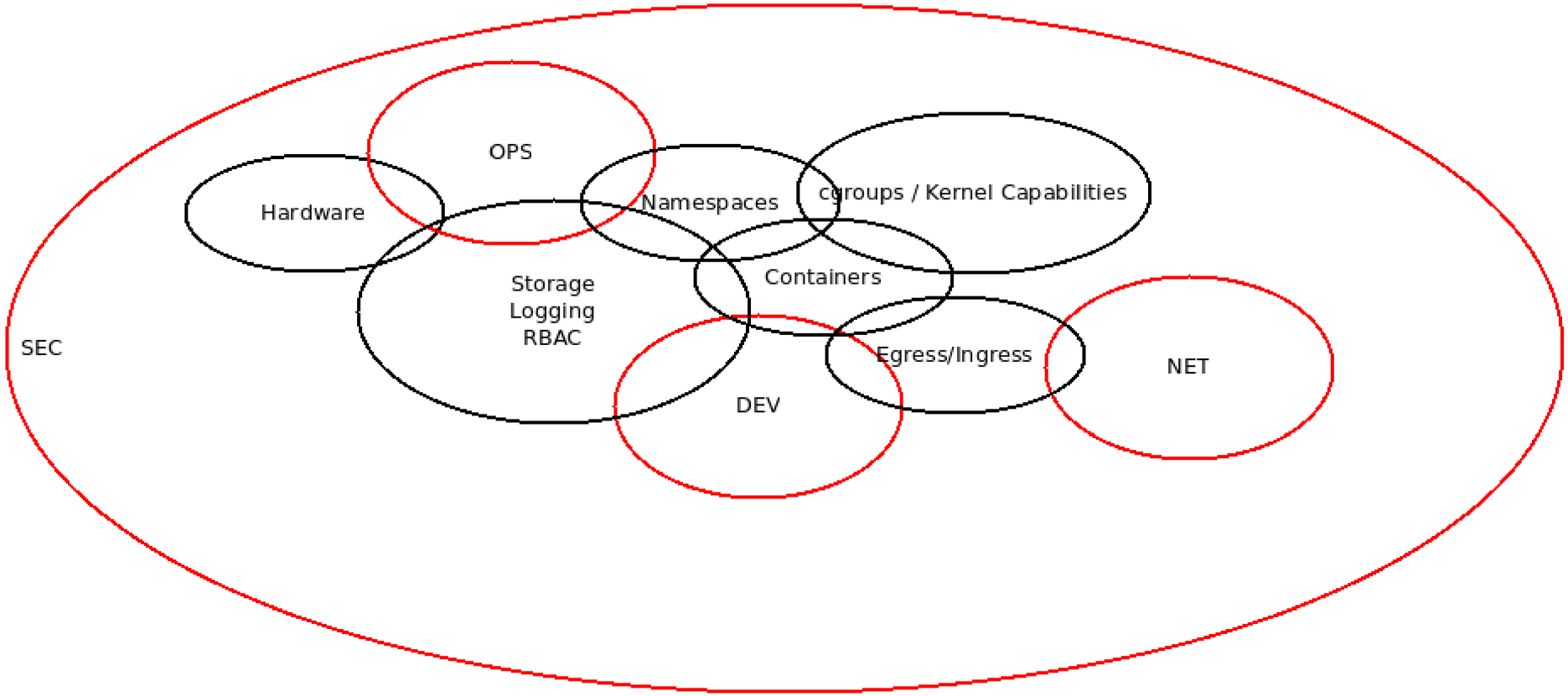
l.cncf.io

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE Landscape

Redpoint Amplify PARTNERS

Conclusion


SEC
OPS
Hardware
Namespaces
cgroups / Kernel Capabilities
Containers
Storage
Logging
RBAC
Egress/Ingress
NET
DEV

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