Do's and don'ts in building Cloud Native Platforms

Max Körbächer | Liquid Reply

REPLY
LIQUID

Introduction

Max Körbächer

Co-Founder & Manager Cloud Native Engineer, focusing on:

- Platform Engineering
- Application Delivery
- Cloud Native Advisory

Part of the Kubernetes Release Team & Release Engineering Team

Running a Cloud Native Blog/Newsletter nativecloud.dev

Common patterns & problems with Platforms

Platforms abstracts infrastructure complexities away.

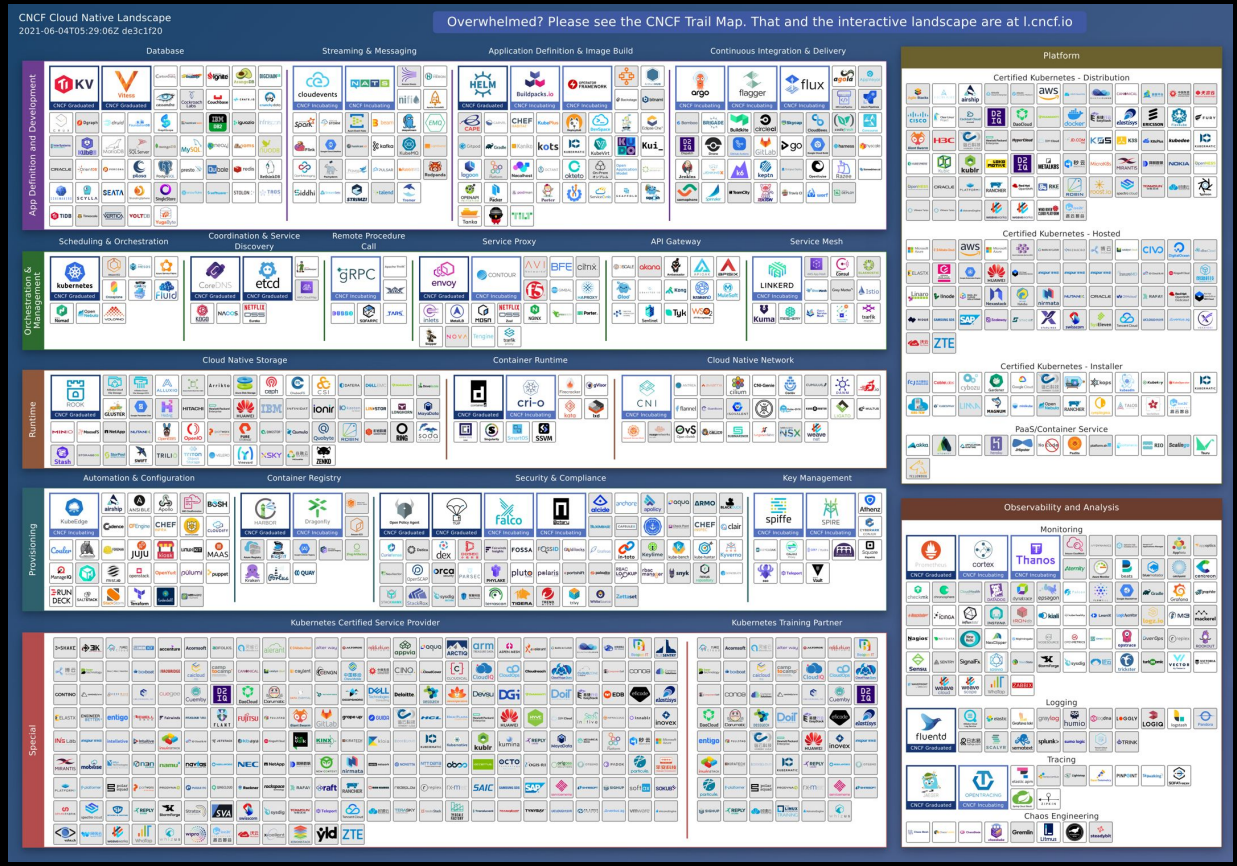
BUT they create new unknown, custom complexity:

- New responsibilities
- 100 options for one problem
- Single vs Multi Clusters
- CICD, GitOps or better something else
- How to build the application?
- How to ensure security, compliance and governance?

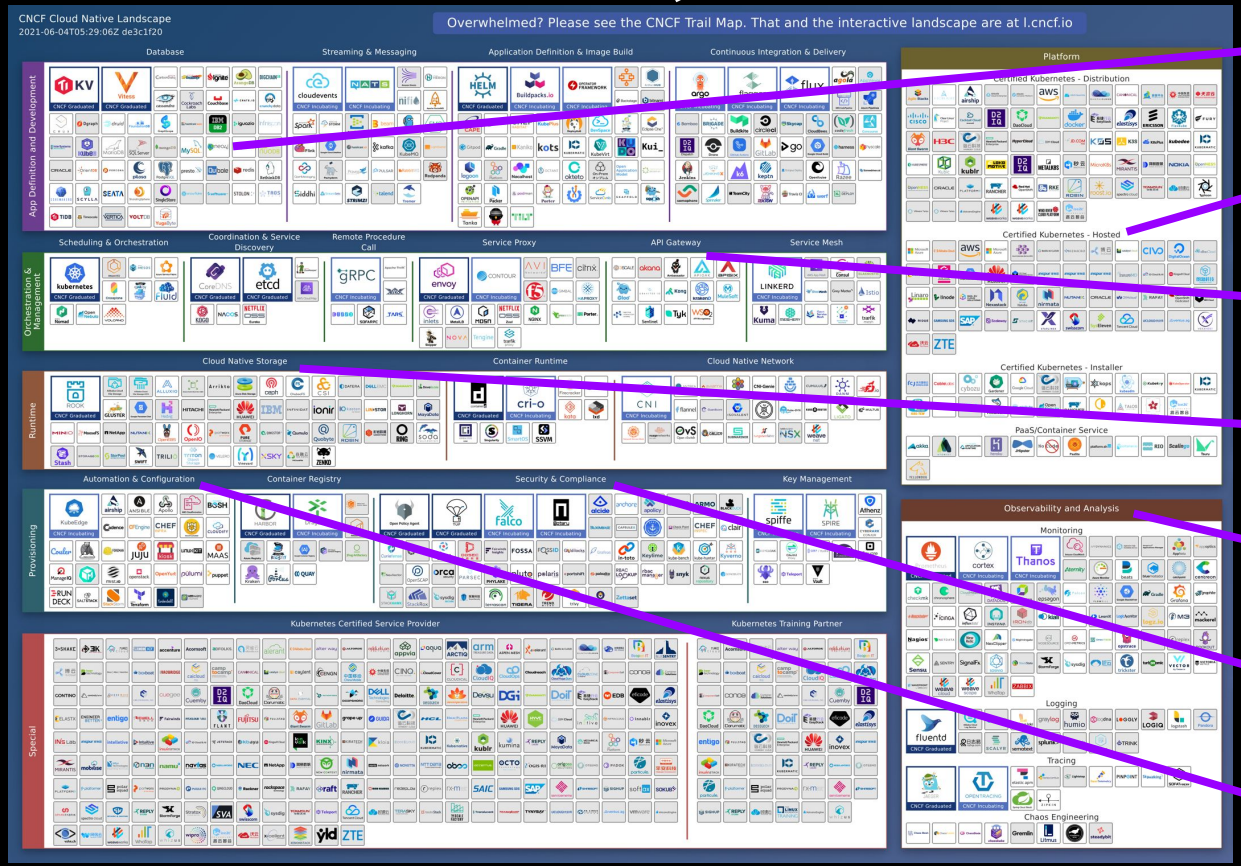
1 - Don’t get lost in open source

Open source is awesome!

But there is so many options



Pick what is useful, not what can do most



- Databases
- Hosted K8s
- API Gateways
- Storage
- Observability
- Security
- Automation

Open Source Office

Clarify early your organizational rules:

- Which licences can be used?
- With whom and where to document the used tools and licenses?
- Are there freemium conditions?

Photo by Mark Duffel on Unsplash

2 - Keep your Network simple

Networking is by far the most over-engineered part

- you don’t need to go through five firewalls if they do the same thing
- overcomplicated networks prevent the usage of cloud resources from an application development perspective
- avoid hundreds of accounts, multi-cloud and to many hub & spoke architectures

Utilize simple setups and extend them through in cluster tools like cert-manager, external dns, native integrations to LBs & Gateways

3 - Don’t focus to much on the provisioning of K8s

Don’t do it yourself

Enterprises and companies are obsessed with do it yourself. (Maybe a PTBS of decades of utilizing commercial tools)

But, building tools or internal products purely on IaC, Ansible and co is a never ending story.

1. Use IaC for the basic ressource definition
2. a cluster deployer for the K8s and its configuration.

Photo by Kevin Kandlbinder on Unsplash

You don’t need to reinvent the way of installing K8s

On Cloud

Use Managed Services

In most cases a managed cluster suits all the needs - (if the right provider used)

On Prem

Build on what you know

Beside a pure bare metal provisioning a VM foundation is recommended for simplicity and speed.

In Case

Go cloud native

Sometimes you will go wild and crazy - open source tools utilizing the APIs on any provider like Crossplane & co

It’s the future!

K8s provisioning is not magic

With the right tutorial anyone can do

The real challenges just start after K8s is up and running

- Configure users and access rights right
- How really to use K8s?
- Getting CICD or GitOps right
- Turn on security means normally to turn down all apps
- Traffic steering and management

4 - Platforms are for Products/Applications

Platforms are not build for themself

A platform is a target for the dev/product teams

Support various entry points or even manage the application delivery path!

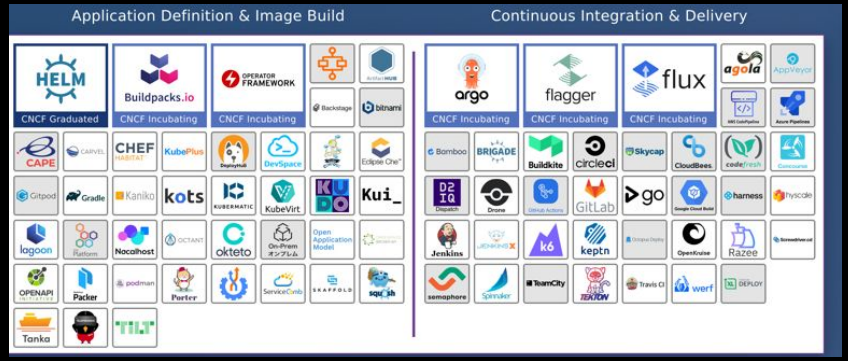
- apply best practices on container
- … deployment configs
- … security
- … integrity
- integration between cluster wide solutions and the apps

Let’s focus on what is important

Application Definition & Delivery

Right now we ride a wave of complexity. The target systems getting highly complex, a simple executable is not enough to run and responsibilities getting newly sorted.

“Application Definition & Delivery”, ADD or short Application Delivery is a part of the platform engineering which is developer focused and try to support as good as possible their mission:



NOT to learn 3x Cloud Provider, K8s, Helm, min. 5 possible sidecar injections and fixing your CICD every 2 days


KubeVela
TELEPRESENCE
SR-71
keptn
TiLT

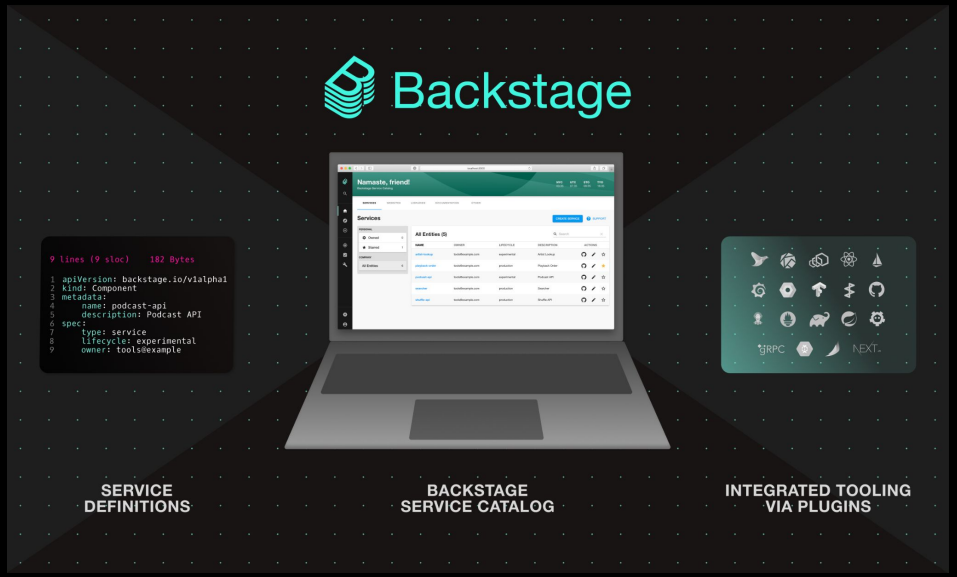
5 - Platform Engineers need to provide support and insights towards development

Disciplines are mixing up

Yet there is a difference between Dev, Ops, Platform, DevOps, etc

The vast majority of software projects don’t utilize the Cloud Native tools and services, but stuck with building software as done the past years, just in smaller junk sizes.

With platforms like Backstage you can work with all teams together on implementing and deploying blueprints for your apps.



6 - Multi-Cluster, many tenancies, the platform team stumble

a haiku by Max

The highest art of designing platforms

And the most diverse questions to answer

How many cluster, one per app, per domain, or better shared for all?

How to isolate users and workload, how secure it this, how needed is this?

Multicluster vs Single

There is no right and wrong

Whether you do one or many clusters, important is you are good with:

- automation of provisioning
- less scripts more declarative
- ease of use
- build on usefulness not on fanciness

Photo by Alora Griffiths on Unsplash

Workload Isolation

Containers are not secure, isolated bunkers

There are three levels of isolation:

- Linux Containers - simple and light but insecure
- Sandboxed Containers - not for every use case but simple, better security
- VM based Containers - resource hungry but great isolation your Security ❤️ lives here

Multi-Tenancy

Critical is the integration between cluster and further services

- Isolation is nothing without a proper IAM
- Most difficult: align the on cluster rights and roles with the e.g. cloud resources - requires often a poor 3rd party integration
- On K8s the OSS project Kiosk is very helpful

Check out the Kubernetes Multi-Tenancy WG benchmark to see where you are standing:
https://github.com/kubernetes-sigs/multi-tenancy/tree/master/benchmarks

Summary

Building platforms can be hard

But you don’t need to reinvent the wheel

Keep things simple

Design and build for change!

Think about the end user who is maybe not a 5y K8s veteran ;)

Photo by Stillness InMotion on Unsplash

Thank you!

REPLY
LIQUID