
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Paradox Of Choice: How To Pick an Application Definition That Works For You!

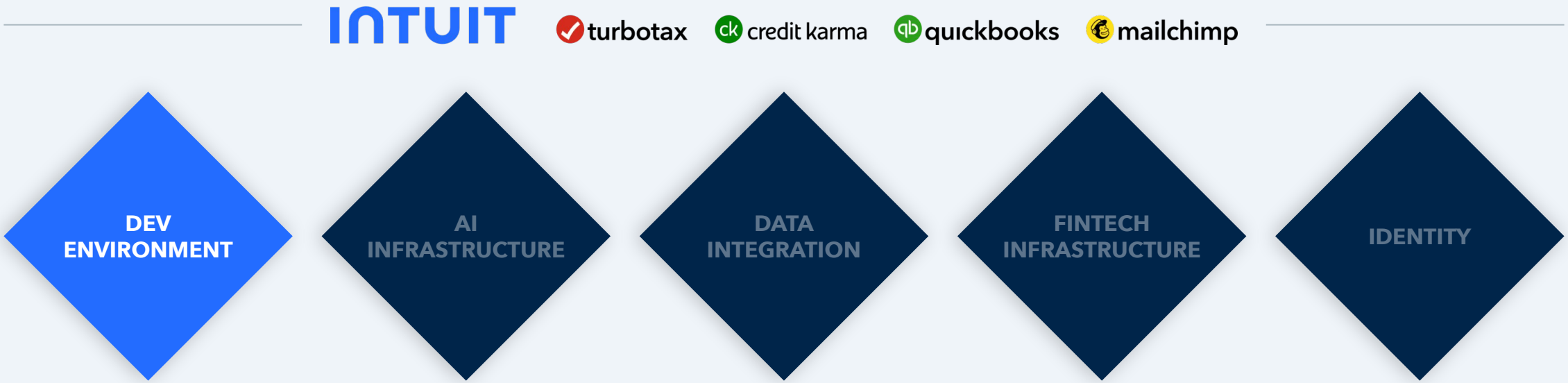
Anusha Ragunathan & Kevin Downey, Intuit Inc

Agenda

- Background
- Problem
- Paradox of Choice
- Solutions
- Results
- Takeaways

Background

AI-driven expert platform



Enable engineers to develop code in a fast, secure, and compliant fashion

~1M active CPU cores

6x increase in development productivity since 2019

900+ teams

16k+ namespaces

2k+ services

6k+ developers

Background: Intuit & our infra at a glance

Background: Intuit & our infra at a glance

900+
Teams

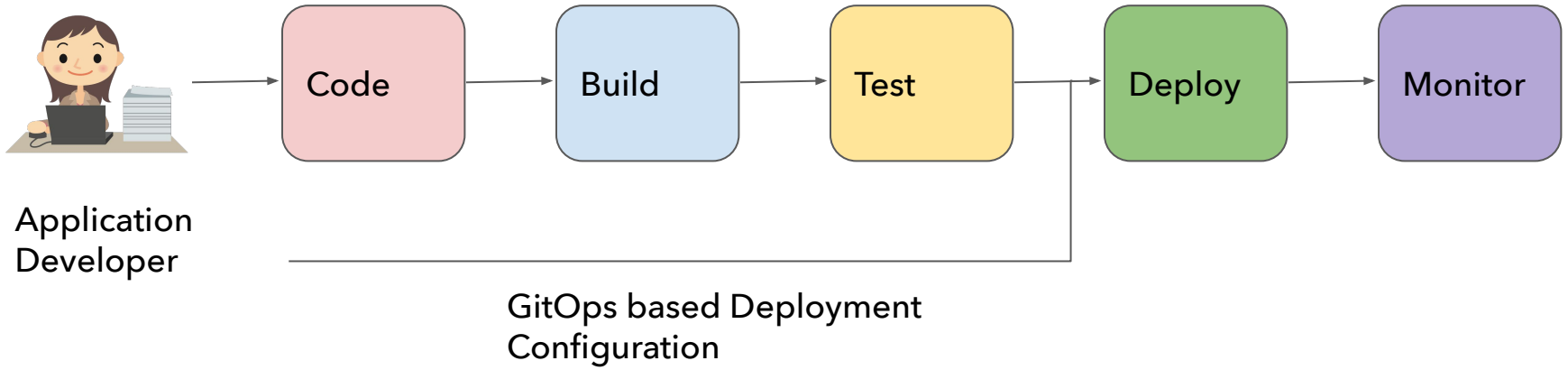
5000+
Developers

2000+
Services

245+
Clusters

16000+
Namespaces

Developer's CICD pipeline



Closer look into our deployment pipeline

Problem

Problem 1: Kubernetes & Cloud Complexities exposed

Application Developer

How do I set my horizontal scaling needs? minReplicas? maxReplicas?

hpa

Is a 15 seconds health-check-interval too low for my ALB Ingress object?

ing

What's the right maxUnavailable value for my PDB?

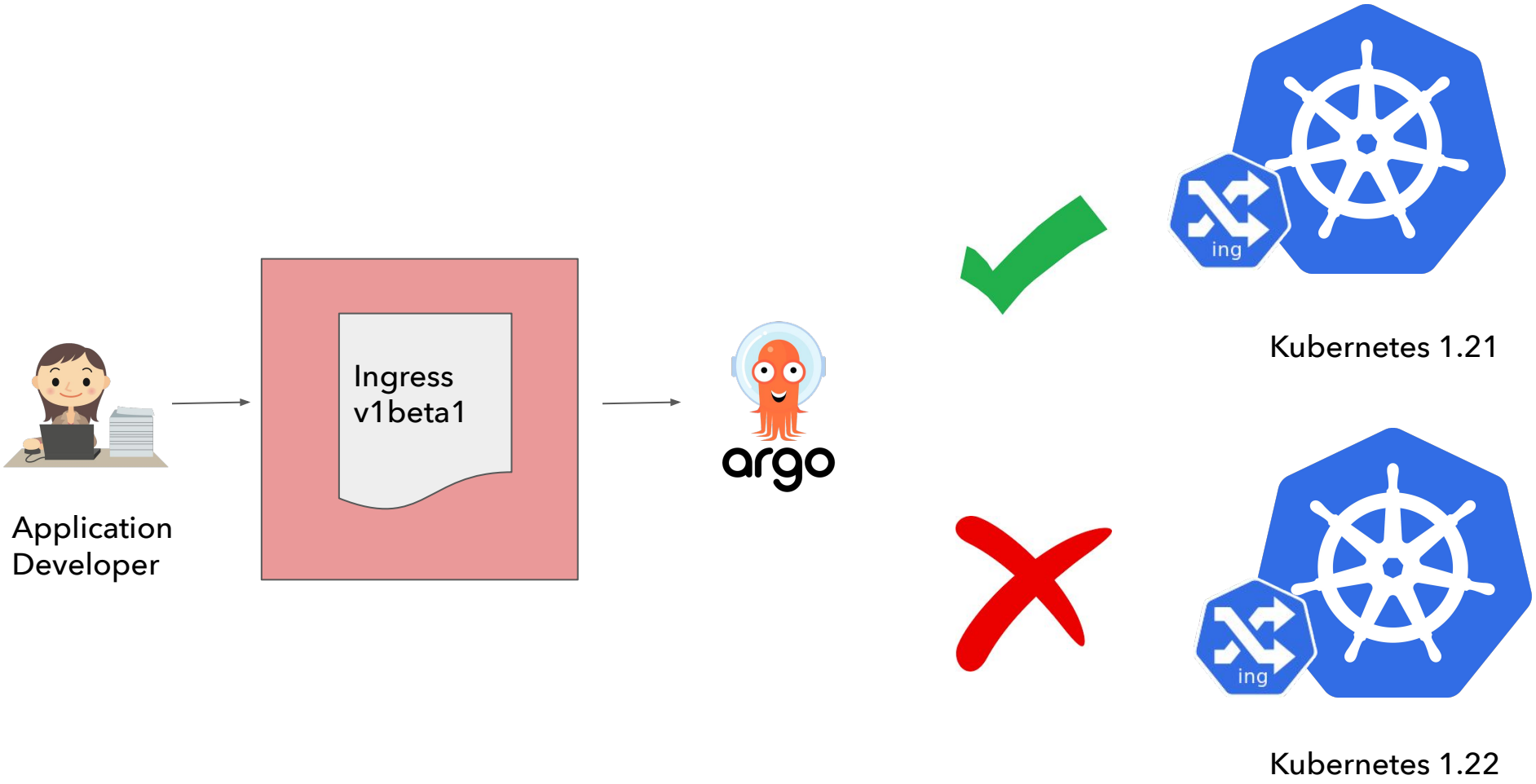
What are good CPU and memory Limits for my service?

limits

What quota should I set for my app's namespace?

quota

Problem 2: Kubernetes Deprecations exposed



Application Developer

Kubernetes 1.21

Kubernetes 1.22

Problem 3: Lack of operational input in app definition

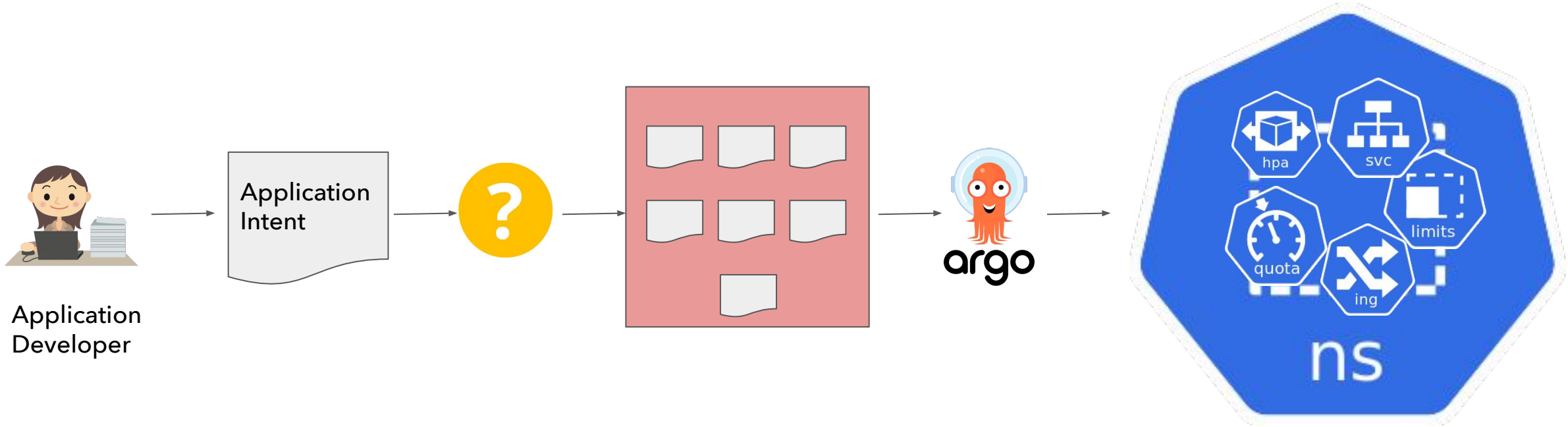
Application Developer

How can I enable High Availability for my service?

How can I enable active-active Disaster Recovery for my service?

Some services will accept external traffic. Can I spec that in the app definition?

Desired Target State



Paradox of choice

Application Definition & Image Build

HELM — CNCF Graduated

Backstage — CNCF Incubating

Buildpacks.io — CNCF Incubating

KubeVirt — CNCF Incubating

OPERATOR FRAMEWORK — CNCF Incubating

Artifact HUB

bitnami

CARVEL

CHEF HABITAT

KubePlus

codezero

DeployHub

Devfile.io

DevSpa

docker COMPOSE

Eclipse Che™

Gitpod

Gradle

itopia

Kaniko

KONVEYOR

kots

KRATOR

KUBERMA

KubeOrbit

KubeVela

KUDO

Kui_

lagoon

mia Platform

MONOKLE

Nocalhost

OCTANT

okte

On-Prem
オンプレム

Open Application Model

OPEN SERVICE BROKER API

OPENAPI INITIATIVE

HashiCorp Packer

podman

Porter

raftt

sealer

ServiceComb

SHIPWRIGHT

SKAFFOLD

squash

Tanka

TELEPRESENCE

Tilt

But first, the App Spec!

Requirements:

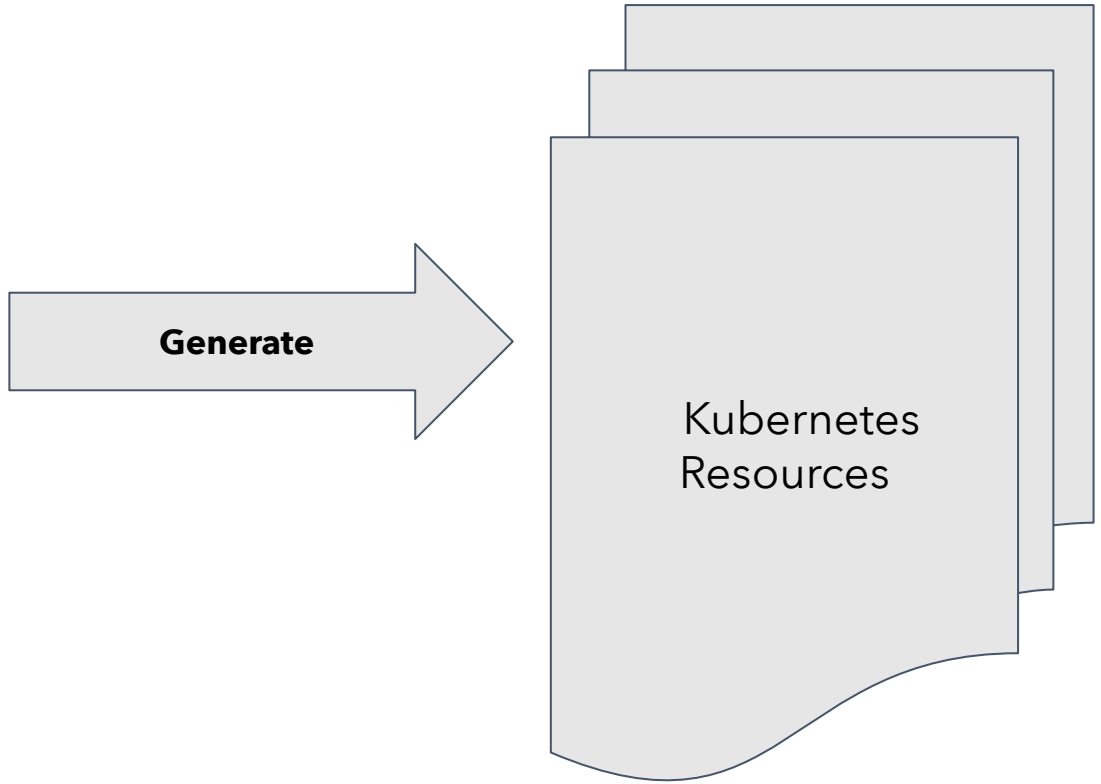
- Need an **application centric** specification.
- Need a specification that can help deploy and **operate**.

Choices:

- **Open Application Model (OAM) style specification**
- Templating style specification

Criteria: Simple app spec

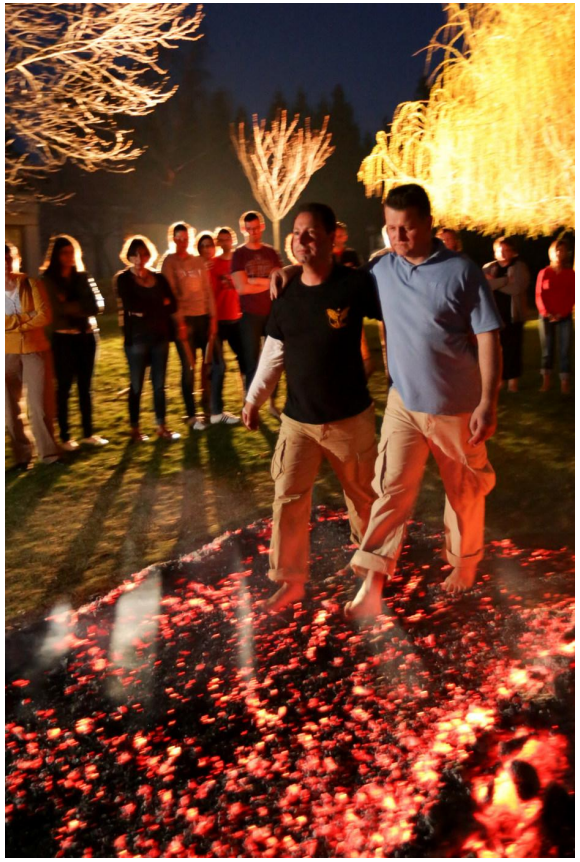

```yaml
apiVersion: iks.intuit.com/v1beta1
kind: ExpressApplication
metadata:
  name: kubeconNA2022-app
spec:
  components:
    - type: webservice
      name: blitz-webservice
      image: docker.intuit.com/devx/awesometeam/awesomeservice:v1.0
      traits:
        - type: sizing
          properties:
            horizontal:
              size: small
            vertical:
              size: small
  environments:
    preprod:
      - name: qal
        overrides:
          - type: webservice
            name: blitz-webservice
            traits:
              - type: sizing
                properties:
                  horizontal:
                    size: medium
                  vertical:
                    size: medium
```



Solutions

Choices: Just PoC it!

Into the fire!



HELM

Client Utility

Crossplane

Controlplane

Kustomize KRM Function

KubeVela

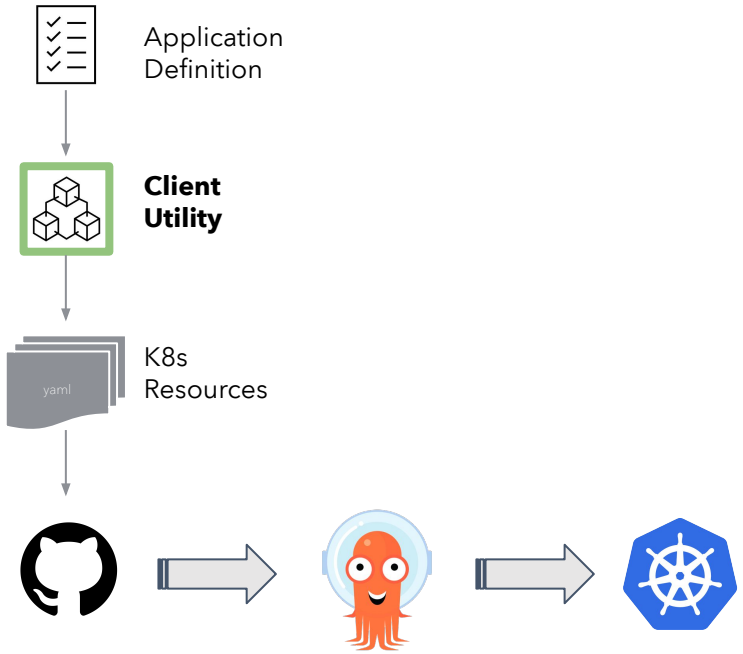
PoC Breakdown: Client Utility

Packages Kubernetes configs as Charts and provides deployment application lifecycle. CNCF Graduated project.

Kustomize KRM Functions

Client-side functions that operate on Kubernetes Resource Model (KRM) configuration. Supports Generators, Transformers and Validators as Kustomize Plugins.

CD Pipeline for Client Utility



PoC Breakdown: Controplane

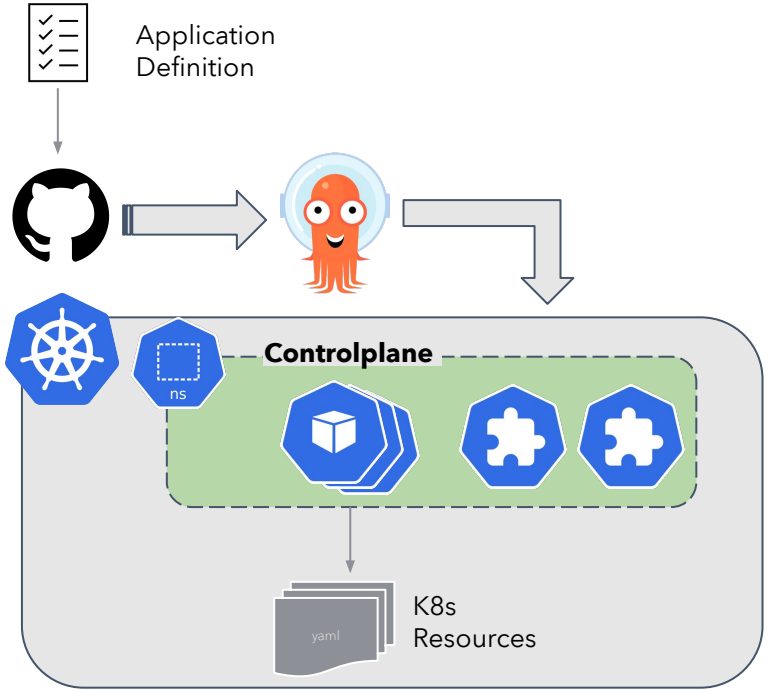
Crossplane

Open source control plane framework, orchestrate applications and infrastructure.

KubeVela

Open source application engine based on Kubernetes and OAM (Open Application Model).

CD Pipeline for Controlplane



Results

Comparison

Pros

- GitOps Compatible
- Active Community

Cons

- Chart of Charts
- Template Logic

Kustomize KRM Functions

Pros

- GitOps Compatible
- OAM Compatible
- Logic-less Templates

Cons

- Alpha Status
- Sparse Docs

Crossplane

Pros

- OAM Compatible
- Extensible
- Multi-Cloud

Cons

- Multiple Controllers
- Lackluster ArgoCD
- Limited Features
- No GitOps

KubeVela

Pros

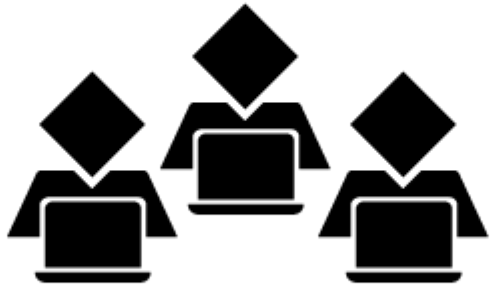
- OAM Compatible
- Extensible
- Multi-Cloud

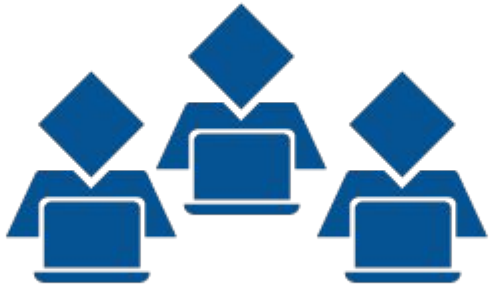
Cons

- Multiple Controllers
- All or nothing
- CUE Template DSL
- No GitOps

Different Camps

The result of the PoCs were just more advocates for each solution, no clear winner.





Survey

Let's Survey?

Rankings (1 - 5)					
Criterion	**Weight**	KubeVela	Helm	Kustomize	Crossplane
Technical Fit	5	4.0	4.0	4.0	4.0
Time to Market	5	3.0	4.0	4.0	3.0
Operability/Scalability	4	4.0	4.0	4.0	4.0
Maturity	4	3.0	5.0	5.0	3.0
Skills Required	4	4.0	4.0	4.0	3.0
Simplicity	4	4.0	4.0	4.0	4.0
Flexibility	4	3.0	4.0	4.0	3.0
Debuggability/Testability	4	3.0	5.0	5.0	3.0
Documentation	3	3.0	5.0	4.0	4.0
Community	3	4.0	5.0	4.0	4.0
Code Readability	3	4.0	4.0	4.0	4.0

Rankings Comparison

What to make of these results?

Client vs. **Controlplane**

Clients clearly won!

Why?

Time to Market

- Learning curve
- How much effort

Technical Fit

- Compatibility

Operability/Scalability

- Client vs. Controlplane

Flexibility

- Code vs. DSL

Results: Helm vs. Kustomize KRM

Helm

Time to Market
- ✅ Learning curve
- ❌ How much effort

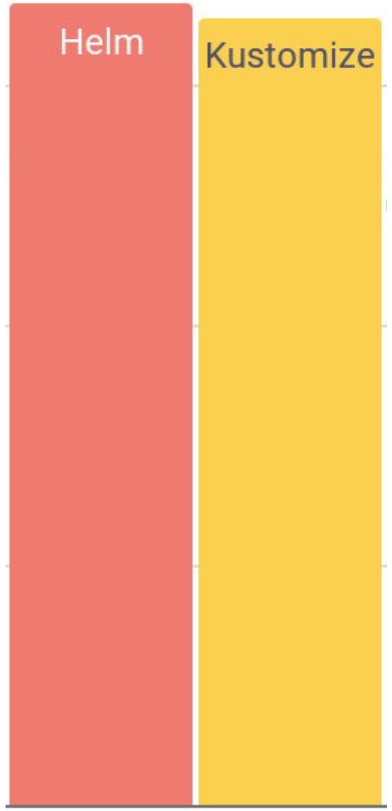
Technical Fit
- ❌ OAM Compatible

Operability/Scalability
- ✅ CD Pipeline Utility

Flexibility
- ✅ Code, Yamls, Templates

Community
- ✅ Active Users

Documentation
- ✅ Examples, User Guides



Kustomize KRM Functions

Time to Market
- ✅ Learning curve
- ✅ How much effort

Technical Fit
- ✅ OAM Compatible

Operability/Scalability
- ✅ CD Pipeline Utility

Flexibility
- ✅ Code, Yamls, Templates

Community
- ❌ Active Users

Documentation
- ❌ Examples, User Guides

Results: Kustomize KRM Functions Wins

Kustomize KRM Functions

Supports Generators, Transformers and Validators as Kustomize Plugins

Supports declarative specs (KRM)

GitOps Compatible

Demo

Takeaways

- When faced with the CNCF Paradox of Choice, you can use a **methodical, data-driven** approach to pick a solution that's appropriate for you.
- Abstracting application developers from the complexities of Kubernetes and cloud is doable. **KRM plugins** are an efficient client side implementation to achieve this.
- We observed that client solutions are suited for GitOps based deployments; **control-plane** solutions maybe be more suited for **non-GitOps** based deployments.
- **Velocity** and innovation of platform teams will improve with application abstraction.
- **Speed to Benefit** - use what you know, use what you have and avoid unneeded complexity

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Please scan the QR Code above to
leave feedback on this session