Platform Real State: Abstract Your Organization’s Tenancy Model Away with Crossplane

@mestredelpino

Who am I?

@mestredelpino

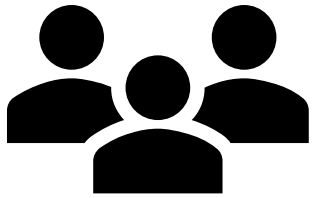
Carlos Mestre del Pino
KCD NL Organizer
Cloud Solution Architect

Platform Real State: Abstract Your Organization’s Tenancy Model Away with Crossplane

What is a platform?

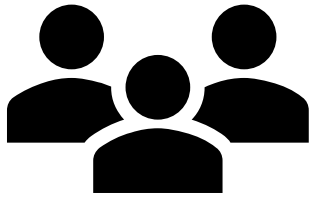
A platform is

The portal / API we use to deploy our code as containers in a managed infrastructure.
Some also have some tools to make development easier



Developers

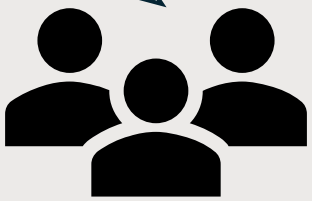
The infrastructure and the tools we use to efficiently manage it, which more often than not it's based on Kubernetes



Platform team

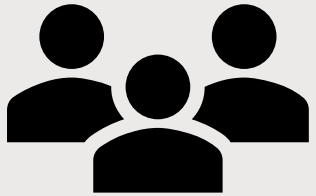
What are you building on it?

Building applications with containers



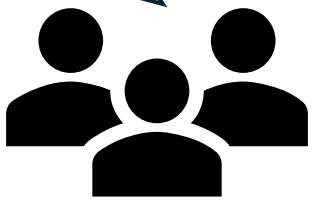
Beginners

Highly scalable, cloud native applications



Intermediate

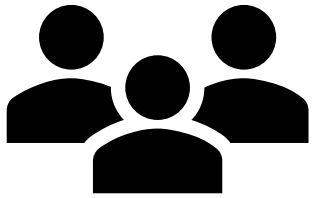
Extending the Kubernetes API



Advanced

What are you building on it?

I'm on a never-ending journey to improving the way we manage things, as well as providing more and more services to developers which simplifies or abstracts infrastructure away for them



Platform Team

Main goals of the platform team

- Minimize infra spend
- Reduce toil (for everyone)
- Provide self-service tools to developer teams

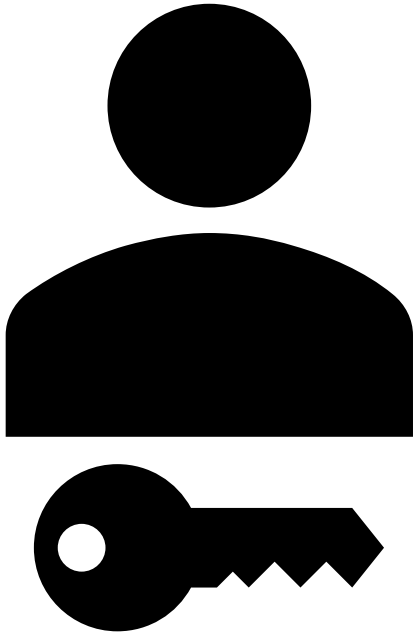
Platform team
Challenges

Growing set of tools and disaggregated platform components

Which are running anywhere and everywhere

Glueing components from different backends with IaC and code with pipelines

Tenancy

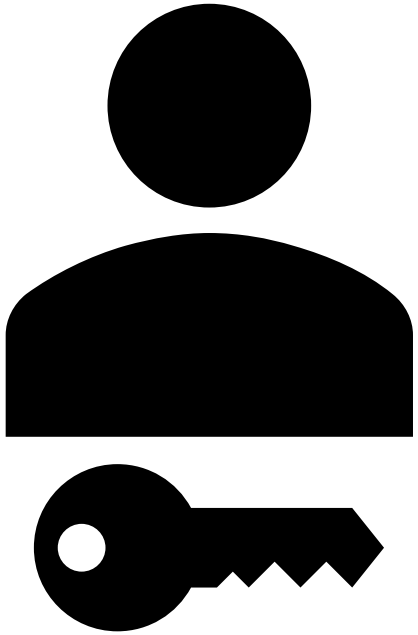


What is a Tenant?

- A person who occupies land or property rented from a landlord

In the context of software engineering:

- A tenant is **a group of users** who share a **common access** with **specific privileges** to the **software instance**.
- It is also commonly used as the highest level of resource hierarchy when it comes to the grouping of resources



What are we trying to achieve?

- Save on infra costs by sharing resources across tenants
- Provide infra and services to our tenants
- Minimize management effort

What are we trying to prevent?

- One tenant’s workload from running on the other tenant’s environment / infra
- Waste of resources
- Noisy neighbors on shared resources

Crossplane

- K8s-Native Infrastructure as Code
- Manage non-k8s (and k8s) resources as Kubernetes CRDs
- Create CRDs without writing operators / controllers

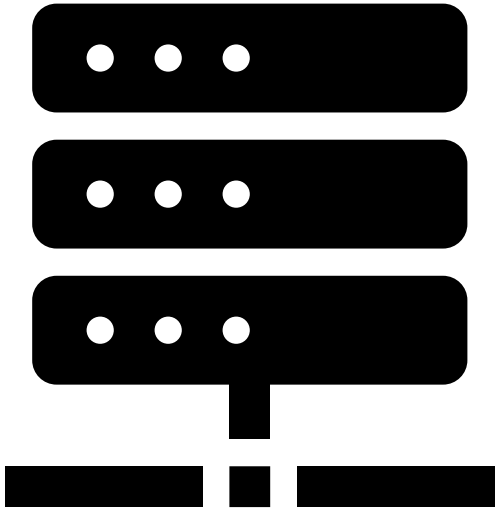
Questions you should ask yourself

- What infra and app services do my tenants need? Which ones do I need to manage everything?
- What scale will my tenants be running? Is there big difference between large and small?
- What kind of isolation is needed? At what levels?
- How much K8s do I want/need to expose to the users?
- How can I make this as efficient as possible while fulfilling my tenant’s requirements?

Scenario

What infra and app services do my tenants need?

Which ones do I need to manage everything?



Tenant Needs

- A way to expose their applications
- Automated life-cycle of DNS records
- A way to import their secrets
- Auth services for their apps
- GitOps based deployments

Platform team needs

- Create resources outside of Kubernetes
- Automated life-cycle of DNS records
- Provide in-cluster network isolation
- Manage and enforce network and rbac policies
- Manage my environments with GitOps

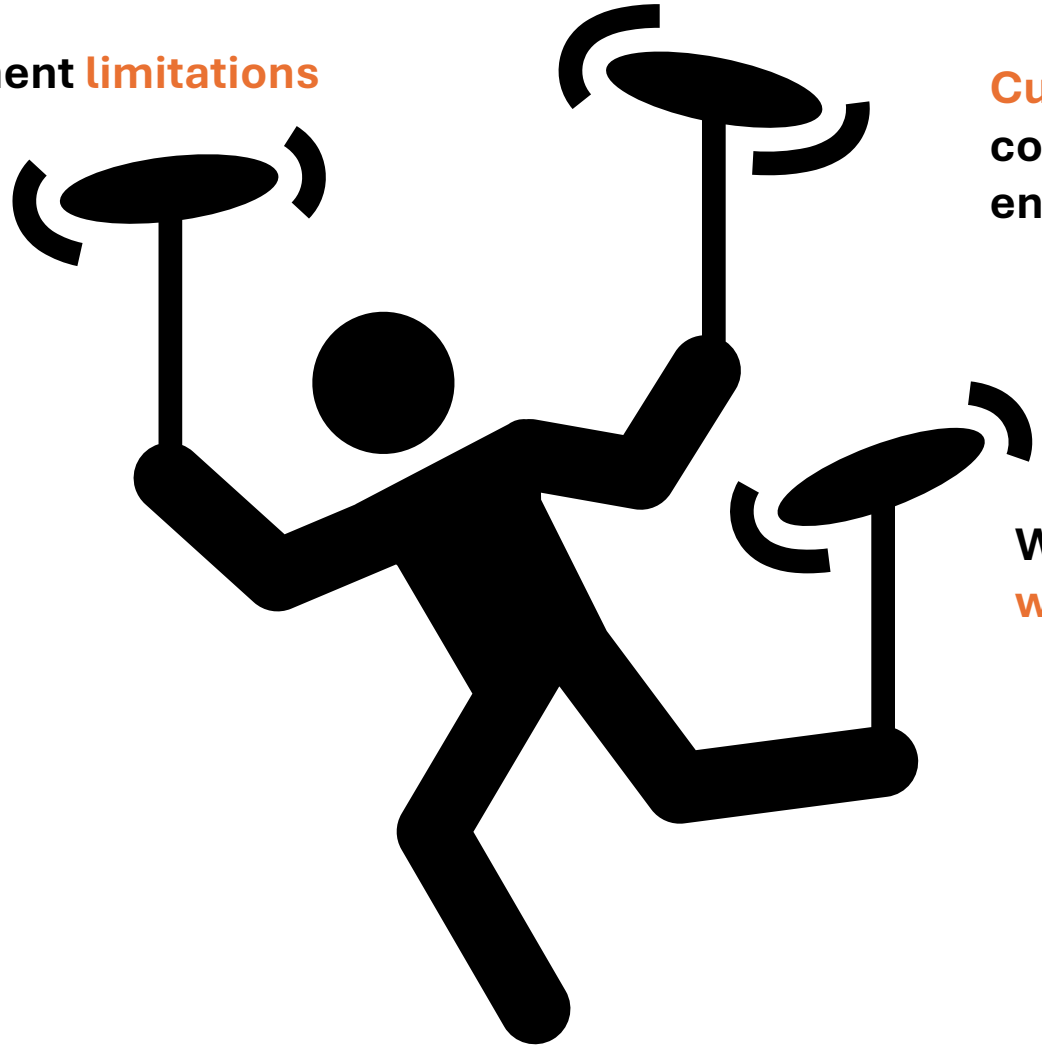

Platform Team's
capacity and knowledge
What developer
teams want/need
Perfectly balanced...
...As all things should be

Governance

Component limitations

Customer's policies about consuming multi-tenant environments

Platform Team's capacity and knowledge

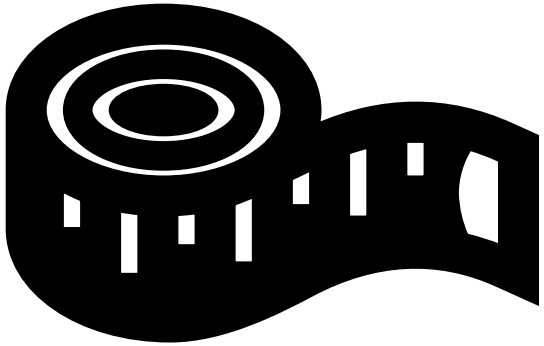


Complexity

What developer teams want/need

What scale will my tenants be running?

Is there big difference between large and small?



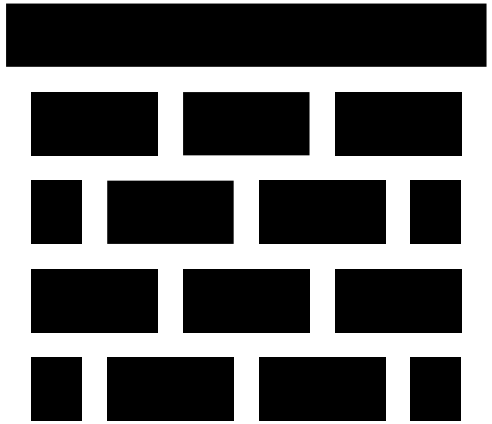
Size considerations

Component limitations

Large tenants can also use small envs

Has a direct impact on dedicated/shared

What kind of isolation is needed?



Isolation levels

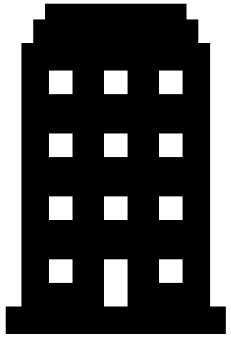
- Compute (shared vs dedicated nodes)
- Network (in and out of cluster)
- Control plane API
- Middleware, tools and app services

Similar to housing

More abstraction

Villa

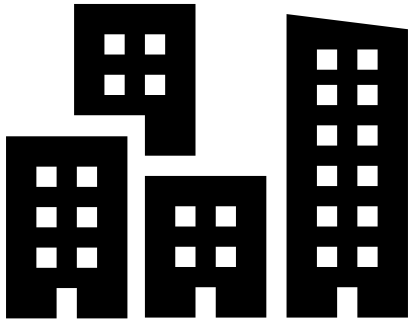
Hotel



Dedicated

Shared

Apartments



Single House

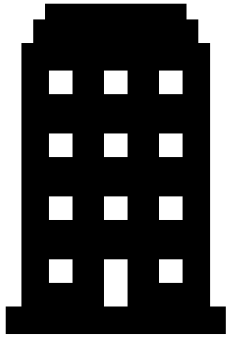
Less abstraction

Similar to housing

More services

Villa

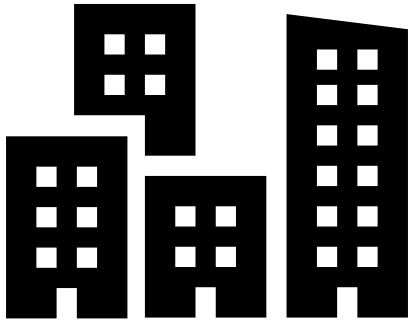
Hotel



Dedicated

Shared

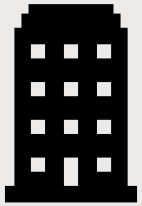
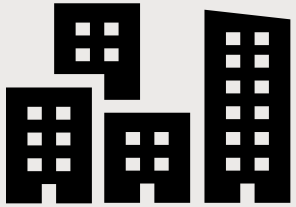
Apartments



Single House

Less services

Now on K8s

				
	Shared cluster	Shared multi-cluster	Single Cluster	Tenant gets multiple clusters
Isolation	Namespace with network policies	Namespaces with network policies	K8s API Compute	K8s API Compute
Disadvantages	Cluster-scoped resources. Sharing K8s API	Complexity	Resource waste	Resource waste Expensive Complexity
Providing services (to tenant)	Allow traffic from system namespaces through Network Policies	Cilium ClusterMesh Global services	Outside of k8s network Cilium External Workloads (beta)	Outside of k8s network Cilium External Workloads (beta)
Providing services (within tenant)	Within / Across namespaces	Within / Across namespaces	Within / Across namespaces	Cilium ClusterMesh Global services

We are not constrained by the physical world

…

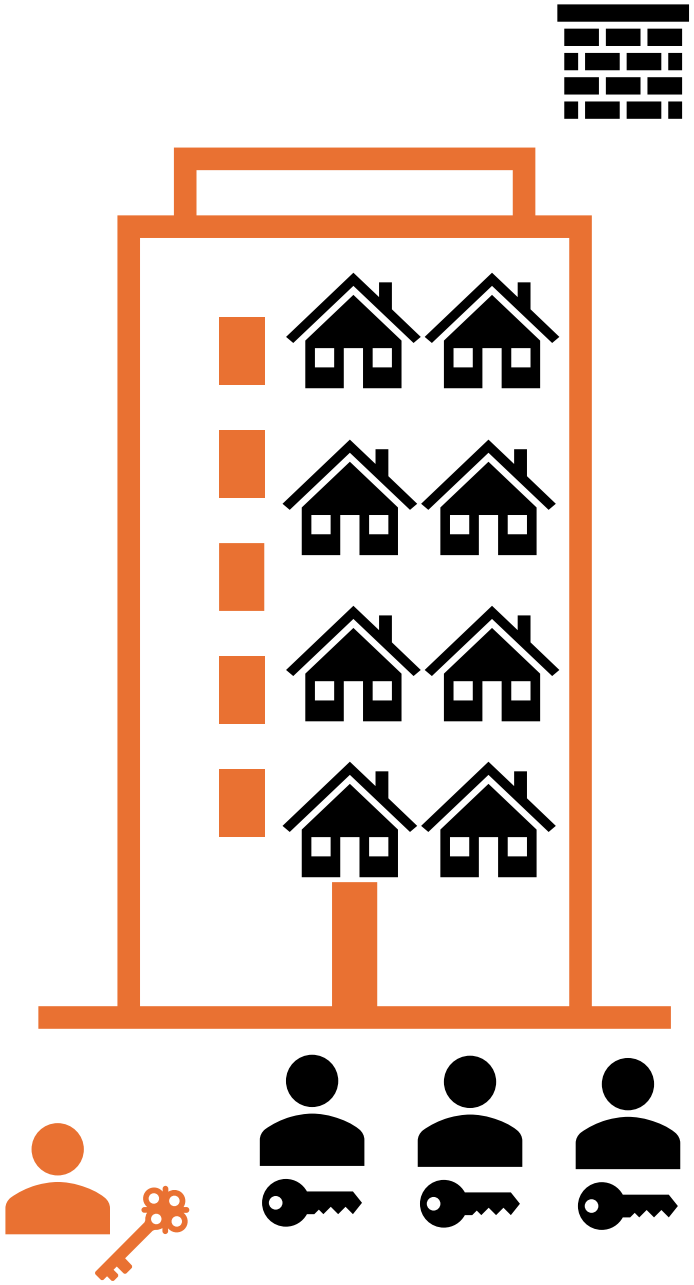
in the same way

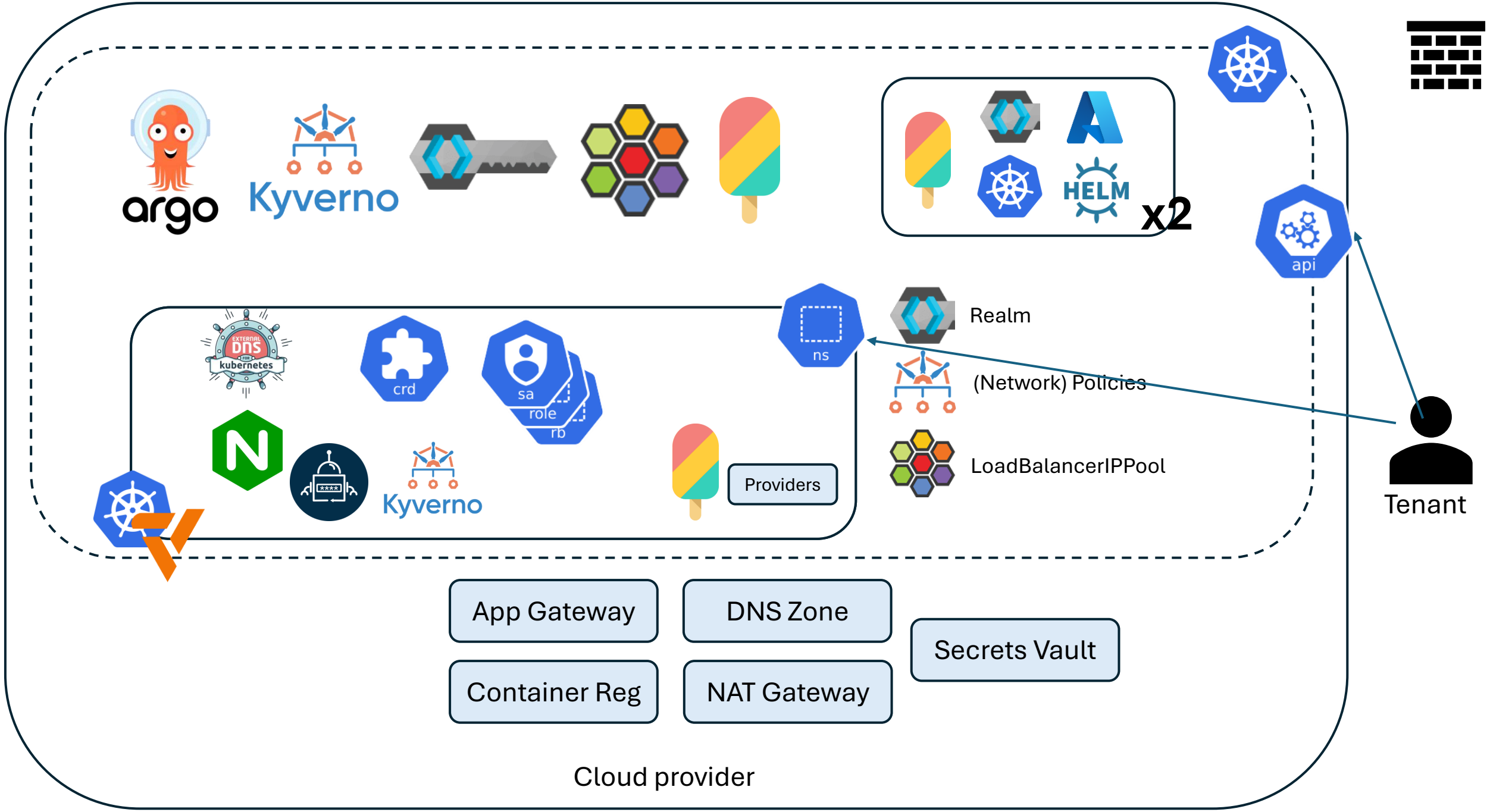
Shared cluster – Isolated k8s API

- vCluster allows you to deploy a virtual Kubernetes cluster which lives in a namespace of the host cluster
- You deploy services in the host cluster's namespaces
- Can provide compute isolation by binding nodes / nodepools to namespaces

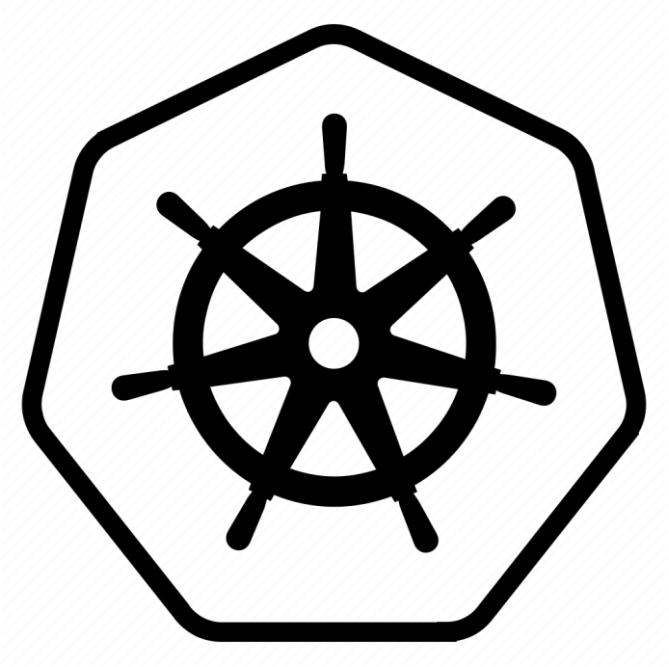
Providing services:

- Platform team to tenants:
  - vCluster can expose services from host to guest
  - Cilium External Workloads (beta)
- Within tenant:
  - Across namespaces in the virtual cluster




argo
Kyverno
HELM
x2
api
ns
Realm
(Network) Policies
LoadBalancerIPPool
crd
sa
role
rb
Kyverno
Providers
Tenant
App Gateway
DNS Zone
Secrets Vault
Container Reg
NAT Gateway
Cloud provider

How much K8s do I want to expose to the users?



Crossplane

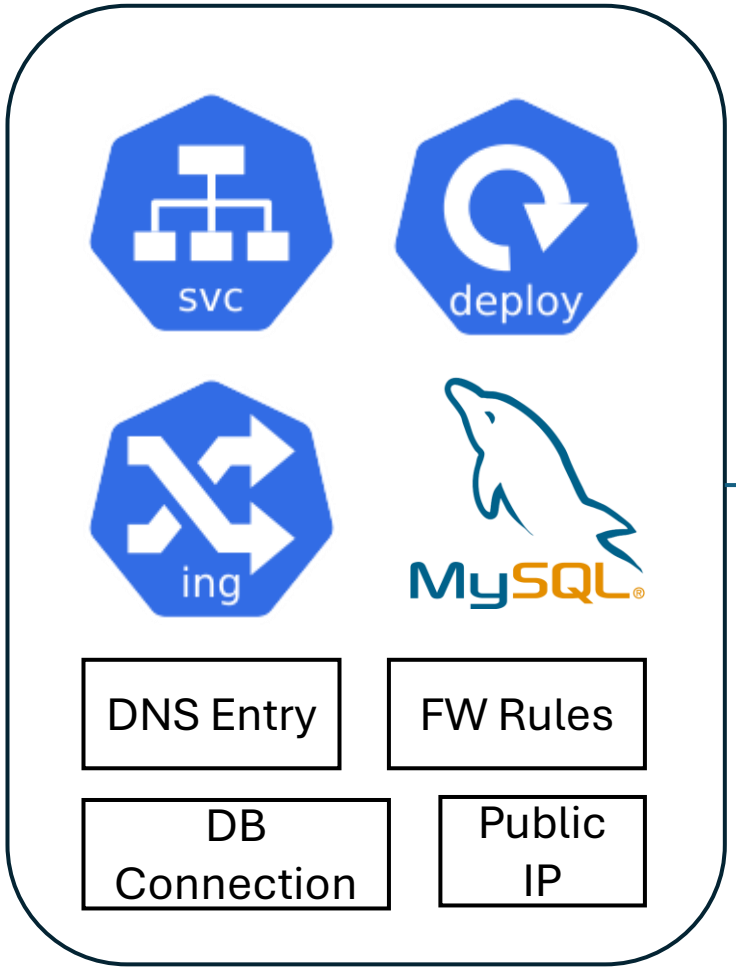
Why Crossplane?

- No need to create and orchestrate pipelines with retry policies, resource cleanup, race conditions etc…
- Enables GitOps on non-k8s resources
- Unifying LCM of resources which run in multiple backends

How much K8s to expose?

- Do my tenants need access to the K8s API?
- Do I need to abstract things away for them?
- What kind of abstractions do they need?

Crossplane Compositions – in short

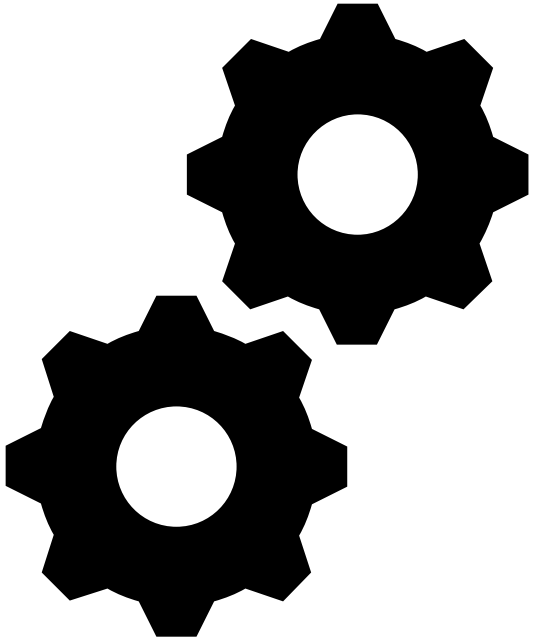


console

```
$ cat devteam1-webapp.yaml

apiVersion: platform.myorg.com/v1
kind: WebApp
metadata:
  name: example-webapp
  namespace: devteam1
spec:
  containers:
    - name: webapp-frontend
      image: cr1.example.com/frontend:2.1
      replicas: 2
    - name: webapp-backend
      image: cr1.example.com/backend:2.3
      replicas: 3
  database:
    size: 100gb
```

How can I make this as efficient as possible while fulfilling my tenant's requirements?



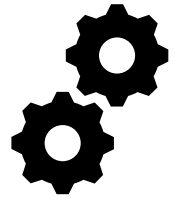
Steps

Gather requirements for needed services, sizing, isolation, desired abstractions and k8s exposure

Categorize platform components

Define a data model

Categorising Services



AppProject

Realm

ClusterSecretStore
SecretStore

App Gateway

DNS Zone

Secrets Vault

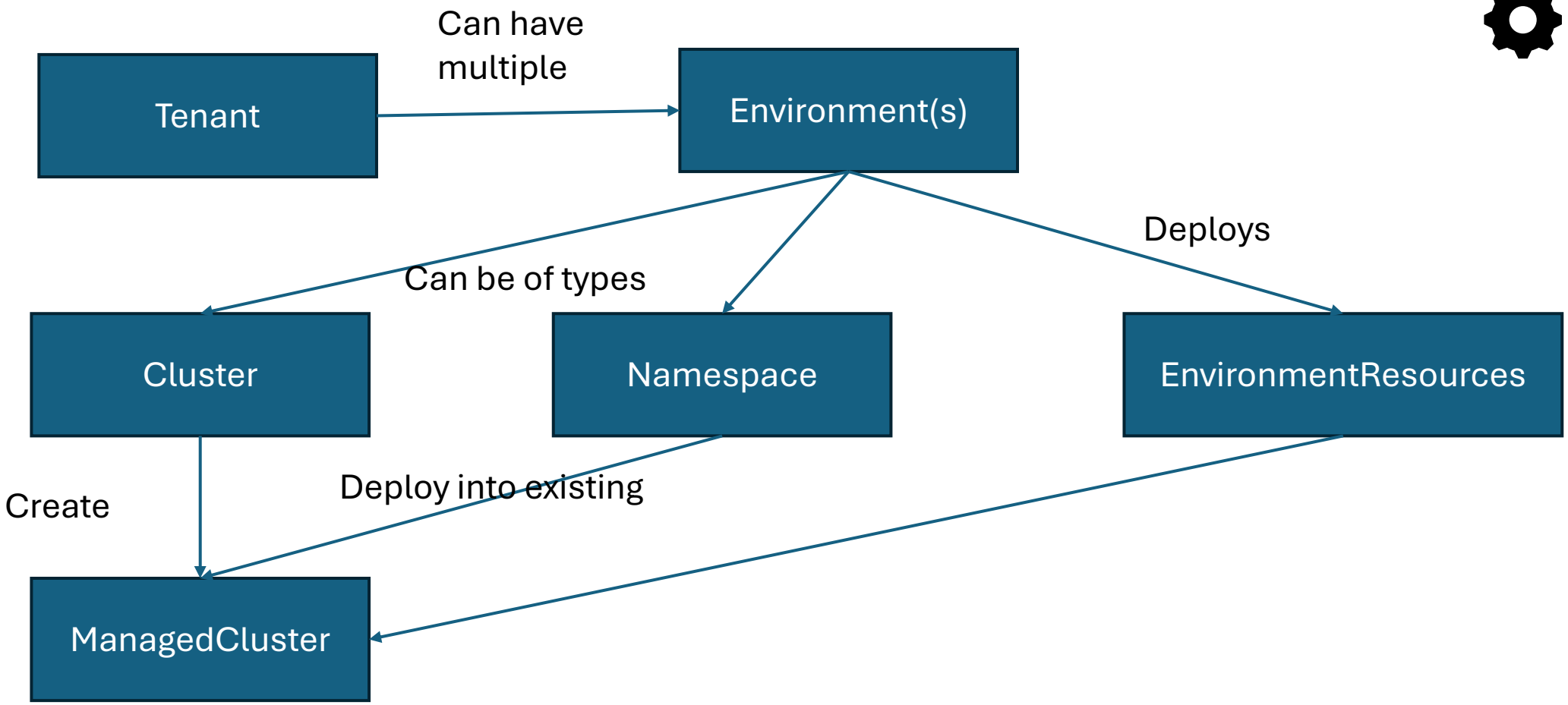
Container Reg

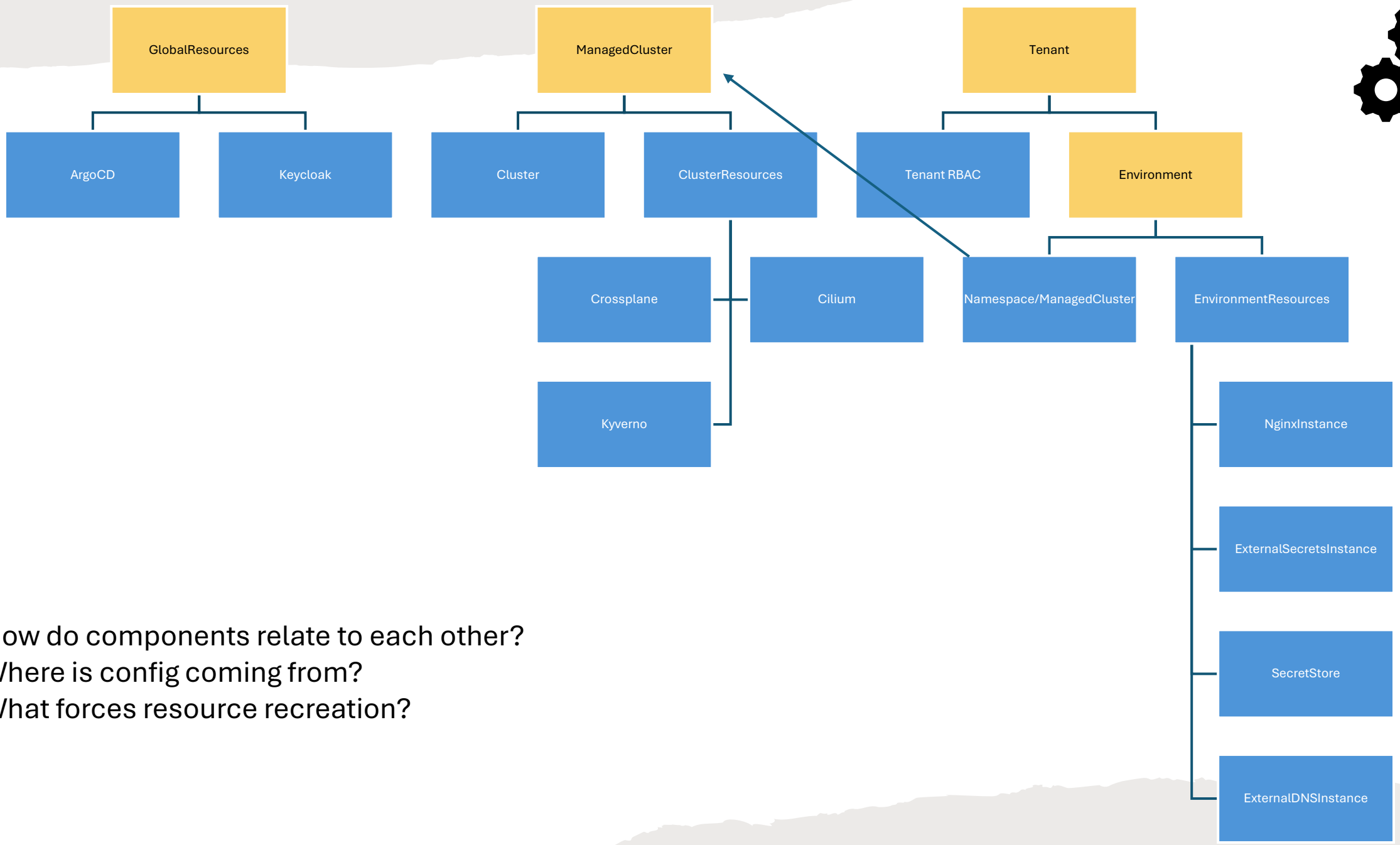
NAT Gateway

Categorising Services

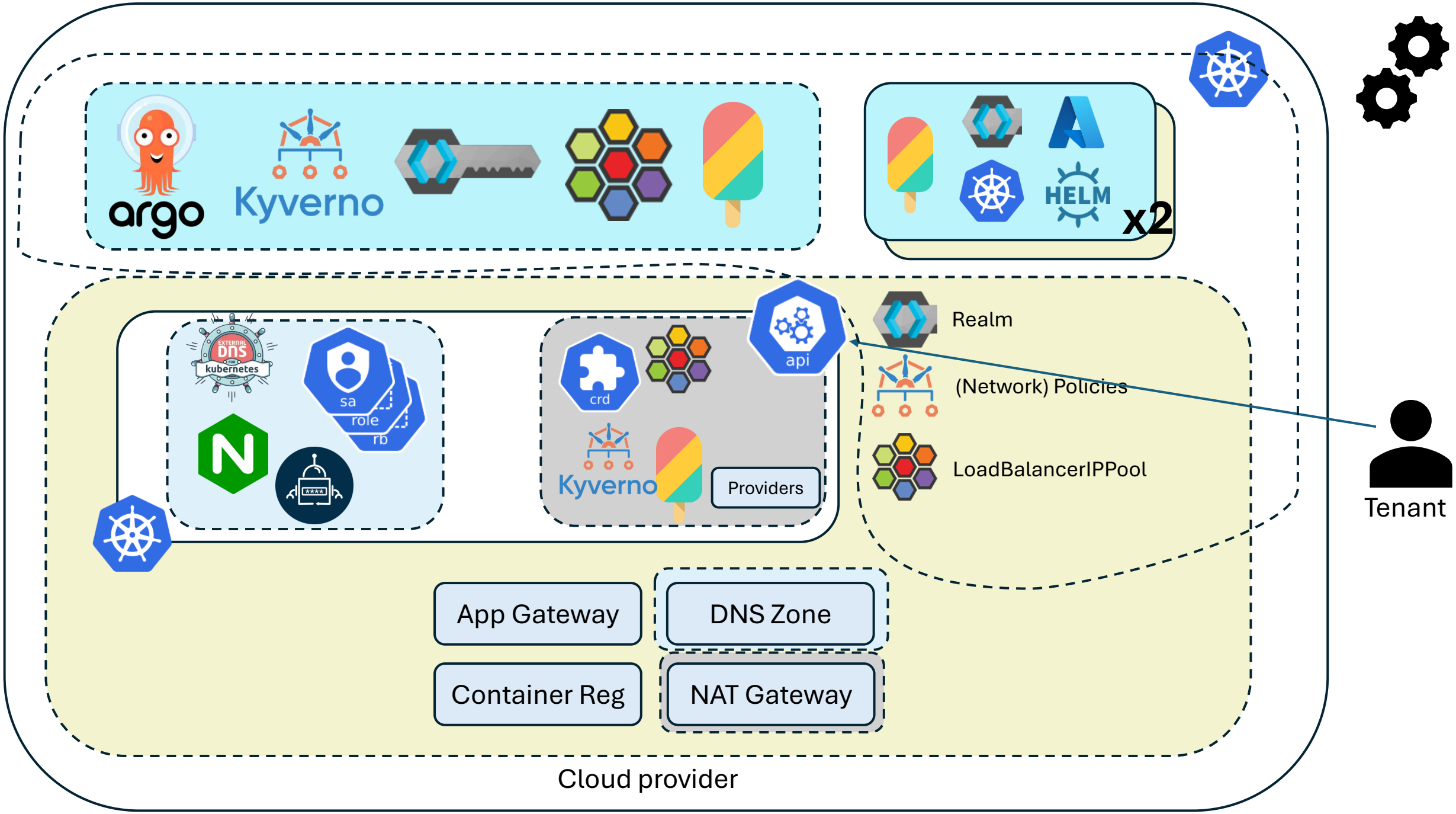
Platform management (global)	Per tenant	Per Cluster (regardless if tenant uses all cluster or ns)	Per Tenant Environment
argo	AppProject Realm	Kyverno	SecretStore
	App Gateway Container Reg	NAT Gateway	Secrets Vault DNS Zone

Defining a Data Model


Tenant
Can have multiple
Environment(s)
Can be of types
Deploys
Cluster
Namespace
EnvironmentResources
Create
Deploy into existing
ManagedCluster



How do components relate to each other?
Where is config coming from?
What forces resource recreation?


argo
Kyverno
HELM
x2
EXTERNAL DNS FOR kubernetes
sa
role
rb
crd
api
Kyverno
Providers
Realm
(Network) Policies
LoadBalancerIPPool
Tenant
App Gateway
DNS Zone
Container Reg
NAT Gateway
Cloud provider

To wrap up

The answers will mostly depend on

- The kind of workloads your customers will run + the way you want to manage your tenants
- The size/type of your organization or customers
- Your org’s and your customer’s strategy, policies and governance requirements
- The customer’s need and proficiency with Kubernetes
- The proficiency, size and range of knowledge of your platform team

Key Takeaways

- Understand what tools are needed to do the job (yours and your customers’). Optimize infra
- Understand the limiting factors each of your stack’s component
- Understand the factors which force isolation at all levels
- Standardize delivery of services and solutions. Build abstractions with Crossplane when needed
- Leverage the orchestrator you already have. Plan and build your platform’s API

Platform Real State: Abstract Your Organization’s Tenancy Model Away with Crossplane

@mestredelpino