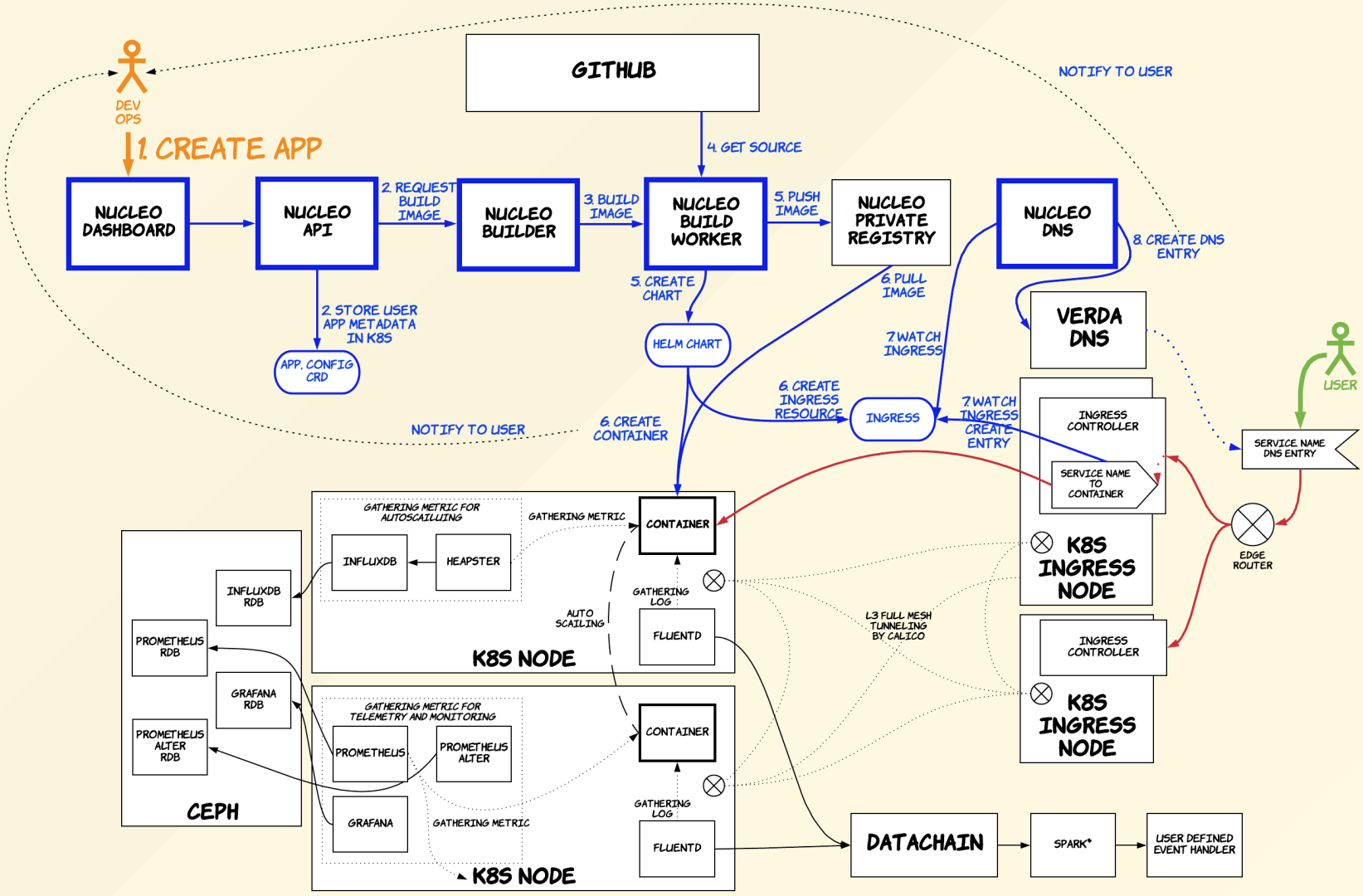
How to make cloud native platform by kubernetes

Eohyung Lee

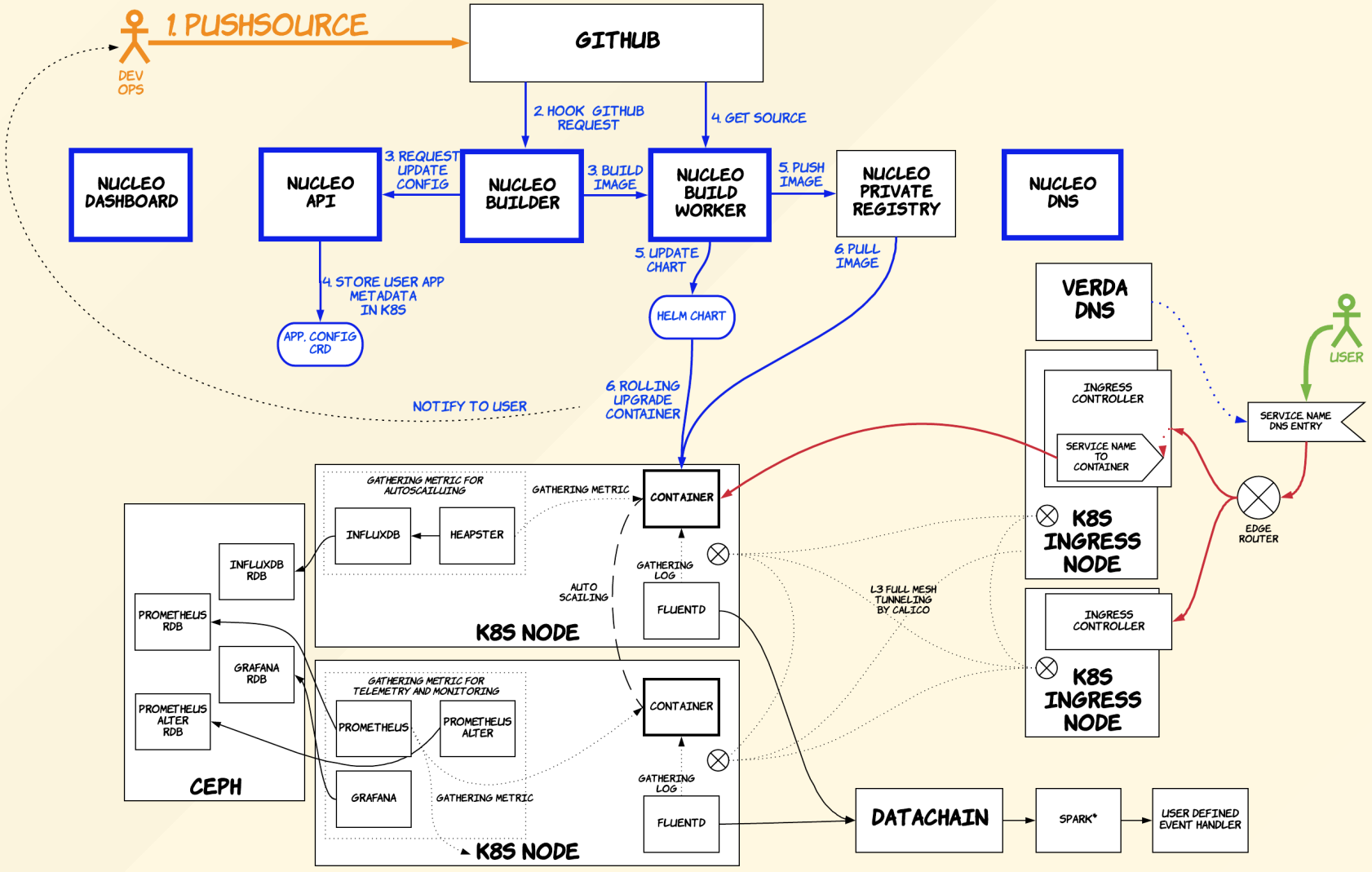
al.lee@linecorp.com

liquidnuker@gmail.com

Nucleo architecture (create app)



Nucleo architecture (update app)



cloud native platform nucleo

Kubernetes

https://kubernetes.io/docs/concepts/

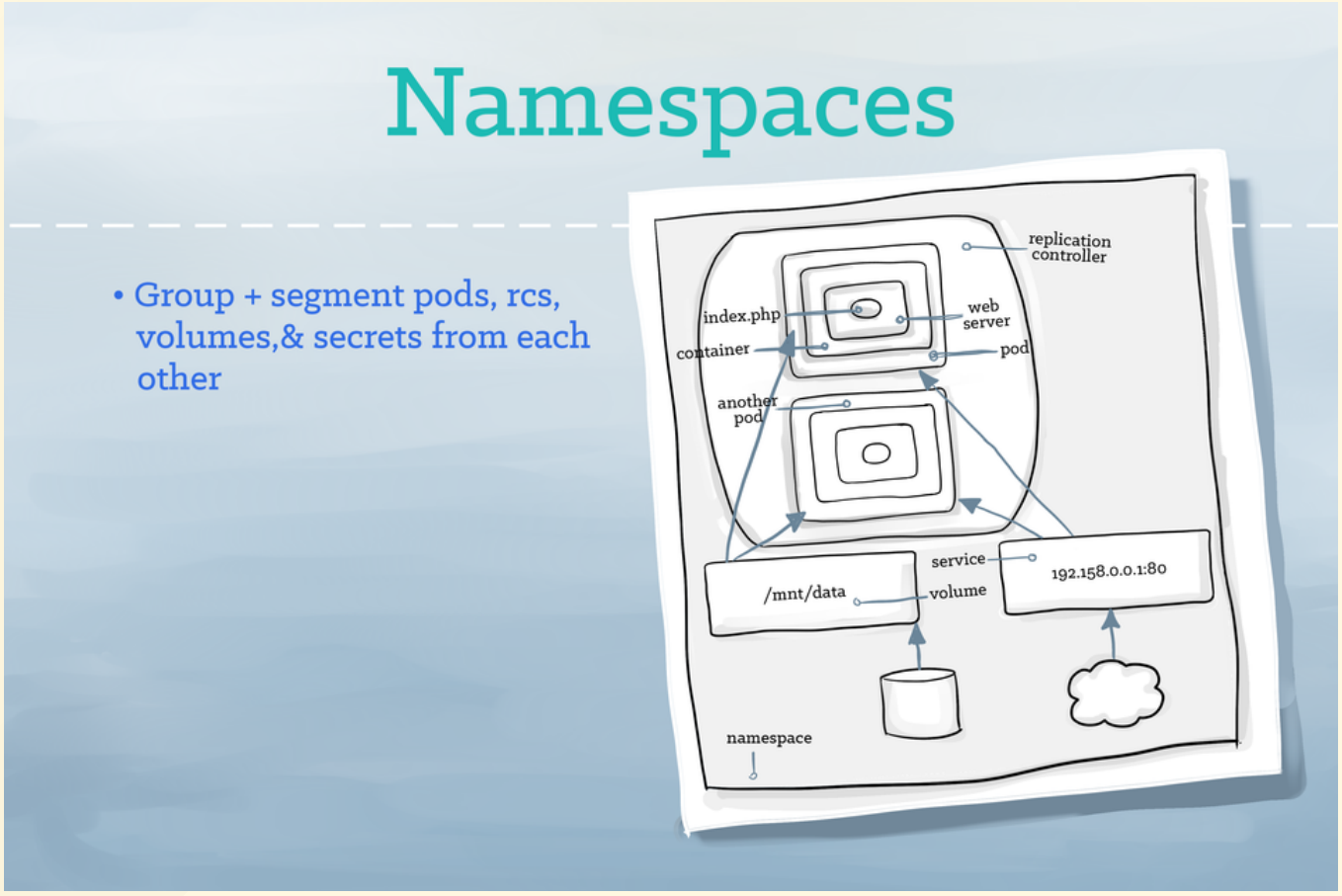
Kubernetes has a number of features. It can be thought of as:

- a container platform
- a microservices platform
- a portable cloud platform and a lot more.

Kubernetes a.k.a. K8S

- k + len(ubernetes) + s = k + 8 + s = k8s
- provides a `container-centric management` environment.
- orchestrates `computing, networking, and storage infrastructure` on behalf of user workloads.

Traditional explanation of k8s



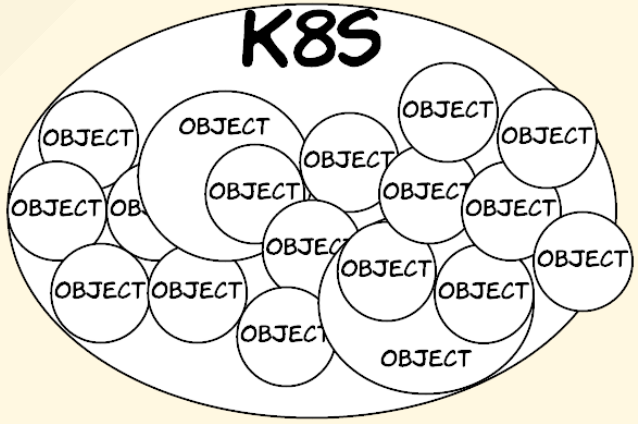
The Illustrated Children's Guide to Kubernetes
https://youtu.be/4ht22ReBjno

So, we focused to talk about

- how to use k8s as a `cloud native backend platfom`.
  - how to use k8s `by api`
  - How to manage `user app`
  - How to manage `user app metadata`
  - How to support `accessing to public`
  - How to support `continuous delivery`

What constitutes a k8s

- k8s `object` (resource object, `resource`)
  - are `persistent entities` in the Kubernetes system.
  - Kubernetes uses these entities to represent `the state of your cluster.`



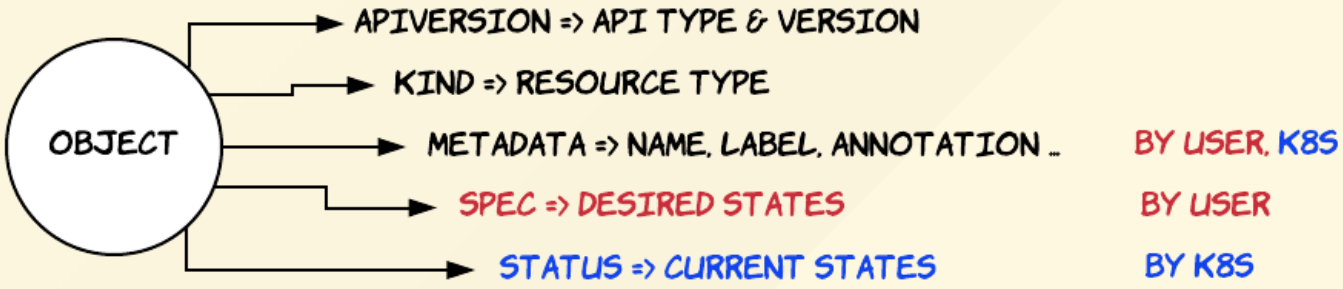
K8S object types

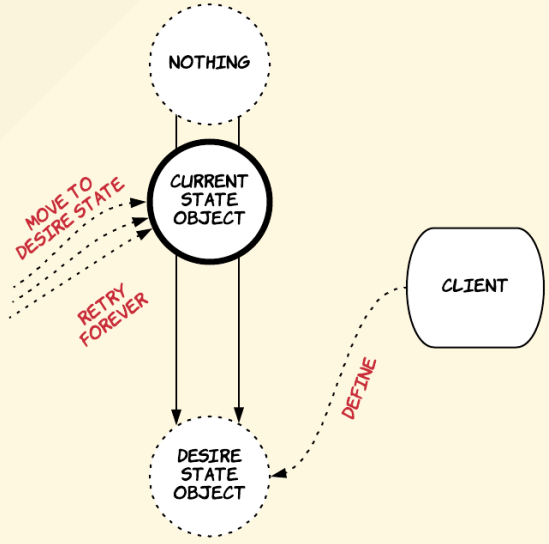
- Workloads
  - container, pods, deployment, daemonset, ...
- Discovery & LB resources
  - endpoint, ingress, service, ...
- Config & Storage resources
  - configmap, secret, persistentvolumeclaim, ...

K8S object types (cont.)

- Cluster resources
  - node, namespace, role, serviceaccount, ...
- Metadata resources
  - event, horizontalpodautoscaler, customresourcedefinition, ...

K8S object components





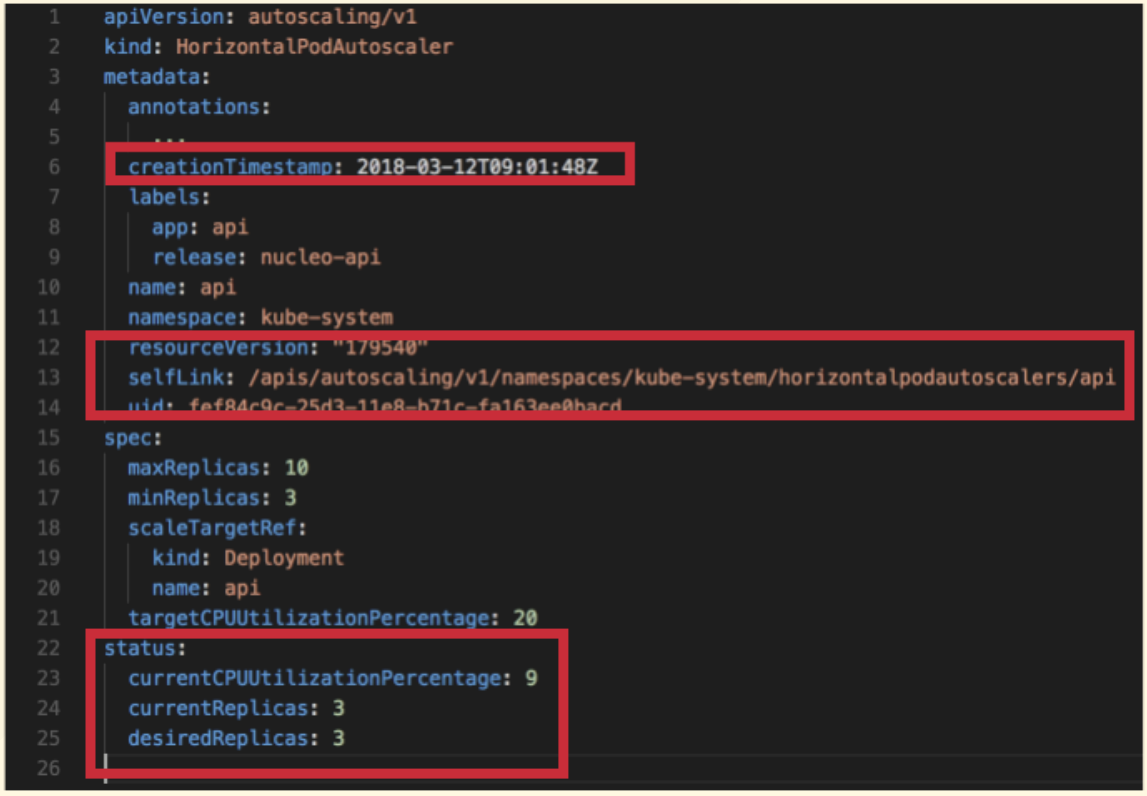
K8S object spec

- before apply

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  labels:
    app: api
    release: nucleo-api
  name: api
  namespace: kube-system
spec:
  maxReplicas: 10
  minReplicas: 3
  scaleTargetRef:
    kind: Deployment
    name: api
  targetCPUUtilizationPercentage: 20
```

K8S object spec & status

- after apply



```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  annotations:
    ...
  creationTimestamp: 2018-03-12T09:01:48Z
  labels:
    app: api
    release: nucleo-api
  name: api
  namespace: kube-system
  resourceVersion: "179540"
  selfLink: /apis/autoscaling/v1/namespaces/kube-system/horizontalpodautoscalers/api
  uid: fef84c9c-25d3-11e8-b71c-fa163ee0bacd
spec:
  maxReplicas: 10
  minReplicas: 3
  scaleTargetRef:
    kind: Deployment
    name: api
  targetCPUUtilizationPercentage: 20
status:
  currentCPUUtilizationPercentage: 9
  currentReplicas: 3
  desiredReplicas: 3
```

Transaction vs spec & status method

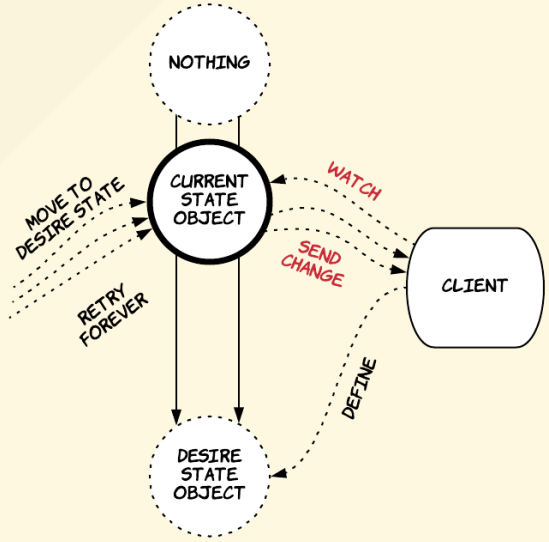
- for supporting micro service architecture
  - infinitely retriable
  - self healing
  - ...

K8S object operations

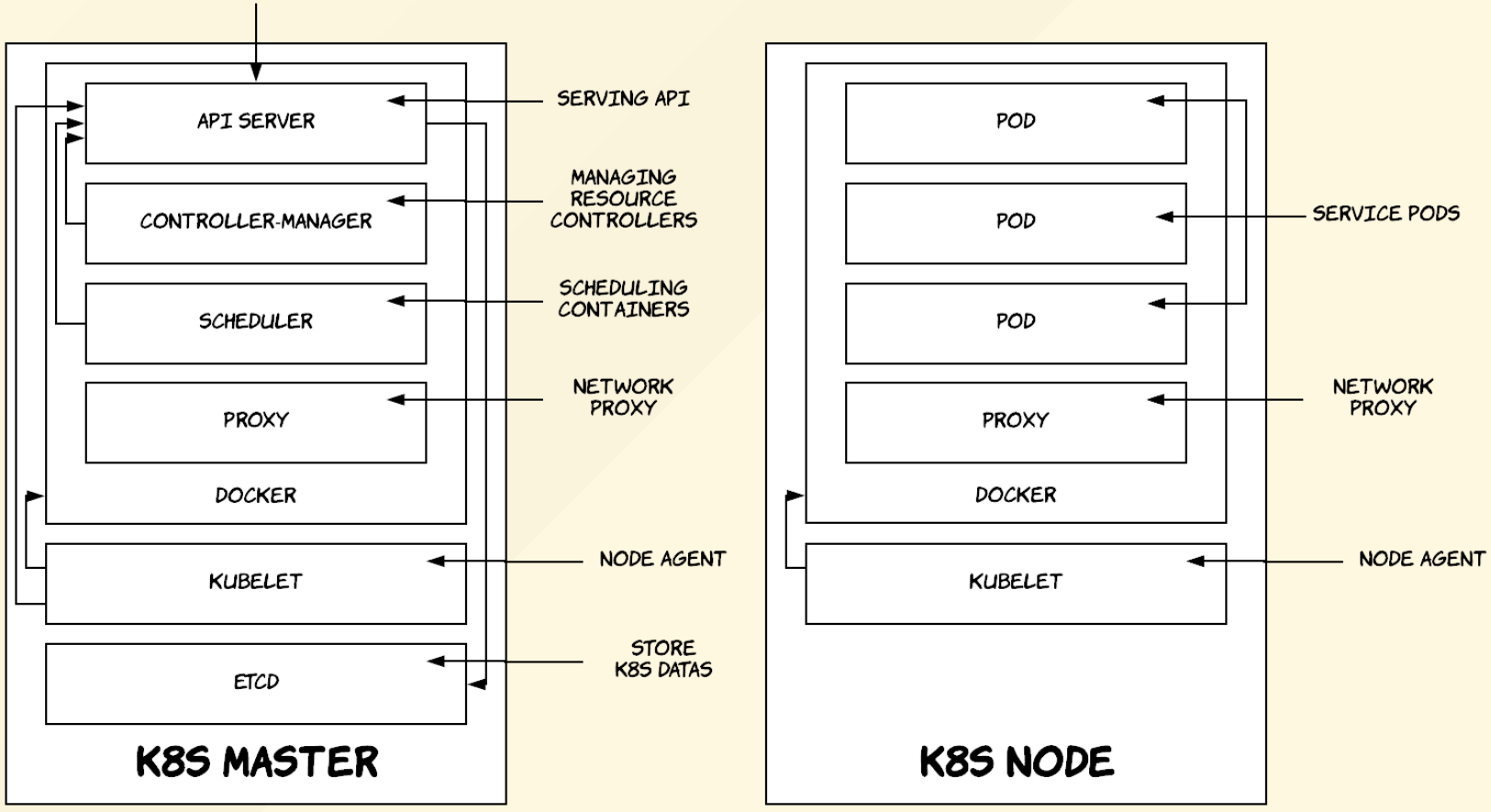
- Create
- Update
  - Replace, Patch
- Read
  - Get, List, `Watch`
- Delete

K8S watch api

- Watch will `stream results` for an object(s) as it is updated.
- Similar to a callback, watch is used to respond to `resource changes`.

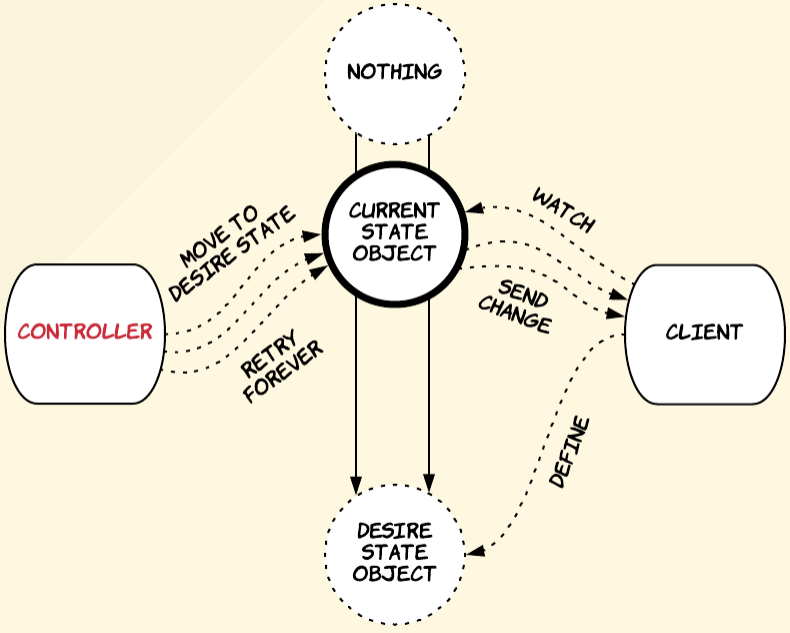


K8S architecture

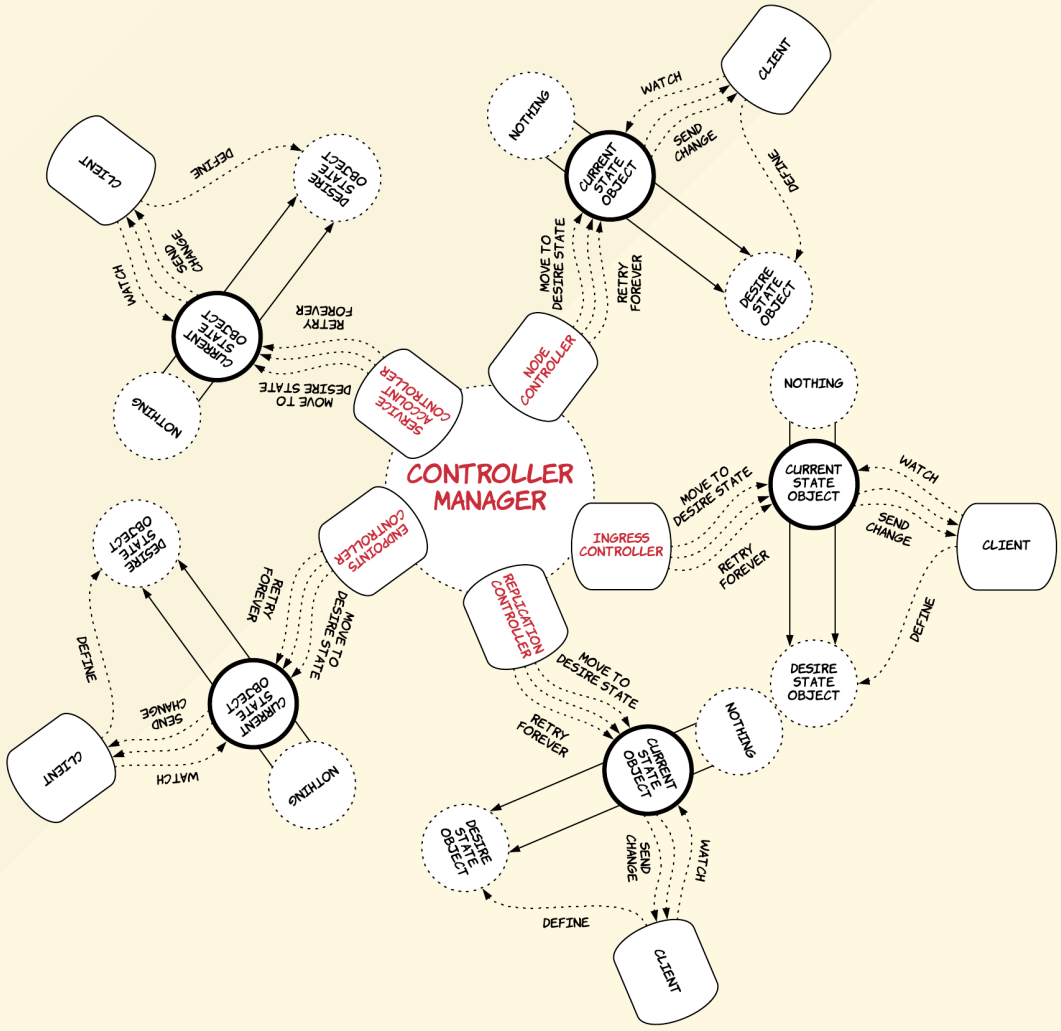


K8S controller

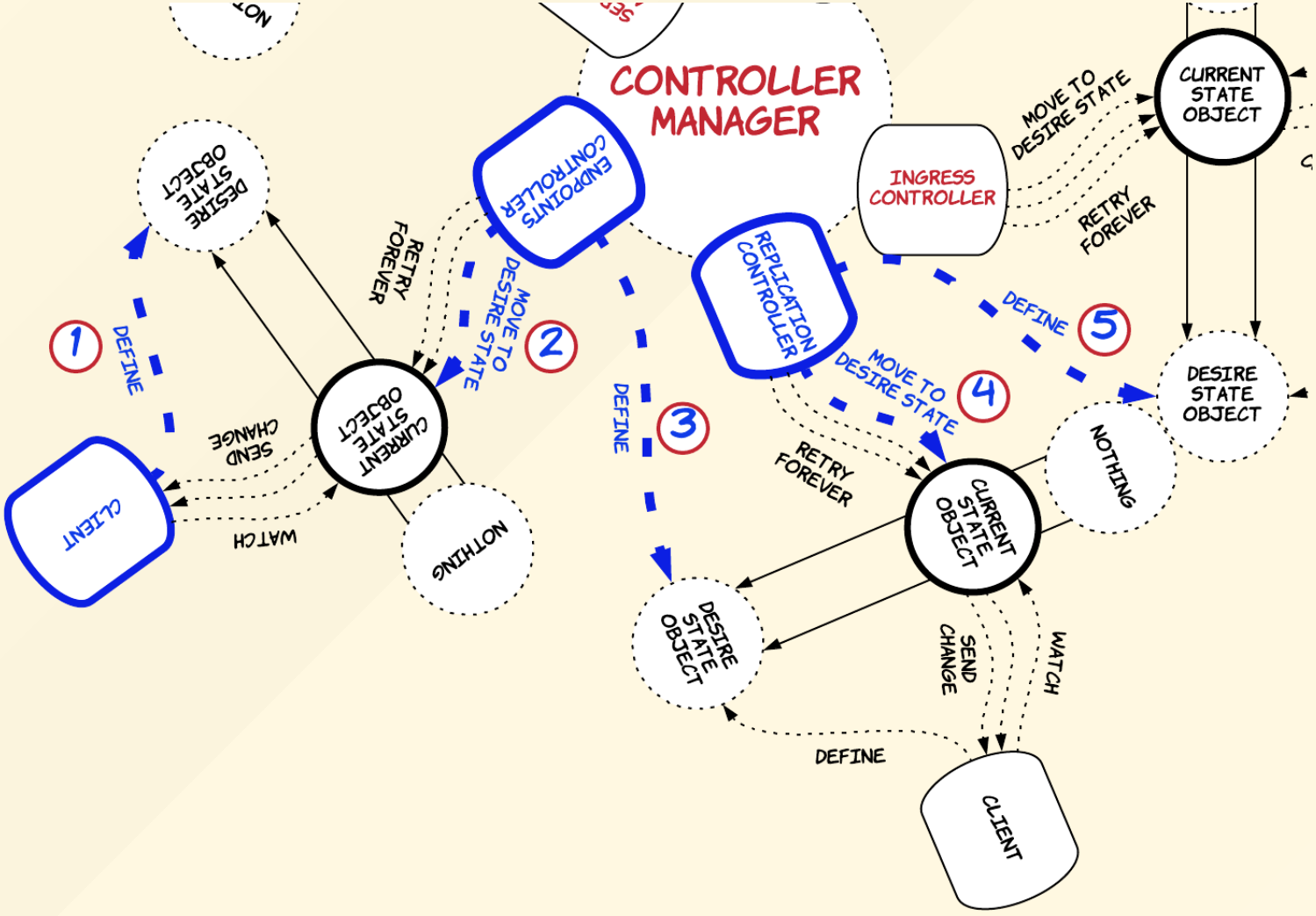
- a controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the `current state` towards the `desired state`



K8S controller manager



For example, chaining events

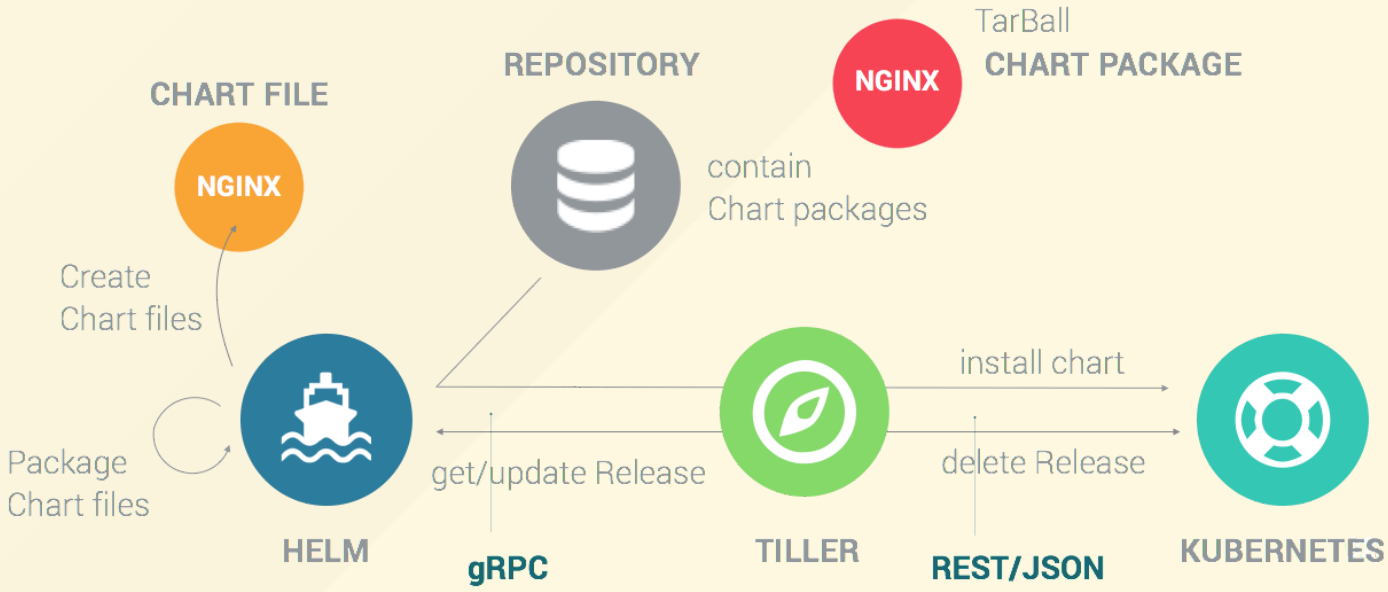


So, k8s API

- remember `spec & status`
- remember `watch`

How to manage user app

- helm: The package manager for Kubernetes



- https://qiita.com/Ladicle/items/63cad824e27aa8aac7e1

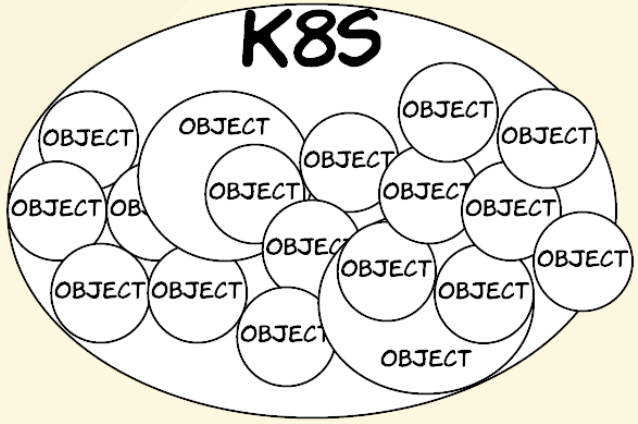
Nucleo chart

- chart: Helm uses a packaging format called charts

```
charts/nucleo/
├── .helmignore # helm ignore file
├── Chart.yaml # chart spec
├── README.md
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl # help function, define values
│   ├── config.yaml # config for user apps
│   ├── deployment.yaml # container deploy method, config
│   ├── hpa.yaml # autoscaling features
│   ├── ingress.yaml # access to outside L7(like nginx)
│   ├── secret.yaml # secret config for user apps
│   └── service.yaml # service discovery, endpoint, L4
└── values.yaml # chart configuration value
```

How to manage user app metadata

- e.g. application information, release histories, build histories ...
- Custom Resource Definition(CRD)
  - `store metadata in k8s` like other objects



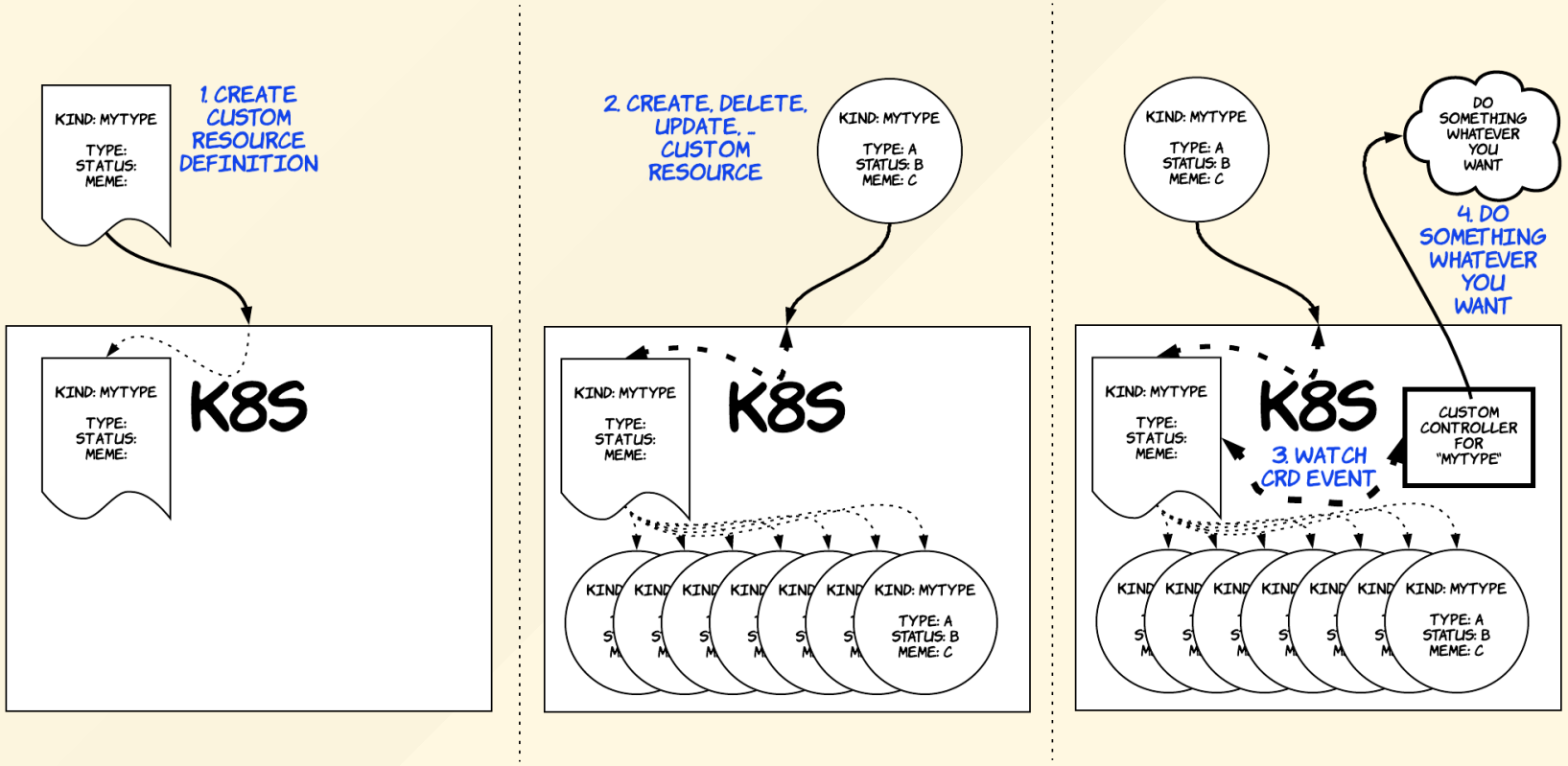
Custom resources

- make your own resource dynamically
- and use k8s database, API, authentication like the other `resource objects`
- support CRUD functions
  - e.g. `Create`, `Update`, `Delete`, `Read`

Custom resources definition

```yaml
apiVersion: nucleo.linecorp.com/v1
kind: DeployConfig # this resource is made by nucleo
metadata:
  creationTimestamp: 2018-01-31T06:03:13Z
  labels:
    nucleo.app: fernet-decryptor
  name: 0162733d-c8e9-44f5-9b3b-f8d587602b17
  resourceVersion: "37313981"
  selfLink: /apis/nucleo.linecorp.com/v1/link
  uid: 6d14f769-064c-11e8-9b84-fa163e9b48b8
spec:
  branch: ^.
  display_name: default
status:
  build_status: success
  status: running
```

Custom resource & controller



How to support accessing to public

- Use ingress for exposing user app to the world
  - like a `aws application load balancer`
  - ingress resources has no prebuilded controller
    - need to use `NGINX Ingress Controller` install by `helm`
    - can support multiple & various ingress controller in one k8s cluster

Ingress with service

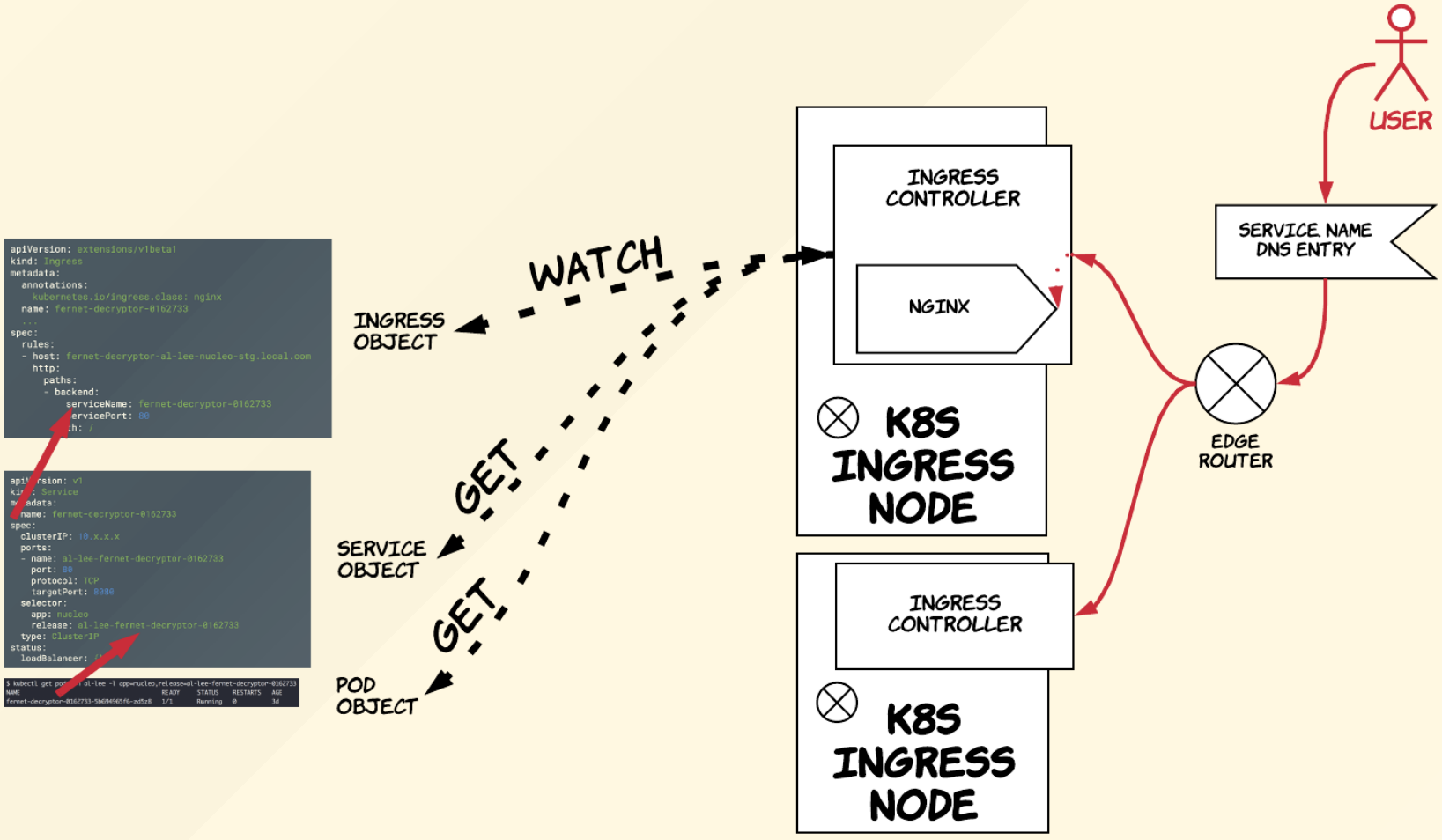
```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
  name: fernet-decryptor-0162733
  ...
spec:
  rules:
  - host: fernet-decryptor-al-lee-nucleo-stg.local.com
    http:
      paths:
      - backend:
          serviceName: fernet-decryptor-0162733
          servicePort: 80
        path: /
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: fernet-decryptor-0162733
spec:
  clusterIP: 10.x.x.x
  ports:
  - name: al-lee-fernet-decryptor-0162733
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: nucleo
    release: al-lee-fernet-decryptor-0162733
  type: ClusterIP
status:
  loadBalancer: {}
```

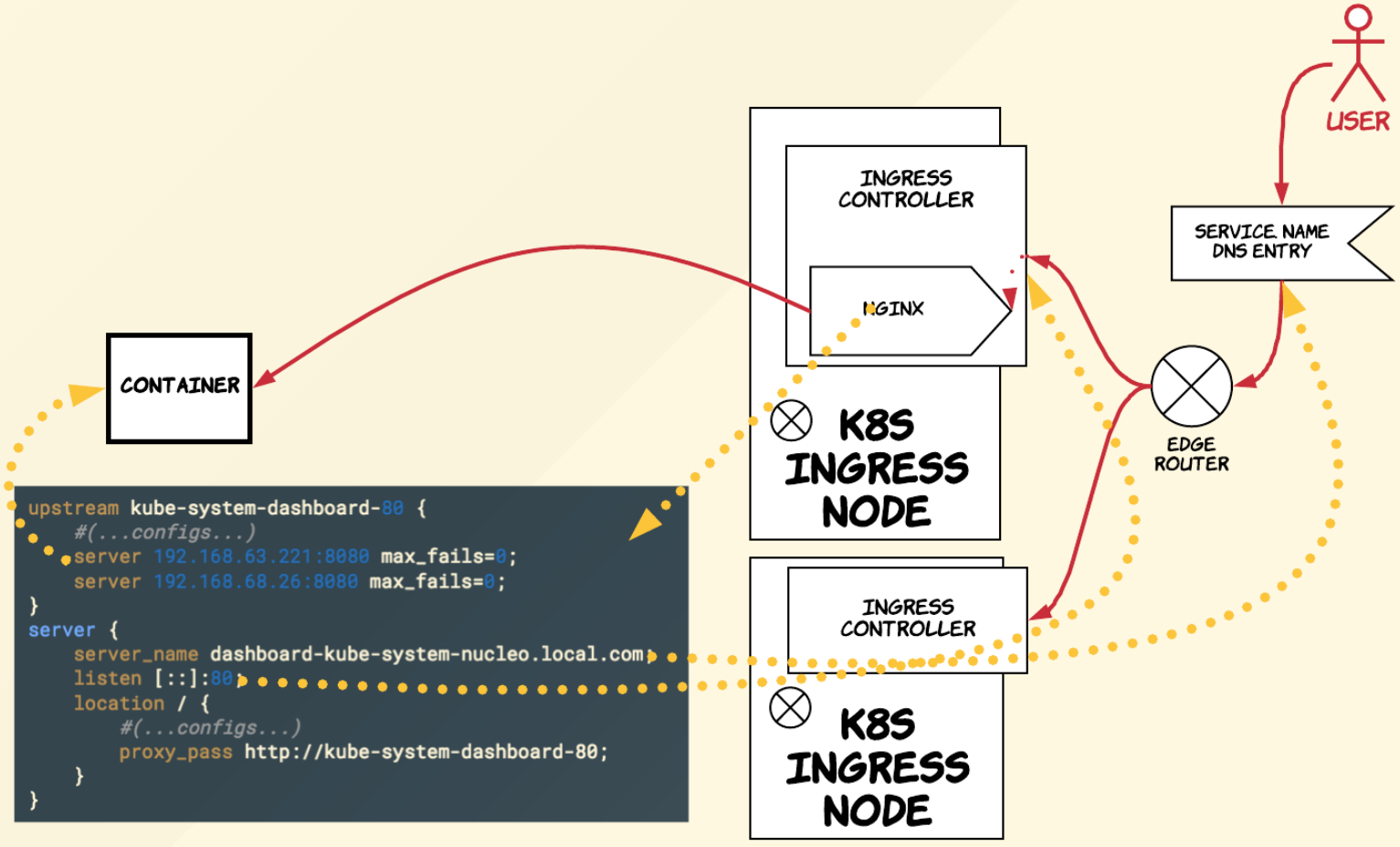
```
$ kubectl get pods -n al-lee -l app=nucleo,release=al-lee-fernet-decryptor-0162733
NAME                                          READY     STATUS    RESTARTS   AGE
fernet-decryptor-0162733-5b694965f6-zd5z8     1/1       Running   0          3d
```

BUT, USE POD IP NOT CLUSTERIP WHICH USE NAT
THIS IS ONLY FOR SERVICE DISCOVERY

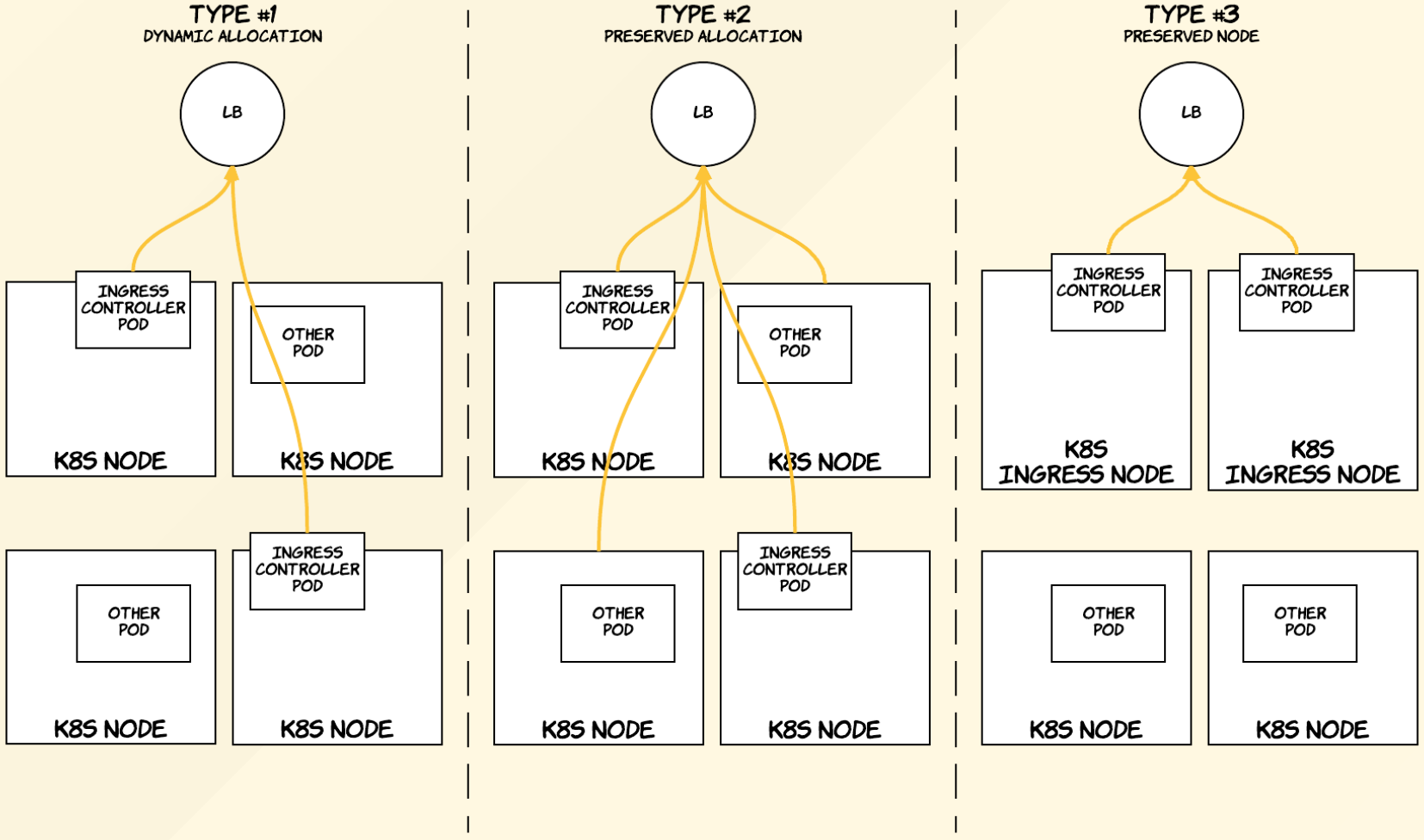
Ingress with controller



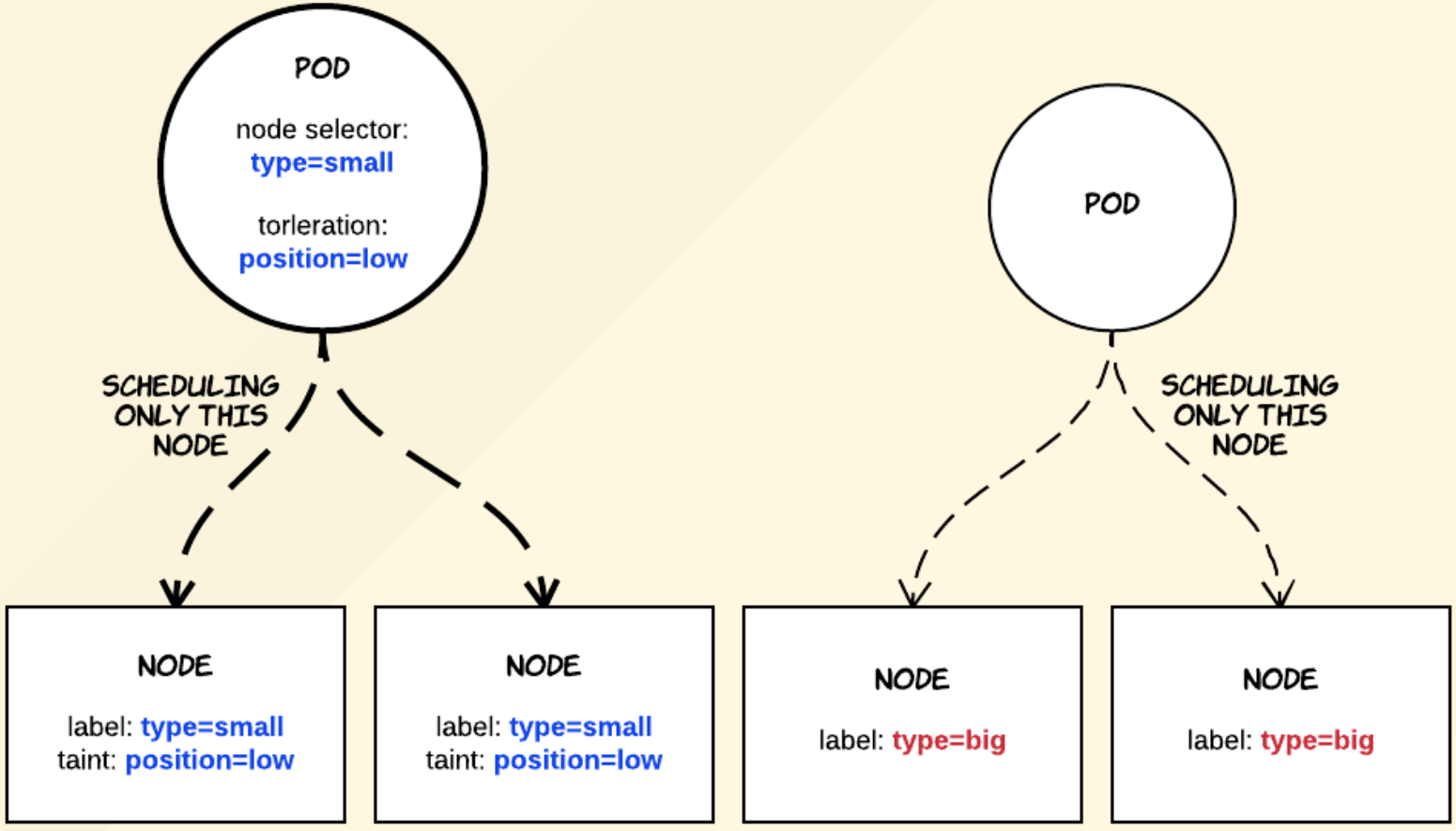
Ingress


USER
SERVICE NAME
DNS ENTRY
INGRESS
CONTROLLER
NGINX
K8S
INGRESS
NODE
EDGE
ROUTER
CONTAINER
INGRESS
CONTROLLER
K8S
INGRESS
NODE
upstream kube-system-dashboard-80 {
#(...configs...)
server 192.168.63.221:8080 max_fails=0;
server 192.168.68.26:8080 max_fails=0;
}
server {
server_name dashboard-kube-system-nucleo.local.com;
listen [::]:80;
location / {
#(...configs...)
proxy_pass http://kube-system-dashboard-80;
}
}

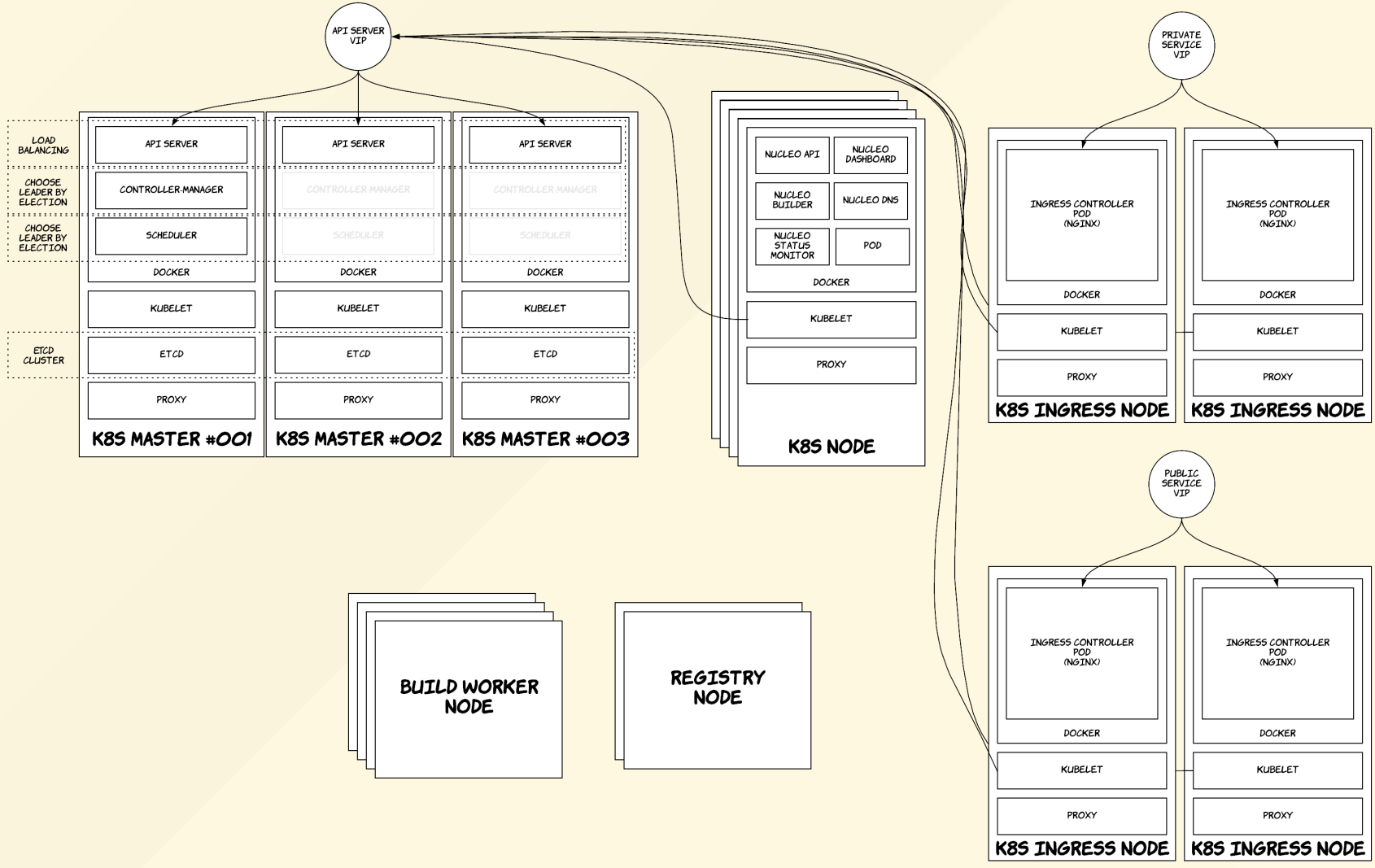
Ingress node deploy types



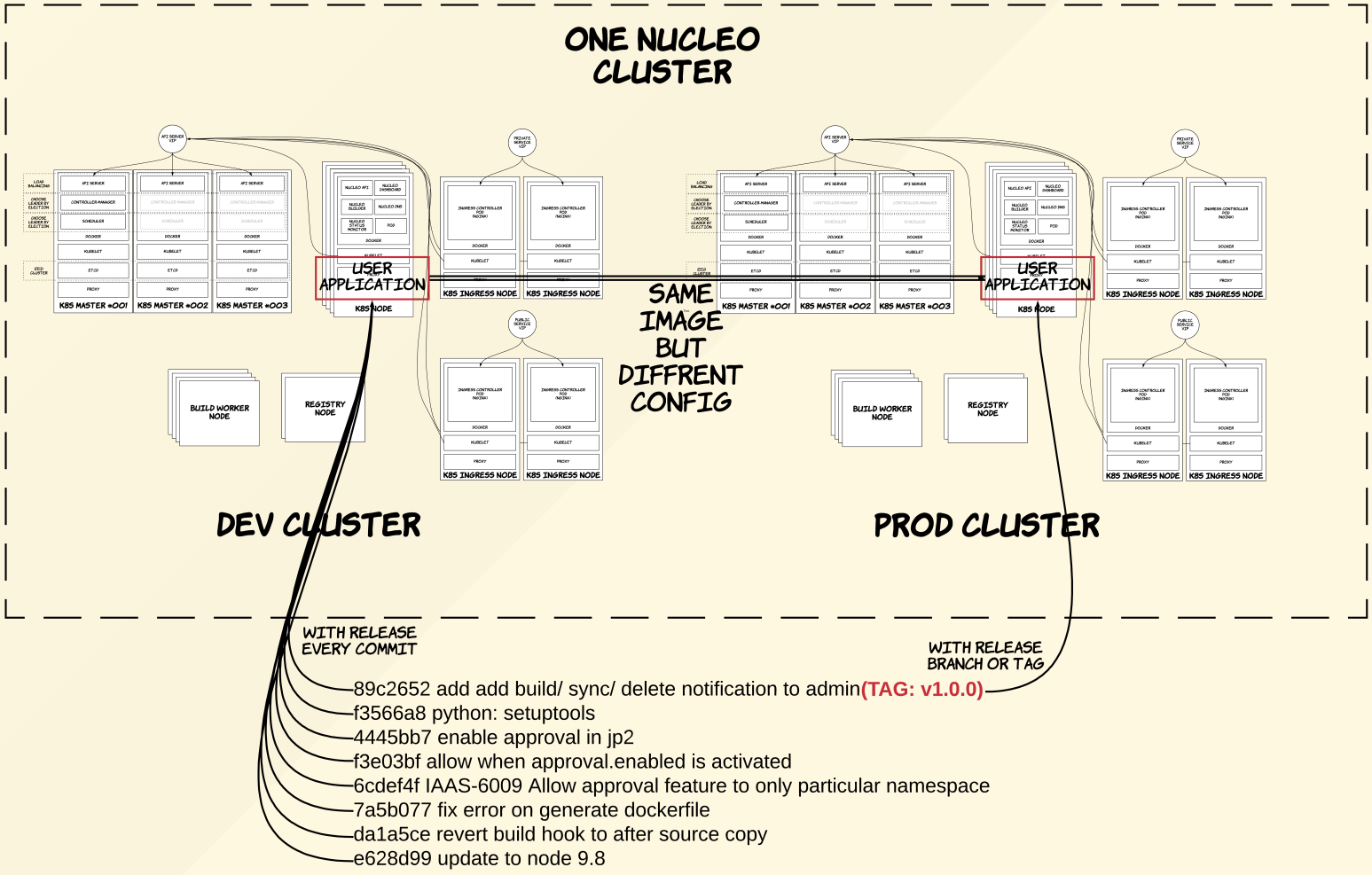
Dedicated node pattern for ingress



Update k8s & nucleo architecture



How to support continuous delivery



- currently not use kubernetes federation

winner winner chicken dinner

Notify kubernetes-dev of lifting code freeze	Lead			20			
v1.10.0	Branch Manager			3/26		week 13	1.10-blocking
v1.11.0-alpha.1	Branch Manager			27			
Release retrospective	Community			29			
1.11 Release Cycle Begins	Next Lead				2		

Thank you