KubeCon CloudNativeCon Europe 2022

From Cloud Naive to Cloud Native - Avoiding mistakes everyone does

Max Körbächer, Liquid Reply

Say hi!

Max Körbächer - Co-Founder of Liquid Reply

My work is all about **Kubernetes Consultancy & Advisory**:

- What configuration/extensions I need?
- How I can make my teams using K8s?
- How do we get our software from local to global?
- What is the best approach to build a managed Kubernetes cluster like in internal developer platform?
- How do we do Security or Operations on Kubernetes?

mkoerbaecher

mkoerbi

Today we are talking about something that is difficult to catch and even more harder to describe

Moving to the Clouds

and becoming cloud native

Taking your steps into the clouds was never so easy as of today, as person or corporation.

You only need a credit card.

But it is on you, if it will be a one way flight ticket through the clouds or a new era of cloud native development.

Old but gold

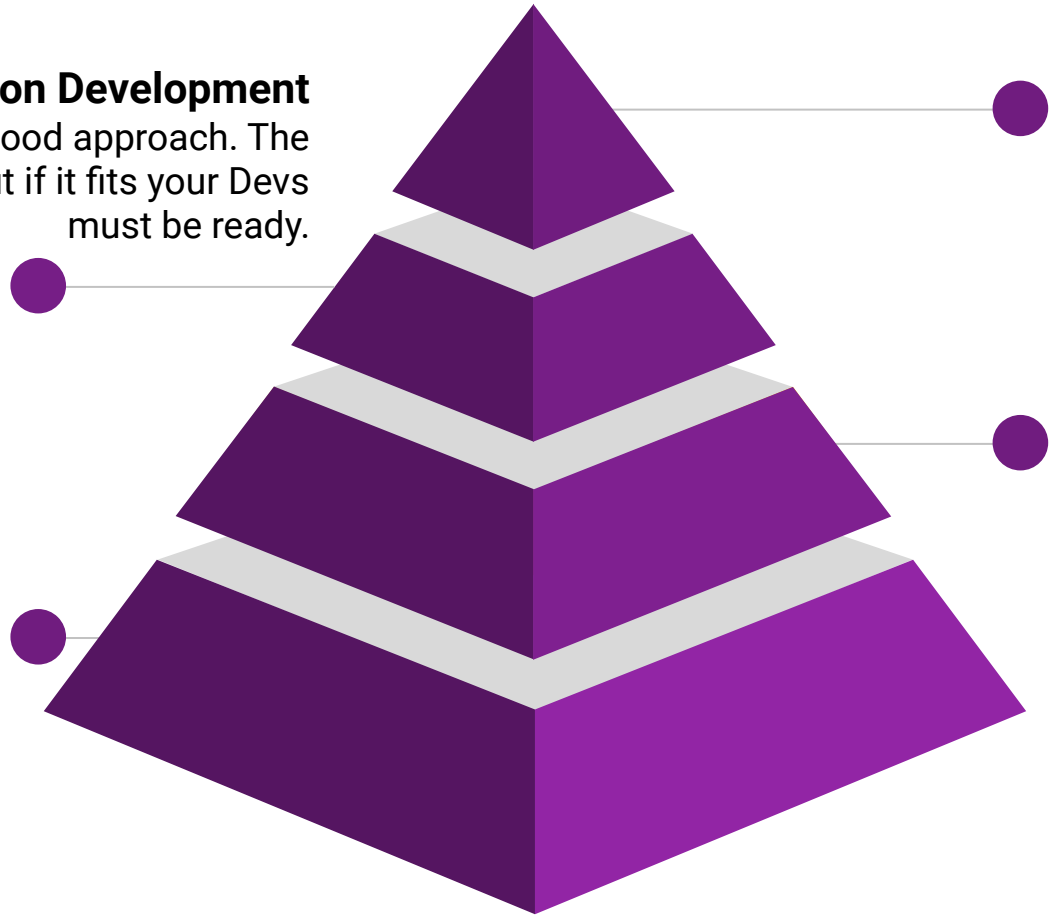
Solved all your problems. You're welcome.

The Cloud Native Pyramid

KubeCon CloudNativeCon Europe 2022

Business Requirement <-> Application Development
Cloud Native is often but not always a good approach. The same with containers and Kubernetes. But if it fits your Devs must be ready.

Cloud Native Mindset/Methodology
Cloud Native isn't a tool, a process or an object you can buy. We are talking about a mindset and a kind of unformalized methodology.

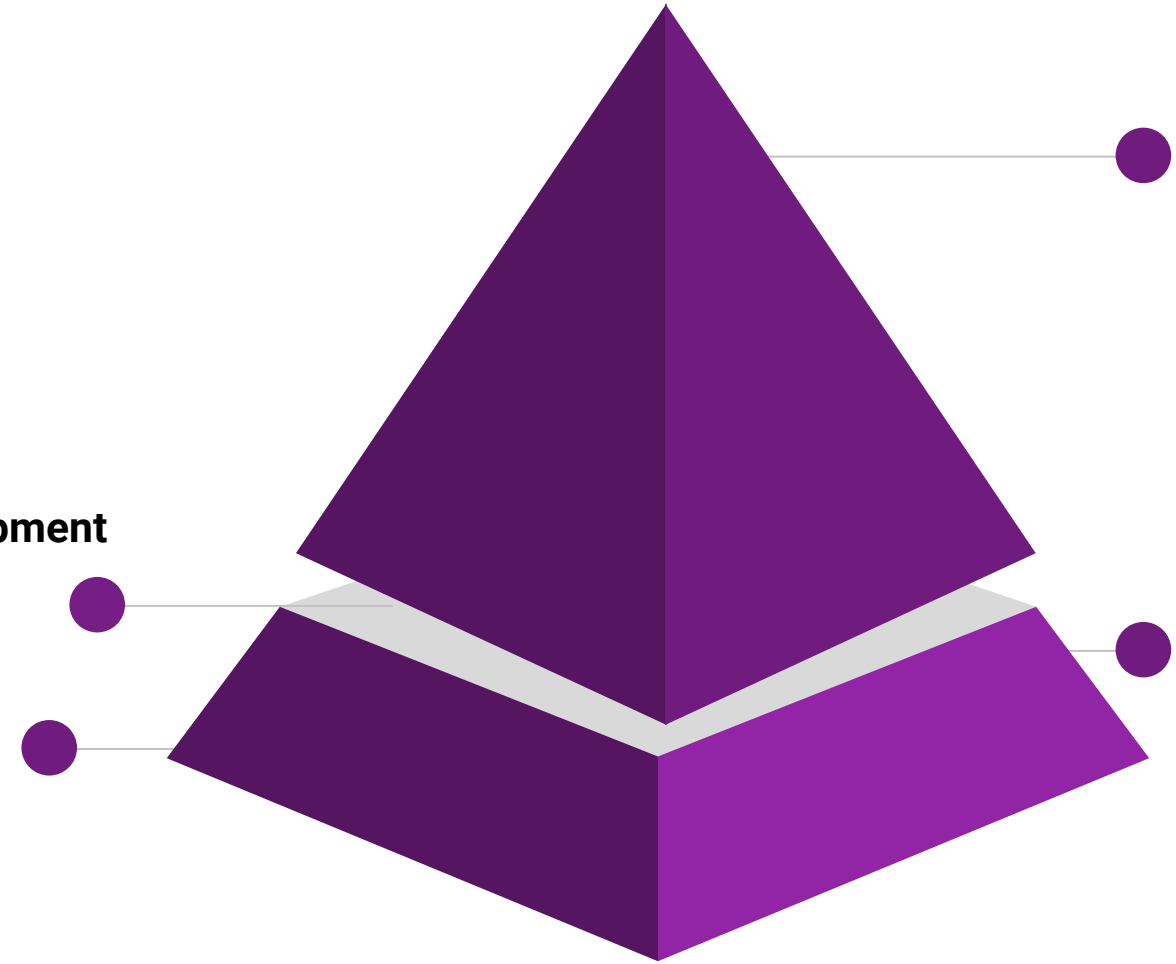


Infrastructure
The more the infrastructure is automated (IaaS/CSP) the easier it is, the easier it is also to miss use it. Don't make from a mouse an elephant.

Internal Developer Platform
An automated, developer centric, operations supporting hyper converged infrastructure. Often a result of containerization and Kubernetes.

The Cloud Native Pyramid

What the customer see



Infrastructure
The infrastructure is seen as the biggest problem - costly, complex, and huge amount of work goes into. Even or especially with cloud providers.

Business Requirement <-> Application Development

Internal Developer Platform

Cloud Native Mindset/Methodology

The Cloud Native Pyramid

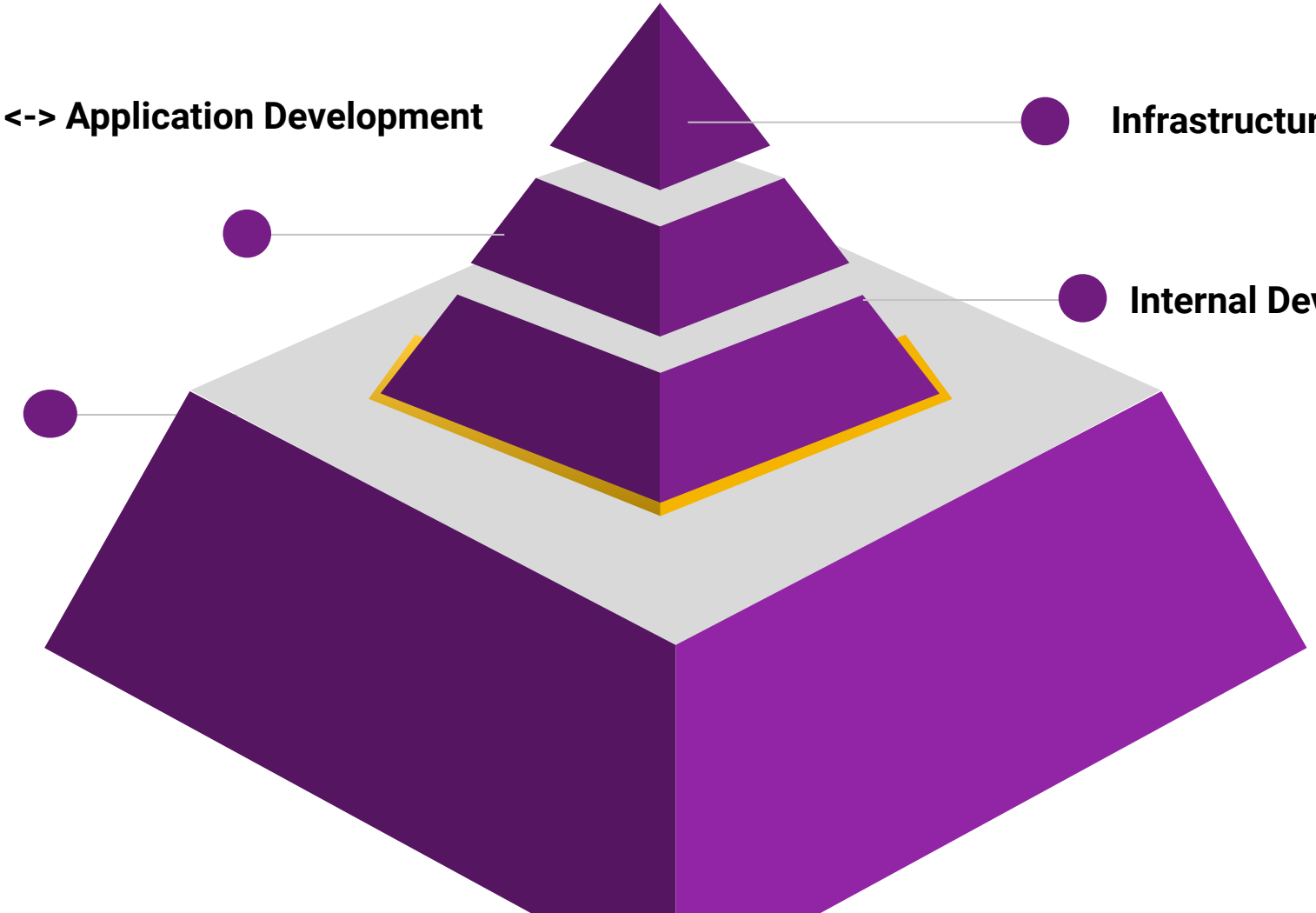
What we see

Business Requirement <-> Application Development

Cloud Native Mindset/Methodology
Your company success with the right mindset, your company fails if it refuses to change.

And you can't train or certify a mindset. It needs to be **cultivated**.

Disciplines/roles/jobs will mix and mingle.



Infrastructure

Internal Developer Platform

Cloud Based vs Cloud Native

Cloud Based	Cloud Native
Leaning towards stateful/sticky workload	Focusing on agnostic runtimes (containers) and serverless
Isolation focused - silos	Community driven distributed collaboration & communication
Semi automation - scripting heavy	Heavily automated - event driven native integration
Slow restarts - sensitive data handling	Fast restarts - data processed securely but disruptions are expected
Capacity overprovisioning	Capacity minimization - no overhead

~~Misconceptions~~ Observations

You can be the Queen of Cloud Nativeness, without ever touching a CSP

Kubernetes is a more dynamic hypervisor I just put my apps on, as they are	Kubernetes is two things: 1. A platform to build cross-platform platforms 2. A system to manage, enrich and run your applications
Kubernetes is treated as a one time implementation → build once, hand over to operations	Companies need to start thinking in products and not projects
We hand out some best practices & blue prints and are enabled to do a mass migration, so that we are getting cloud ~~native~~ naive	You have to actively guide and consult your ICT - typically the group of people understanding cloud native are very small

Apps often aren't made for Cloud Native

By design most software is not build for Kubernetes.

By missing experience & knowledge many of the green field implementations are not built for Kubernetes.

Only 27% of the respondents have ever used K8s, 59% used containers

Around 50% are not interested or never heard of K8s

23% doesn't know what K8s is used for

-CNCF Cloud Native Dev Report


How to stay on the Cloud Native side

Mindset & Strategy

01

You have to be a brain surgeon, not a sledge hammer

- Every cloud native shift depends on the mindset - that is something **you can't train**
- Strategic **mass migrations** towards cloud maybe speed up the process, but comes on **other costs** - frustration, knowledge gaps & wrong usage of cloud resources
- Identify the group of people who are burning for this topics and give them **anything they need**

You need to start by 0

To become cloud native

If you want to be a digital company, you can’t just migrate your engineering, banking, telco, chemistry, aerospace (...) capabilities to a cloud.

You only can enhance it that way.

Because, you are not a cloud born company, like Netflix, Spotify, Uber and co.

The mindset makes the difference

Companies choose CSP by $$$

On a shorthand, the hard facts about $$$ are easy to evaluate

What is never evaluated till the end are the technical capabilities and if they match your demand.

Companies even switch cloud providers because of money, forgetting about that the costs on the people, the migration and the knowledge loss can be higher than the savings.

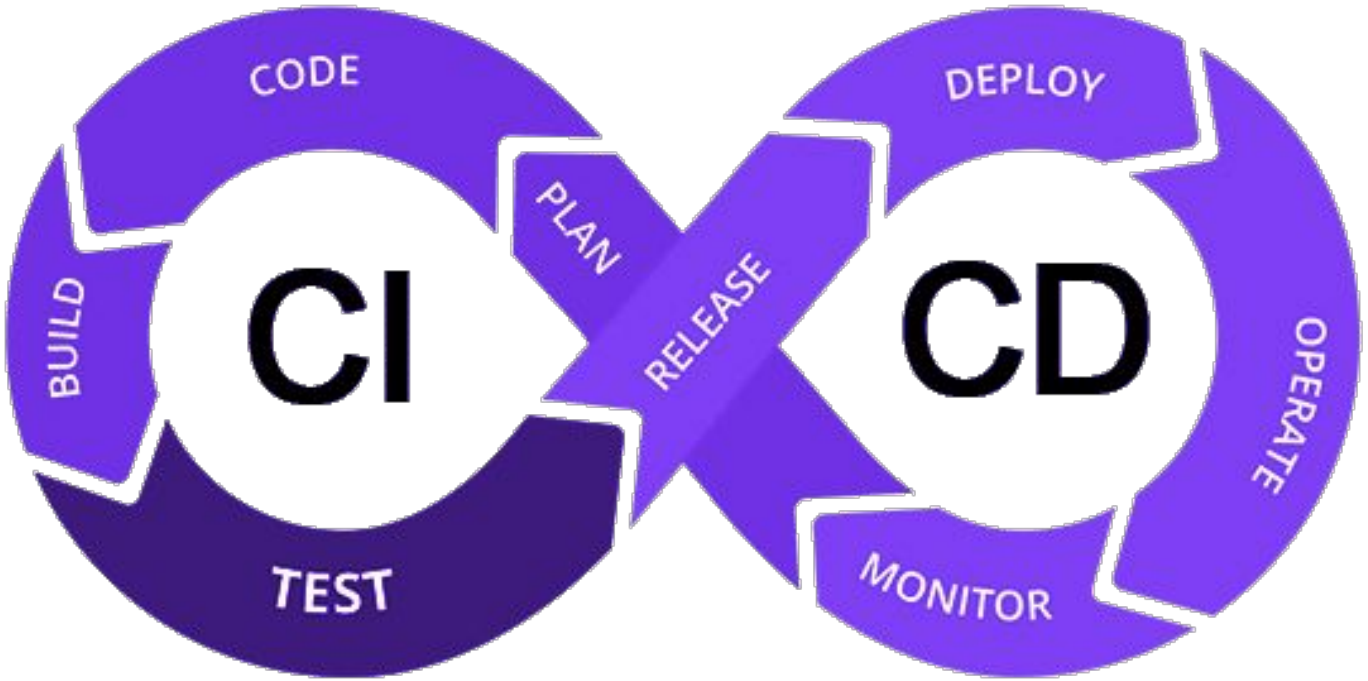
Mindset & Strategy

01	You have to be a brain surgeon, not a sledge hammer	• Every cloud native shift depends on the mindset - that is something you can't train • Strategic mass migrations towards cloud can speed up the process, but comes on other costs - frustration, knowledge gaps & wrong usage of cloud resources • Identify the group of people who are burning for this topics and give them anything they need
02	Start by 0, but demand any resources you can get	• Projects that want to reshape or re-architecture build already on ruins. • A cloud native journey can be successful if started without any technical debts or history • You have to give and dedicate any resource and budget the team needs, because they need to develop themself freely
03	Don't choose your cloud provider because of money	• So many "Strategic Decisions" for a Cloud Provider are taken because of good contracts - you will not save money if the CSP doesn't suits your requirements and the devil lays in the details

“Every” dev tool is critical

If your GIT is unavailable in cloud native there is no way to keep your infra & apps synced

If your deployment tool fails, you can’t play out your fix & patches



If your container registry is locked, you can’t deploy and not push any container

The platforms you build will be used forever

Building IDPs often happen within a timely terminated projects.

After it, normally budget cuts happen.

And the developers get shifted to another project.

Treat your IDPs as **live long product**:

- Continuously update the infrastructure
- Implement new integrations
- Implement new providers
- Improve current features

Don’t read, listen ;)

Pattern

Foundation

Fundamental principles in order to become good cloud-native citizens:

- **Declarative**
- Health Probes
- Lifecycles
- Automation
- **Predictable**

Behaviour

Communication mechanisms and interactions between the Pods and the managing platform:

- Batch & Periodic
- Daemon
- Stateful
- **Service Discovery**
- **Self Awareness**

Structural

Structuring and organizing containers in a Pod to satisfy different use cases:

- Init Container
- **Sidecars**
- Adapter
- **Ambassador**

Configuration

Customizing and adapting applications with external configurations for various environments:

- EnvVar Config
- **Config Resource**
- **Immutable Config**
- Config Template

Advanced

Complex topics which also doesn’t suit to the other categories:

- Controller
- **Operator**
- **Elastic Scale**
- Image Builder

Teach pattern

Pattern are old but gold, and give a unique understanding of the “how”

Because within Kubernetes, we observe a constant miss usage of such approaches.

- Config Pattern as database
- Daemon Pattern as distribution mechanism
- Sidecars & Ambassadors which calls in all different directions and between pods
- Operator which are built without operational logic

Development & Product Management

01	“Dev” tools are sometimes more critical than your production systems	• Treat anything used for build and deploy your apps as critical as your production systems - Git system, CICD, Artefact stores and co • A poor local development environment will not make a great cloud native solution
02	Don’t build for once, build for continuous change	• Platforms, like Kubernetes, are often introduced in a one shot project, and than put into maintenance mode • However, this is your **vehicle for the future**, you need to continuously work on it, and adjust it to your chaninge needs and the changing capabilities the open source community gives you
03	Understand microservice and Cloud Native pattern	• Dynamic changing platforms require that the applications they serve needs to be able to **take dynamics with comfort** • Stateless, event-driven, fault tolerant, decoupled

Kubernetes

01

Security, RBAC and Network Policies from Day One

- Introducing basic security actions like forbidding root or mounting host path, should be enforced by day one - also on Dev
- RBAC for some reason gets often no attention - change that!
- Just to start by a default deny all Network Policy will leverage your platform above most others in terms of security

Kubernetes

02

Do not let anyone skip the basics

- Kubernetes does a lot for you, but you have to accept and support it
- Utilize health checks & probes to maximize your service availability
- Understand the resource demand of your app and ensure your apps can scale horizontally

Multi or Single Cluster

There is no right or 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

Kubernetes

01	Security, RBAC and Network Policies from Day One	• Introducing **basic security** actions like forbidding root or mounting host path, should be enforced by day on - also on Dev • RBAC for some reason gets often no attention - change that! • Just to start by a default deny all Network Policy will leverage your platform above most others in terms of security
02	Do not let anyone skip the basics	• **Kubernetes does a lot for you**, but you have to accept and support it • Utilize health checks & probes to maximize your service availability • Understand the resource demand of your app and ensure your apps can scale
03	Single Cluster, Multi Cluster, Multi-Tenancy - Self or managed service?	• The most difficult question to answer, whatever you do → Don't do only one cluster for all, but also do not make one per app cluster • A central team providing a **self-service provisioning** mechanism with the ability to update clusters and natively integrate into the development process is the sweet spot you are looking for

Most important

The biggest mistake we always see

Avoid:

- Building complex chained CICD processes
- Self coded cloud development kits
- A self management portal per cloud/infrastructure
- Writing own operator where others have done the groundwork

Do not over engineer!

Don’t do it yourself

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

But, building tools or internal products often get abdomed what can cause severe security problems.

The community does a great job of answering and fulfilling needs and you can be part of it!

If you use open source ->

Clarify how you can contribute too

Photo by Kevin Kandlbinder on Unsplash

Summary

Building cloud native 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 ;)

Thank you!

Please visit us at Pavilion 2, SU32

mkoerbaecher

mkoerbi

REPLY
LIQUID