# Application Delivery to Kubernetes

A 101 to a fast-evolving ecosystem

Max Körbächer

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

# CLOUD NATIVE

You are not cloud native just because you run on an CSP

- Scalable applications
- Running in **dynamic environments**
- Based containers or functions
- Utilizing **declarative APIs**
- Structured in (micro) services

**“The cloud isn’t a place, it’s a way of doing IT” – Michael Dell**

# WHAT I MEAN

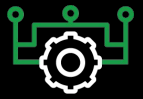
When I talk about cloud native and platforms, I think of

**Platforms**

We build platforms based on Kubernetes where application runs on.

**Application Delivery**

We implement tools to support the development and delivery of applications to the platform.



**Automation**

Everything is automated, tested and proved for reliability and zero-downtimes.

**Declarative**

We utilize APIs and declarative manifests to provision infrastructure, platform and delivery.

**The Foundation**

We build a trustful foundation for valuable solutions. This has to be reliable, secure and supporting the requirements of the applications.

# Common patterns & problems with Platforms

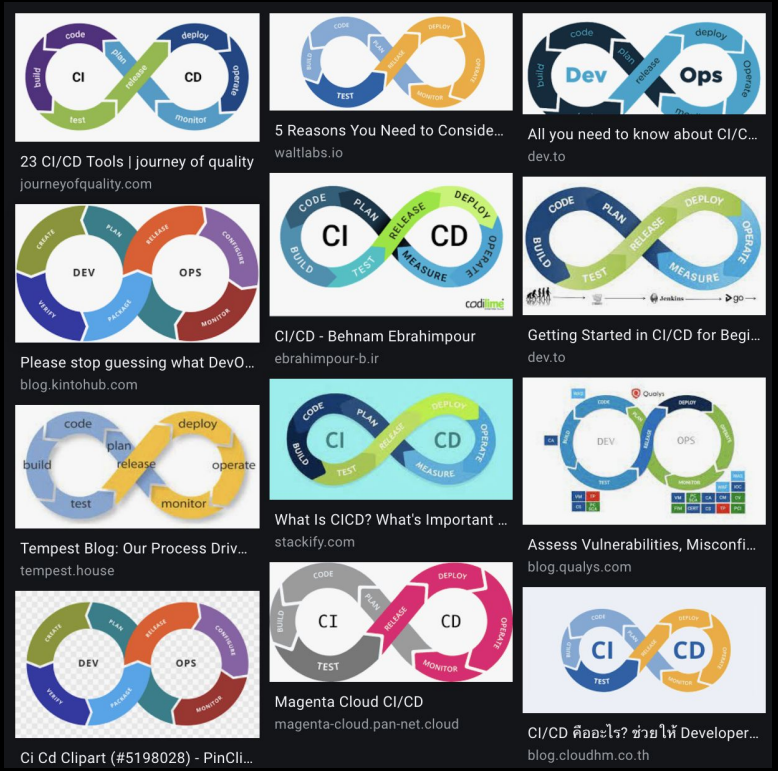
Platforms abstracts infrastructure complexities away.

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?

# How to deliver software for K8s

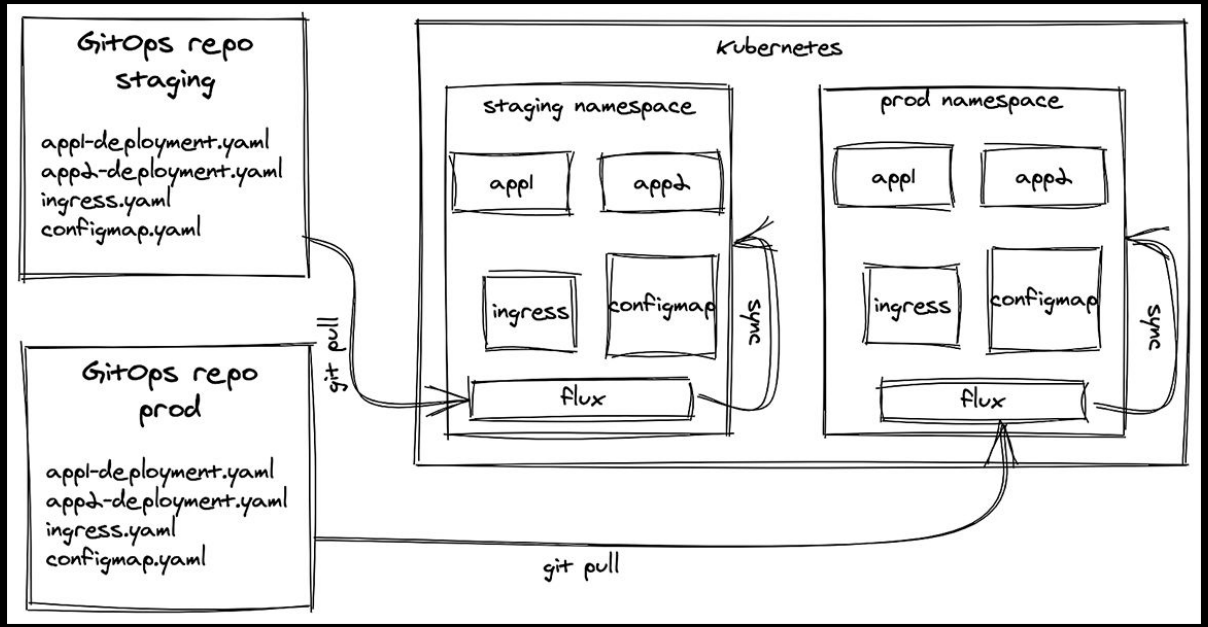
# The fairy tale of CI/CD




- Specially custom build pipelines
- “Hand Made”
- Yet another script

That’s not cloud native!

# The fairy tale of ~~CI/CD~~ GitOps



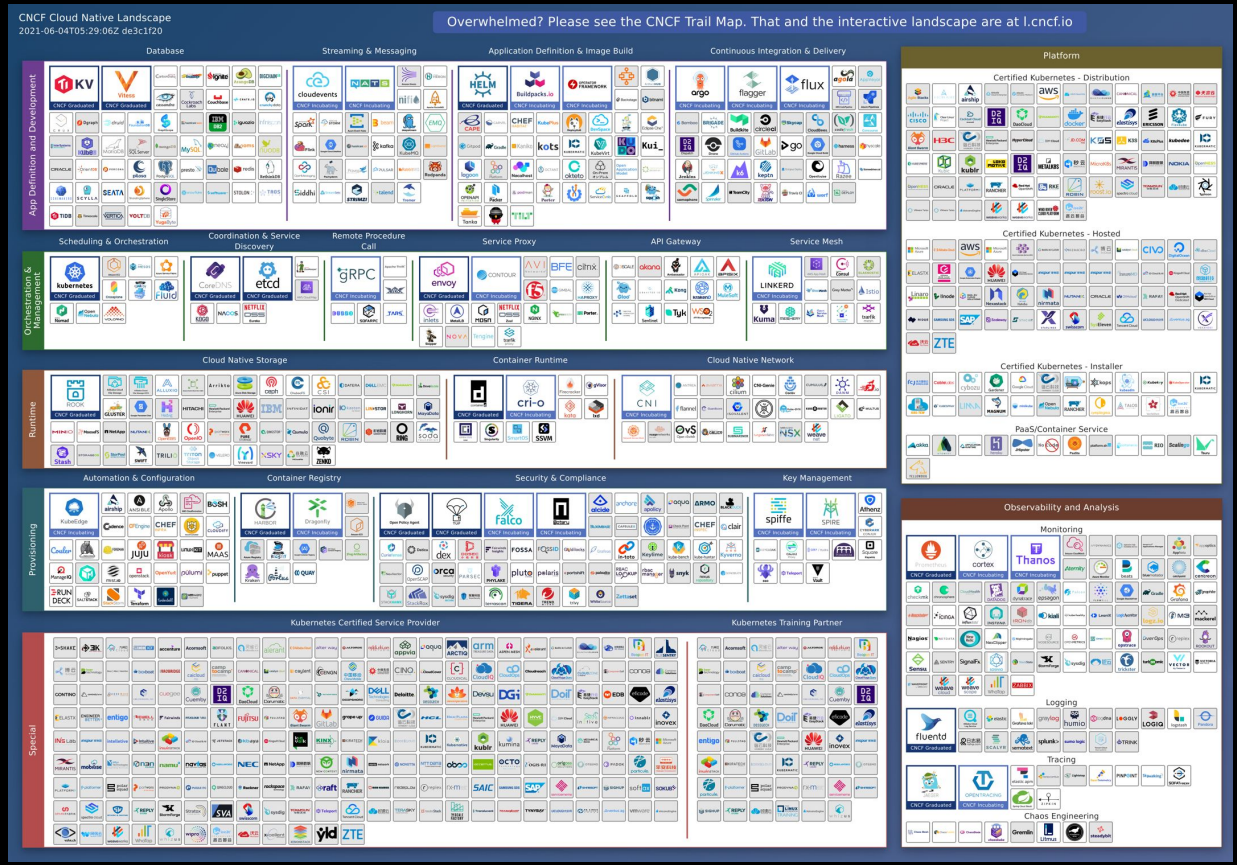

source: https://blog.container-solutions.com/fluxcd-argocd-jenkins-x-gitops-tools

GitOps only gives the answer to 50% of the story.

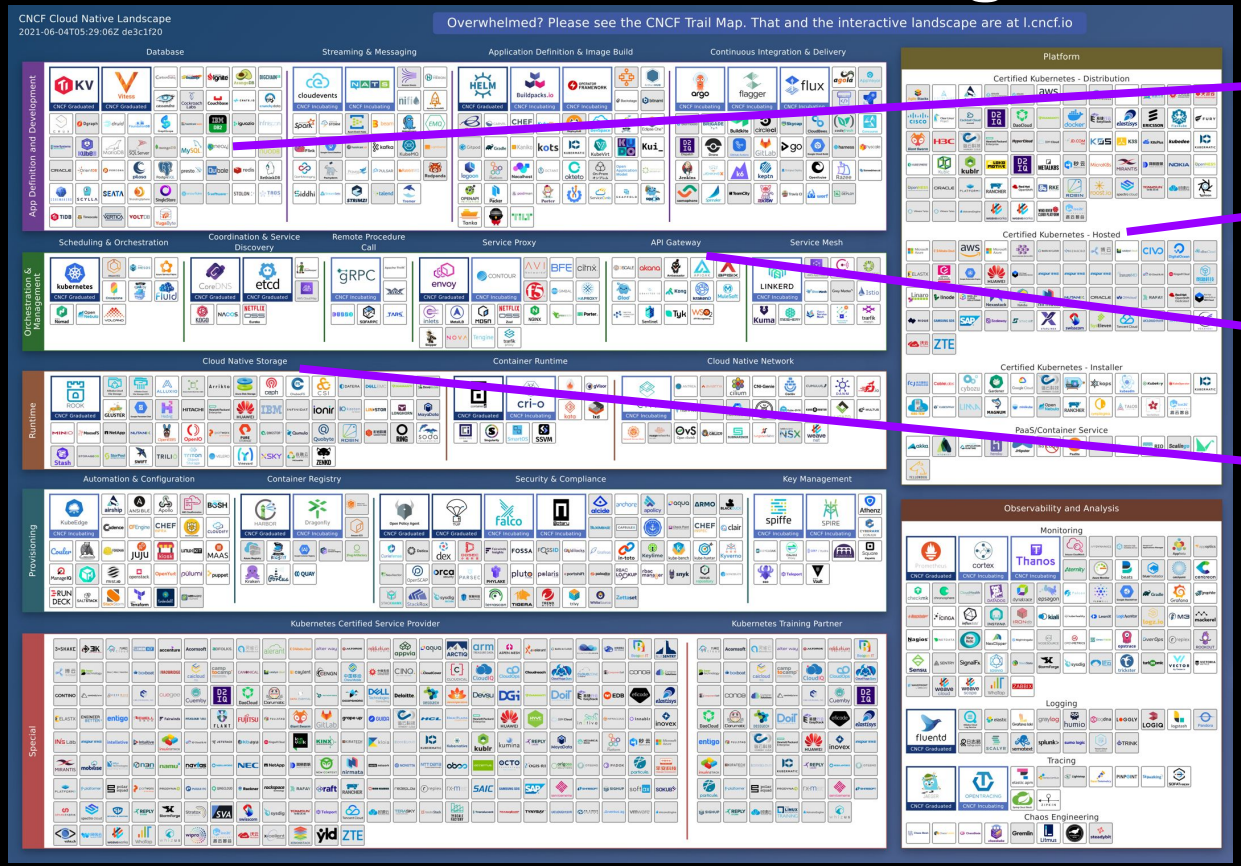
That's awesome, but not a 100% solution!

# None of it is a perfect solution

# Where to find the right tools?



# Where to start to find the right tools?



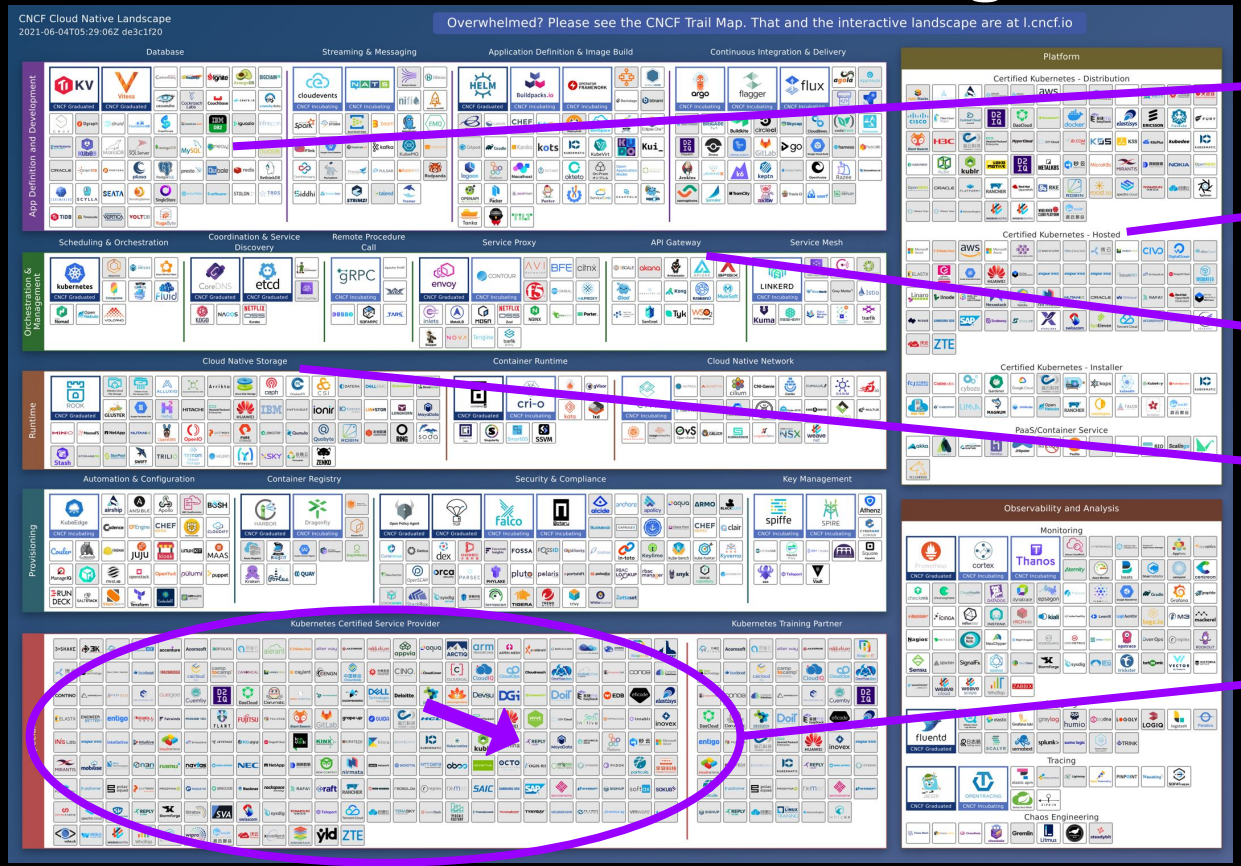

Databases

Hosted K8s

API Gateways

Storage

# Where to start to find the right tools?




Databases

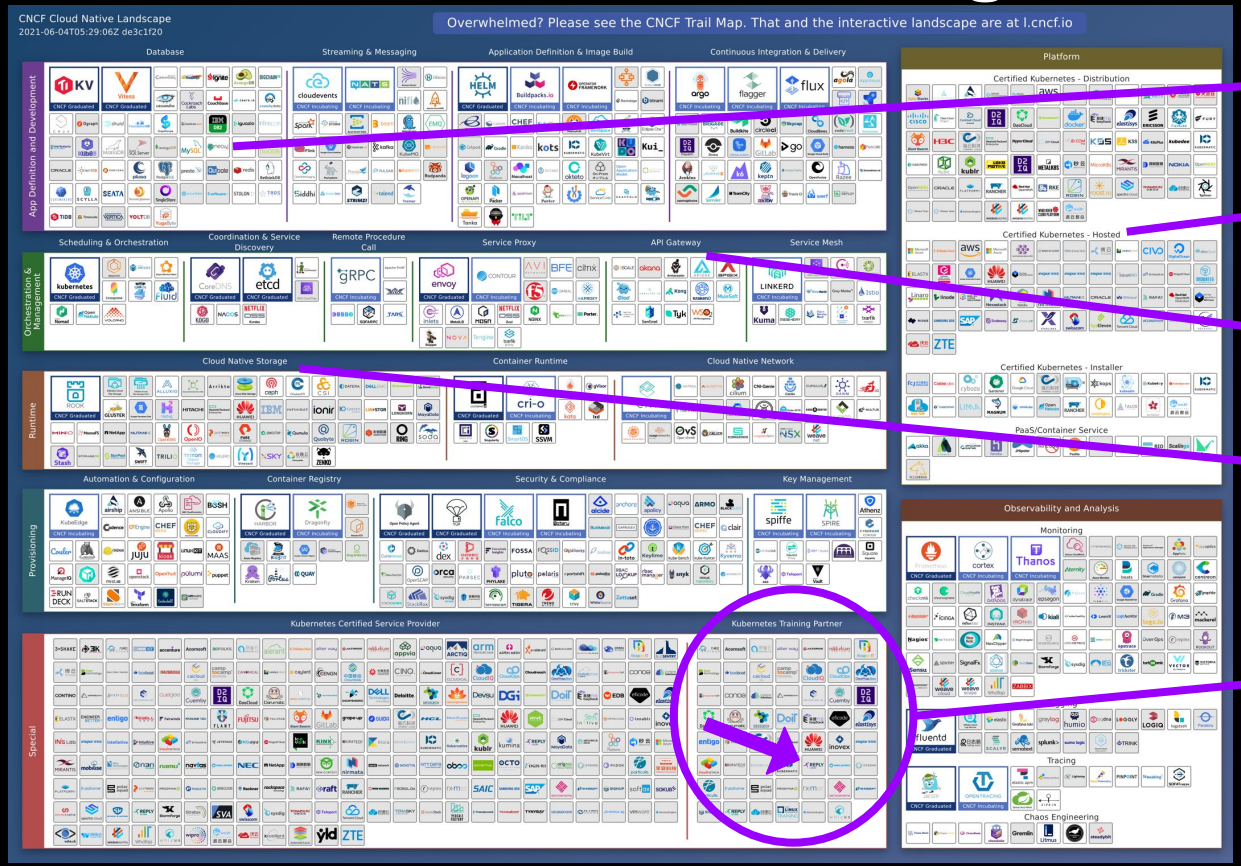
Hosted K8s

API Gateways

Storage

REPLY
LIQUID

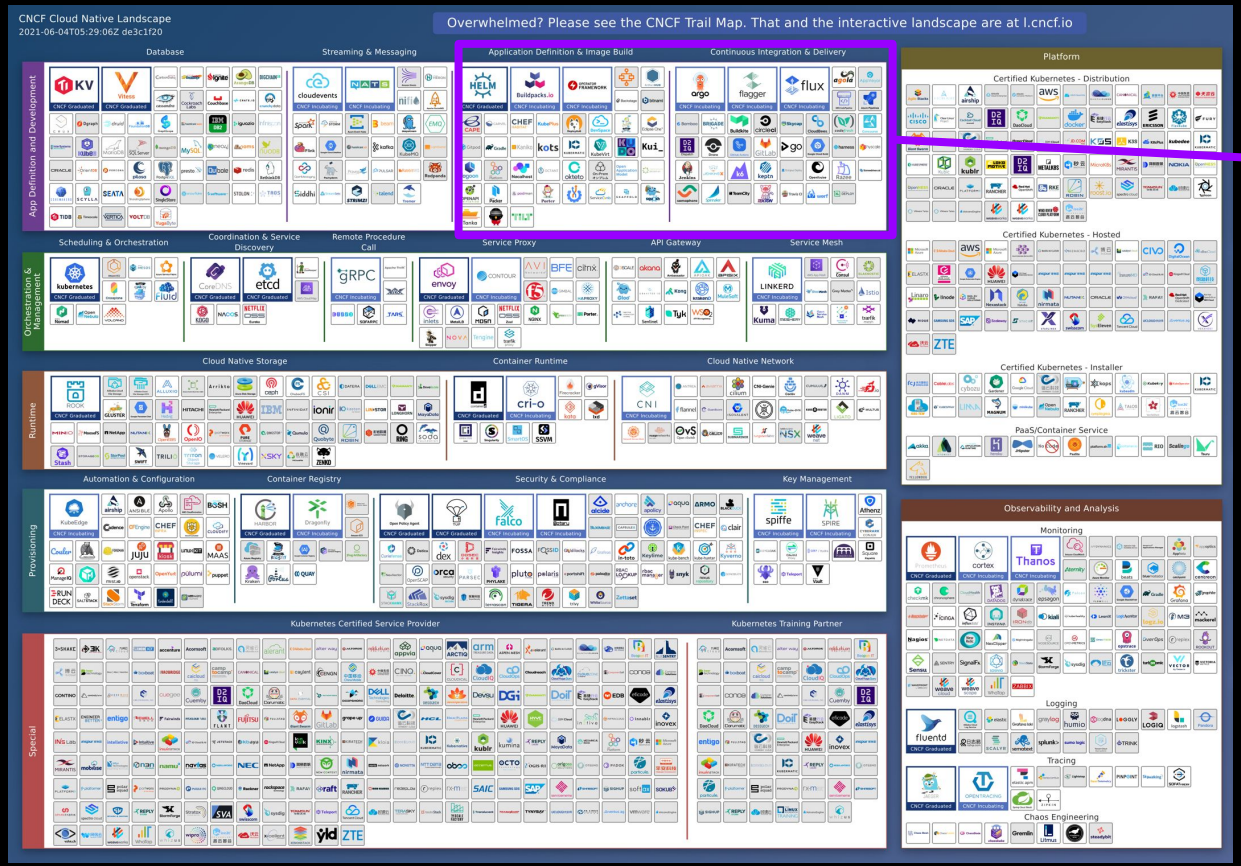
# Where to start to find the right tools?




- Databases
- Hosted K8s
- API Gateways
- Storage

REPLY
LIQUID

# We want to focus



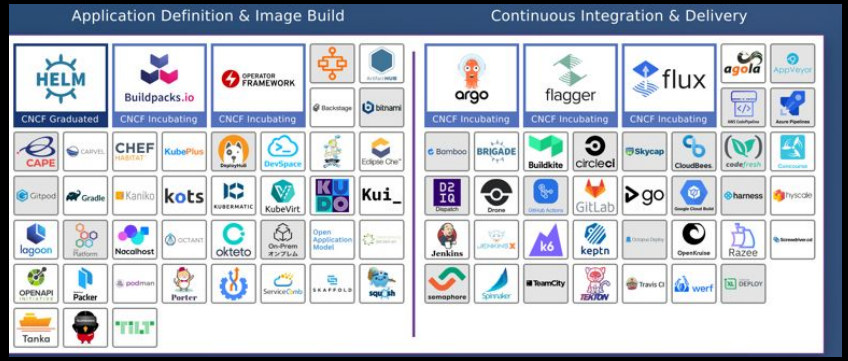

- Application Definition & Image Build
- Continuous Integration & Deliver (other than the standard CICD tools)

# 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

# BuildPacks

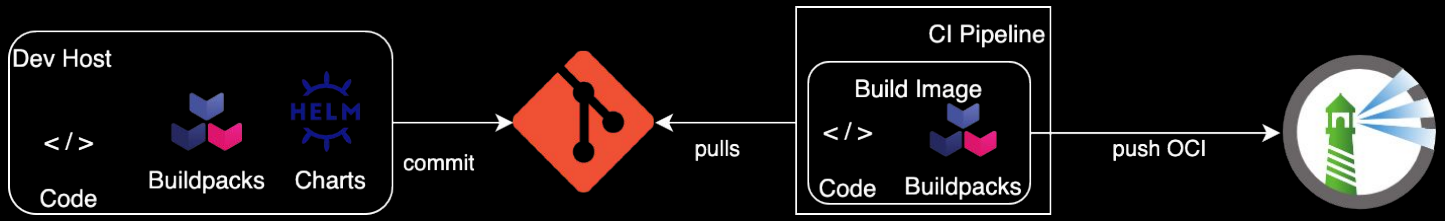
```
# Buildpack API version

api = "0.2"

# Buildpack ID and metadata

[buildpack]

id = "samples/kotlin-gradle"

version = "0.0.1"

name = "Sample Kotlin Gradle
Buildpack"

homepage =
"https://github.com/buildpacks/samp
les/tree/main/buildpacks/kotlin-gra
dle"

# Stacks that the buildpack will
work with

[[stacks]]

id =
"io.buildpacks.samples.stacks.bioni
c"

[[stacks]]

id =
"io.buildpacks.samples.stacks.alpin
e"
```

```
<Main.kt>

package org.kotlinlang.play


fun main() {

        println("Hello, World!")


}
```

```
<Stack Dockerfile>

FROM ubuntu:bionic as base

ENV CNB_USER_ID=1000

ENV CNB_GROUP_ID=1000

ENV
CNB_STACK_ID="io.buildpacks.samples.sta
cks.bionic"

LABEL
io.buildpacks.stack.id="io.buildpacks.s
amples.stacks.bionic"

RUN groupadd cnb --gid ${CNB_GROUP_ID}
&& \

  useradd --uid ${CNB_USER_ID} --gid
${CNB_GROUP_ID} -m -s /bin/bash cnb
```

```
pack build kotlin-sample \
--builder cnbs/builder:bionic \
--buildpack samples/kotlin \
--path samples/apps/kotlin/
```

# How to integrate BuildPacks in your daily doing?



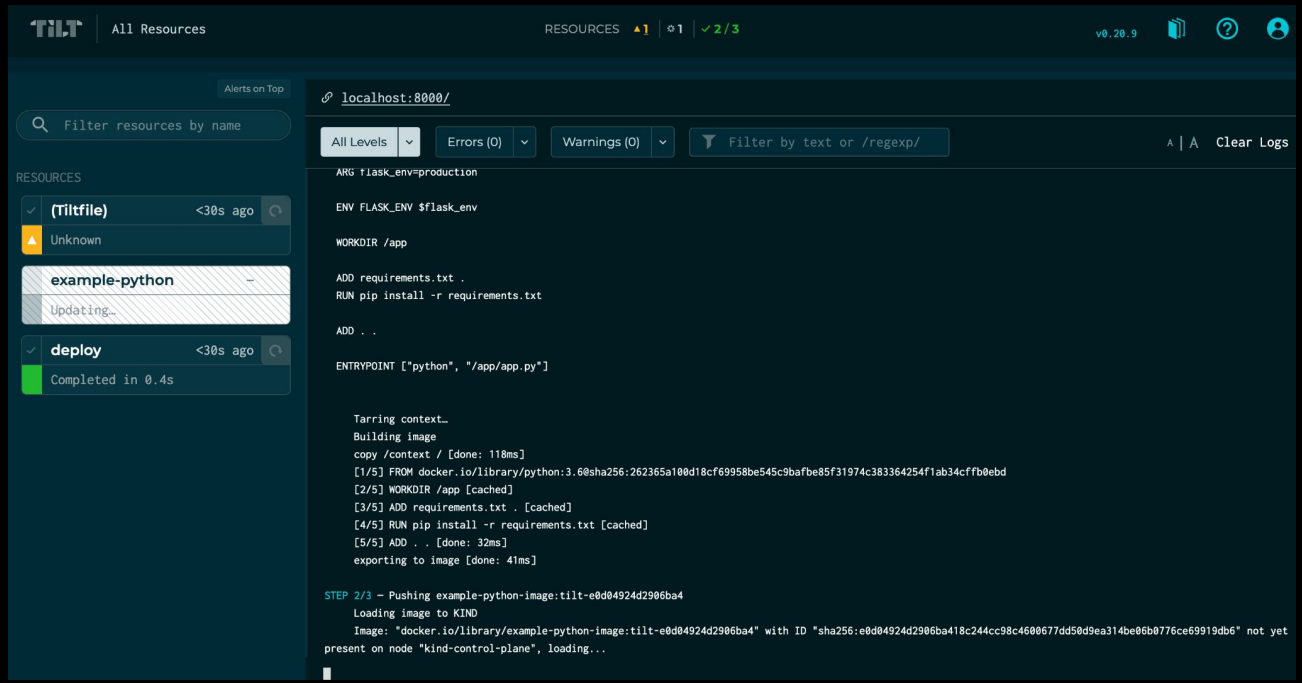

- Require a custom place to run the build
- Someone has to specify, test and develop the stacks
- Feels like chaining up again tons of tools
- Theoretically can be “replaced” by Dockerfile & build
- Still we have to talk about the deployment configuration

# Are CNBs better than the rest?

At least you should have a look at

| | Cloud Native Buildpacks | Dockerfile | source-to-image (s2i) | Jib | ko |
|---|---|---|---|---|---|
| Advanced Caching | Yes | No | Yes | No | No |
| Auto-detection | Yes | No | Yes | Yes | Yes |
| Bill-of-Materials | Yes | No | No | No | No |
| Modular / Pluggable | Yes | No | No | N/A † | N/A † |
| Multi-language | Yes | Yes | Yes | No | No |
| Multi-process | Yes | No | No | No | No |
| Minimal app image | Yes | Yes ♦ | Yes ‡ | Yes | Yes |
| Rebasing | Yes | No | No | No | No |
| Reproducibility | Yes | No | No | Yes | Yes |
| Reusability | Yes | No | Yes | N/A † | N/A † |

# Tilt



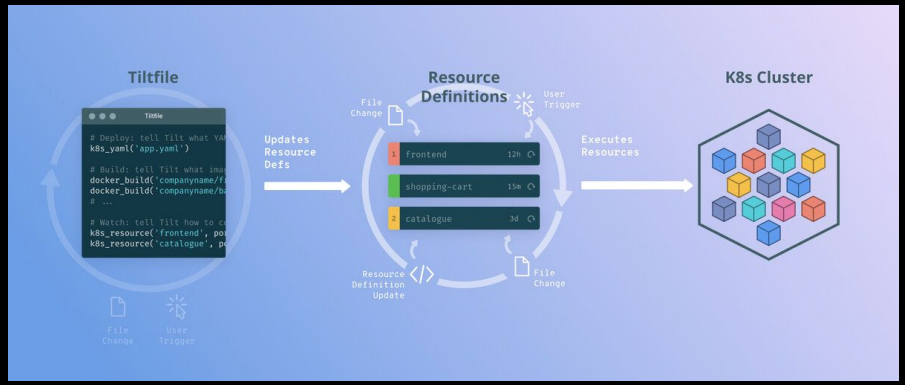

```
Tiltfile

local_resource(

        'deploy',

        'python now.py >
start-time.txt',

)

docker_build('example-python-image'
, '.')

k8s_yaml('kubernetes.yaml')

k8s_resource('example-python',
port_forwards=8000,


resource_deps=['deploy'])
```

.....

Demo

# Tilt for teams (?!)

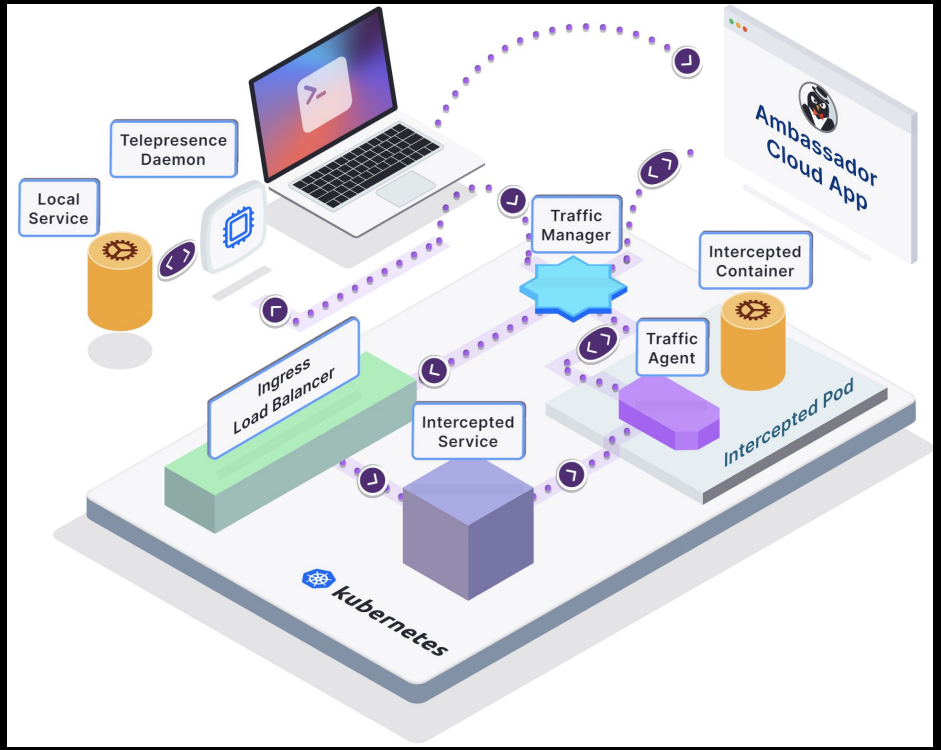
- Still requires docker running locally
- Focuses on K8s
- To interact with your team effectively, you need to use tilt cloud




source: https://docs.tilt.dev/controlloop.html

# Telepresence

Remote interception and debugging




Intercepts remote workload and allows to use your local tools to debug.

Bypass the deployment cycle to test fixes.

Only requires the resources of the services you test, no local k8s needed!

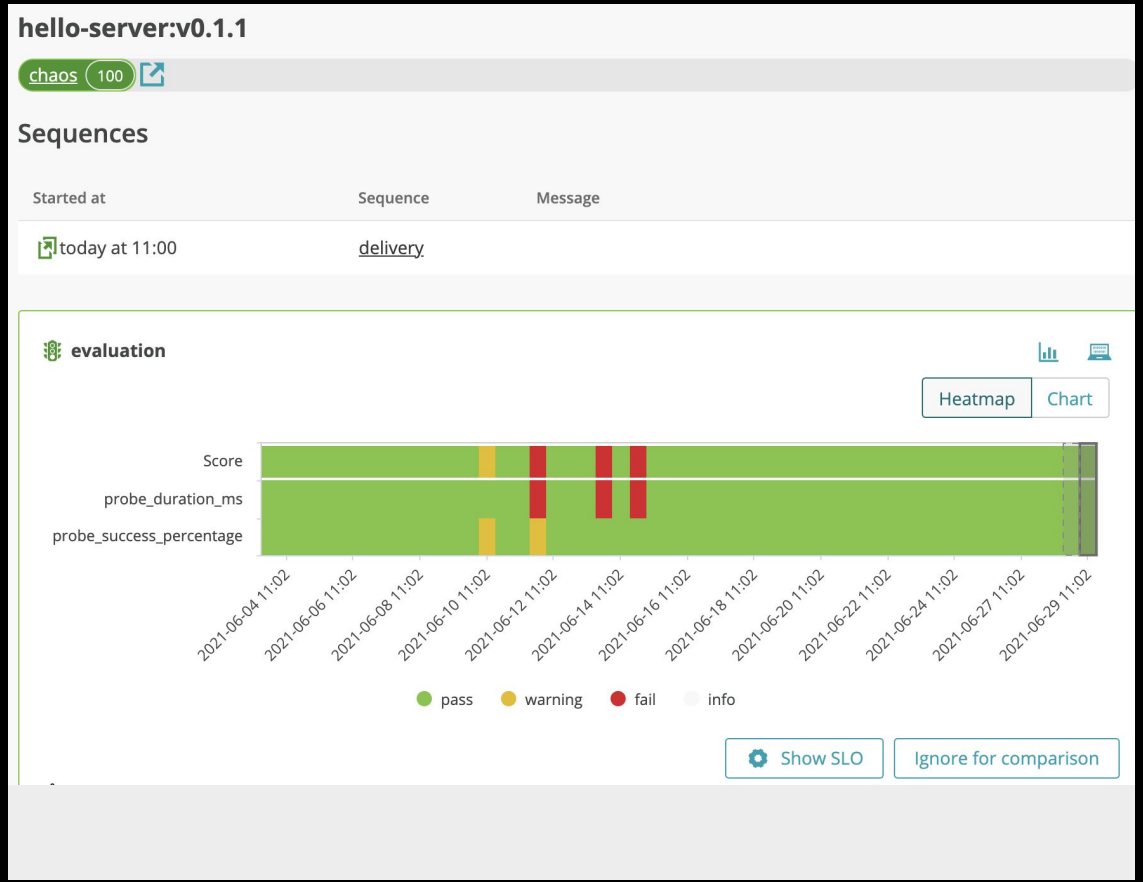
Source: https://www.telepresence.io/docs/latest/reference/architecture/

# Keptn

- Support you in a cloud-native application life-cycle
- Integrates with observability and alerting to provide SLOs
- All declarative
- Closed-loop remediation

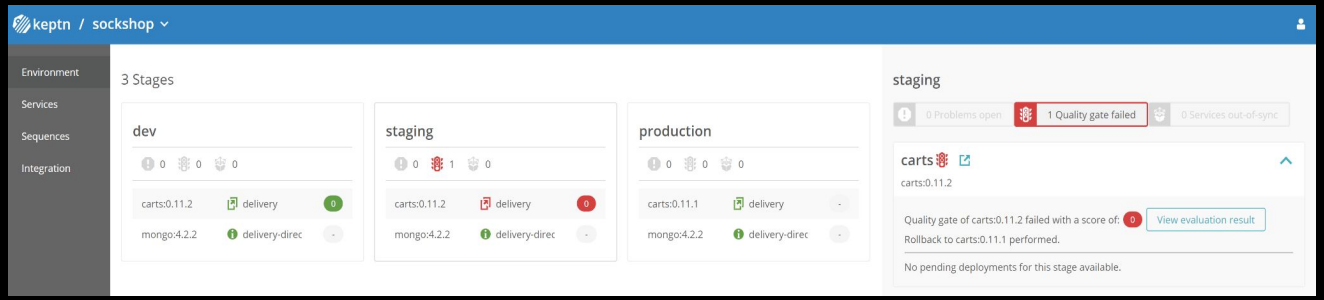
Live Demo System:

https://keptn.public.demo.keptn.sh/



# Keptn

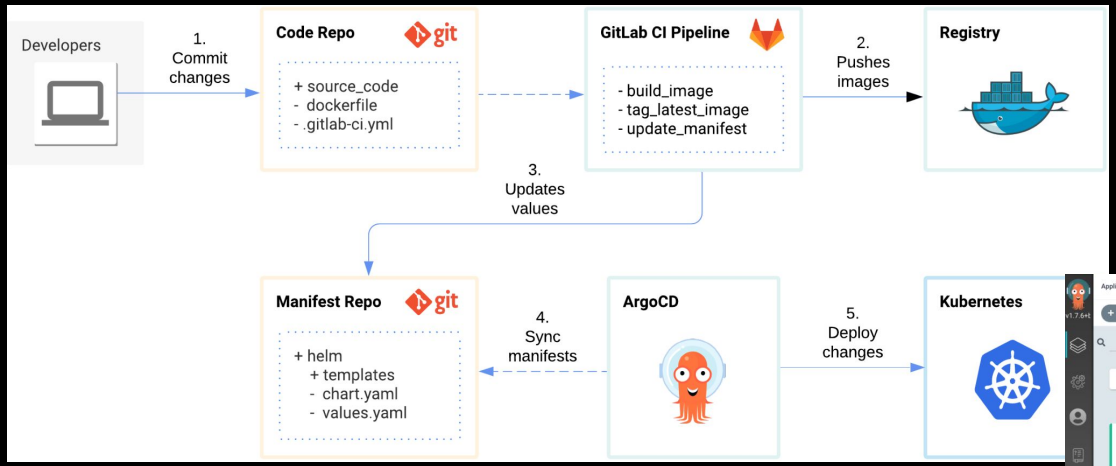
- Walks you through the deployment lifecycle
- Integrates:
  - CICD Systems
  - Observability Platforms
  - Testing Tools
  - Notification, further automation etc.

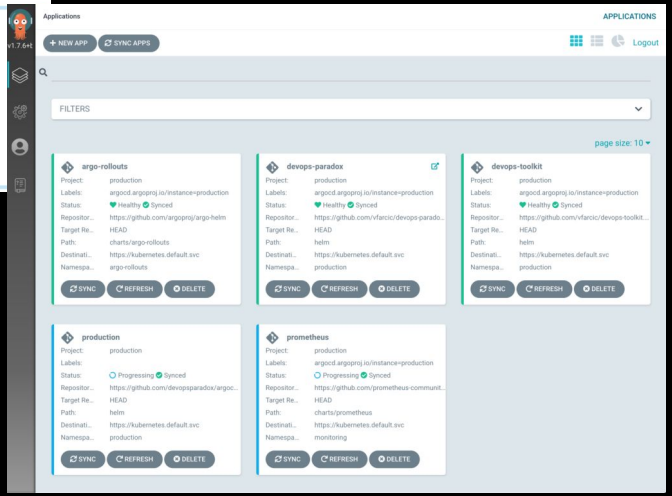



A SRE tool for developer?

# ArgoCD

GitOps with dev feedback

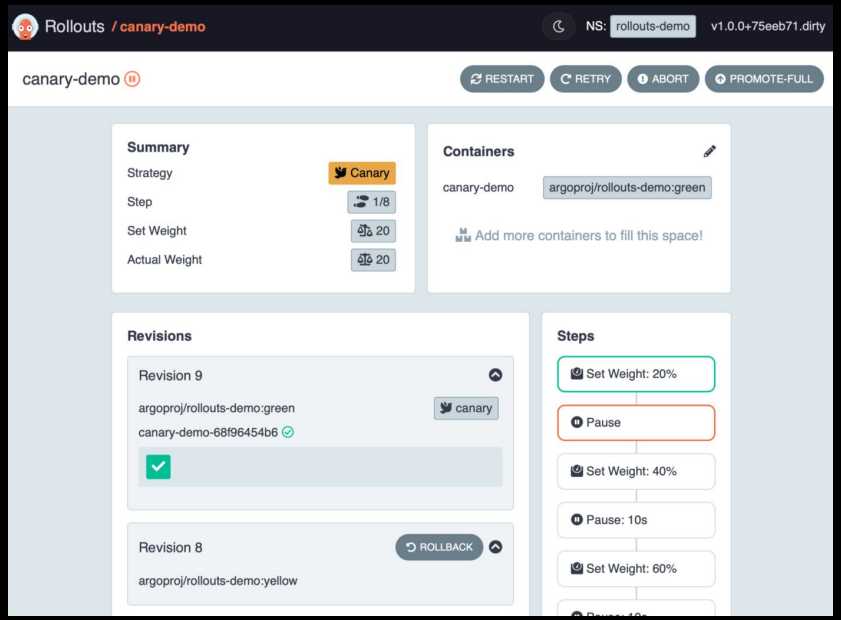


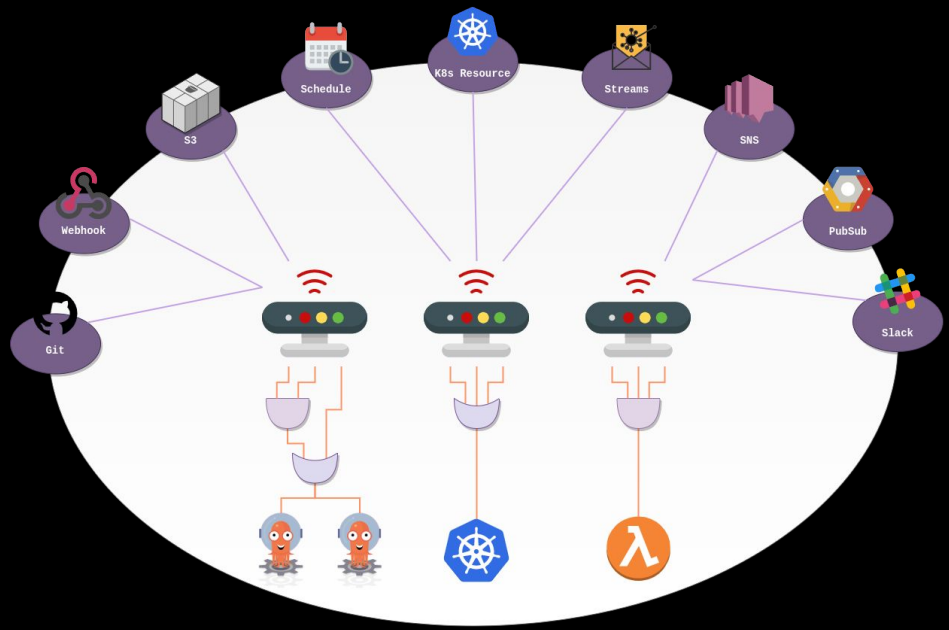




Demo

# Why ArgoCD is great?

It keeps evolving!



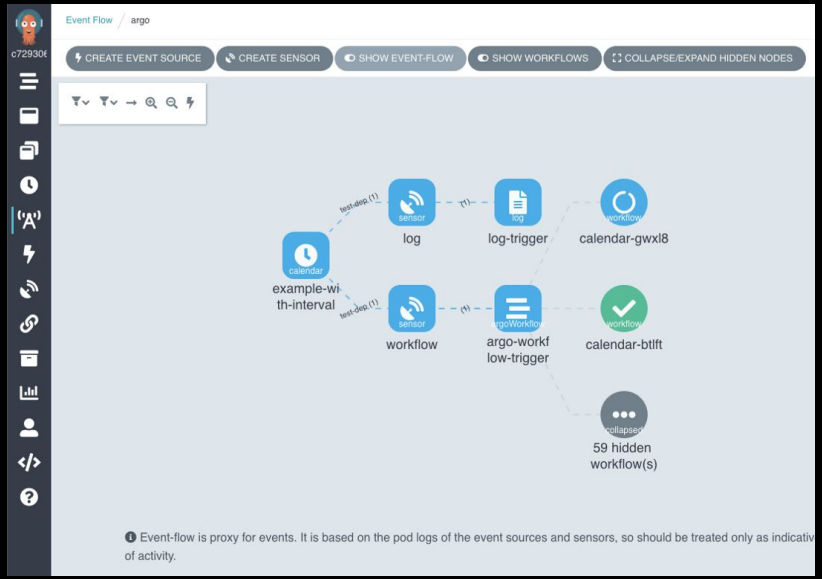




**Argo Rollout**
Progressive Delivery and fine-grained deployments of you apps

**Argo Events**
An event platform to trigger cloud native resources

# Why ArgoCD is great? - Part 2

It keeps evolving!




**Argo Workflow**

Workflow engine to parallelize and orchestrate jobs on Kubernetes

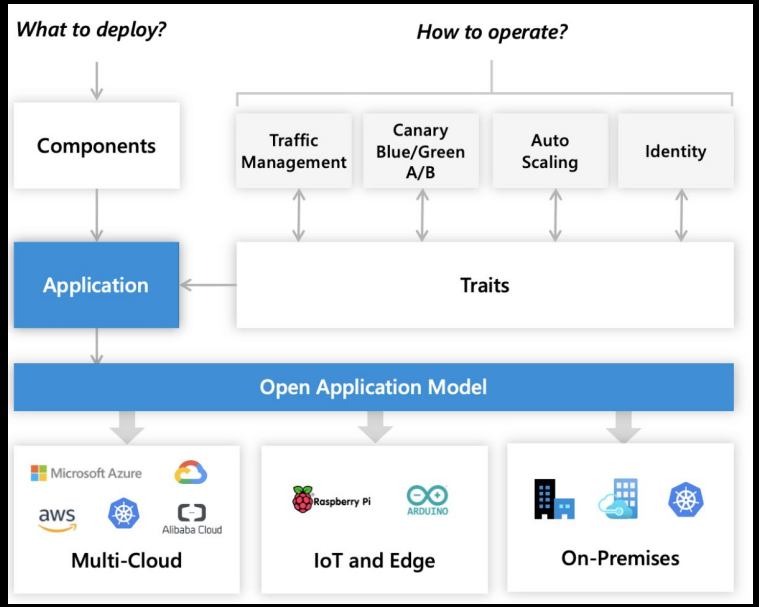
```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook
spec:
  generators:
  - list:
      # Parameters are generated based on this cluster list, to be substituted
      # into the template below.
      elements:
      - cluster: engineering-dev
        url: https://1.2.3.4
      - cluster: engineering-prod
        url: https://2.4.6.8
      - cluster: finance-preprod
        url: https://9.8.7.6
  template:
    # An Argo CD Application template, with support for parameter substitution
    # with values from parameters generated above.
    metadata:
      name: '{{cluster}}-guestbook'
    spec:
      project: default
      source:
        repoURL: https://github.com/argoproj-labs/applicationset.git
        targetRevision: HEAD
        path: examples/list-generator/guestbook/{{cluster}}
      destination:
        server: '{{url}}'
        namespace: guestbook
```

**Argo Application Sets**

Same app, multiple clusters, for unprivileged users

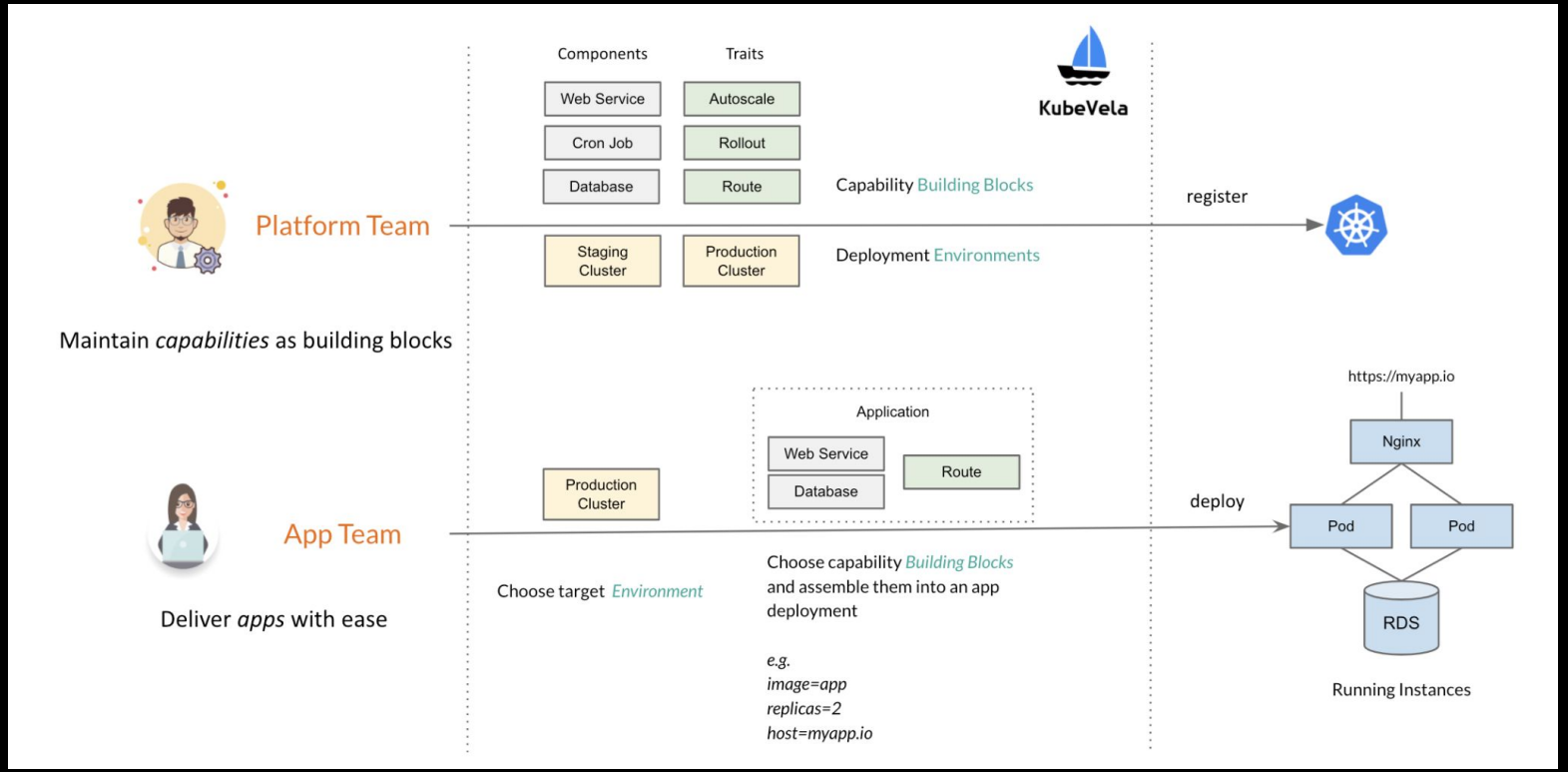
# Open Application Model

- Application first - define the app with a self-contained model, where operational behaviors as part of app definition
- Clarity and extensibility - an open standard to modularize infrastructure capabilities
- Runtime agnostic - a consistent experience to deploy and operate your apps across on-prem clusters, cloud providers or even edge devices



# KubeVela

A reference implementation for OAM




source: https://kubevela.io/docs/concepts/

# Summary

Some things we will not get rid of

## Deployment Manifests

How the app needs to run? How the app can run? What's the limits? When it is problematic?

You have to tell!

## Infrastructure “Stacks”

Not the servers, but the container images, CSP integrations and supporting services.

That’s why you have platform teams ;)

## New Roles, new Responsibilities

A 100% clear role for Dev, Ops and DevOps will not be possible.

I believe Platform Teams can mediate between the roles.

# Q&A

Let’s connect!


REPLY LIQUID