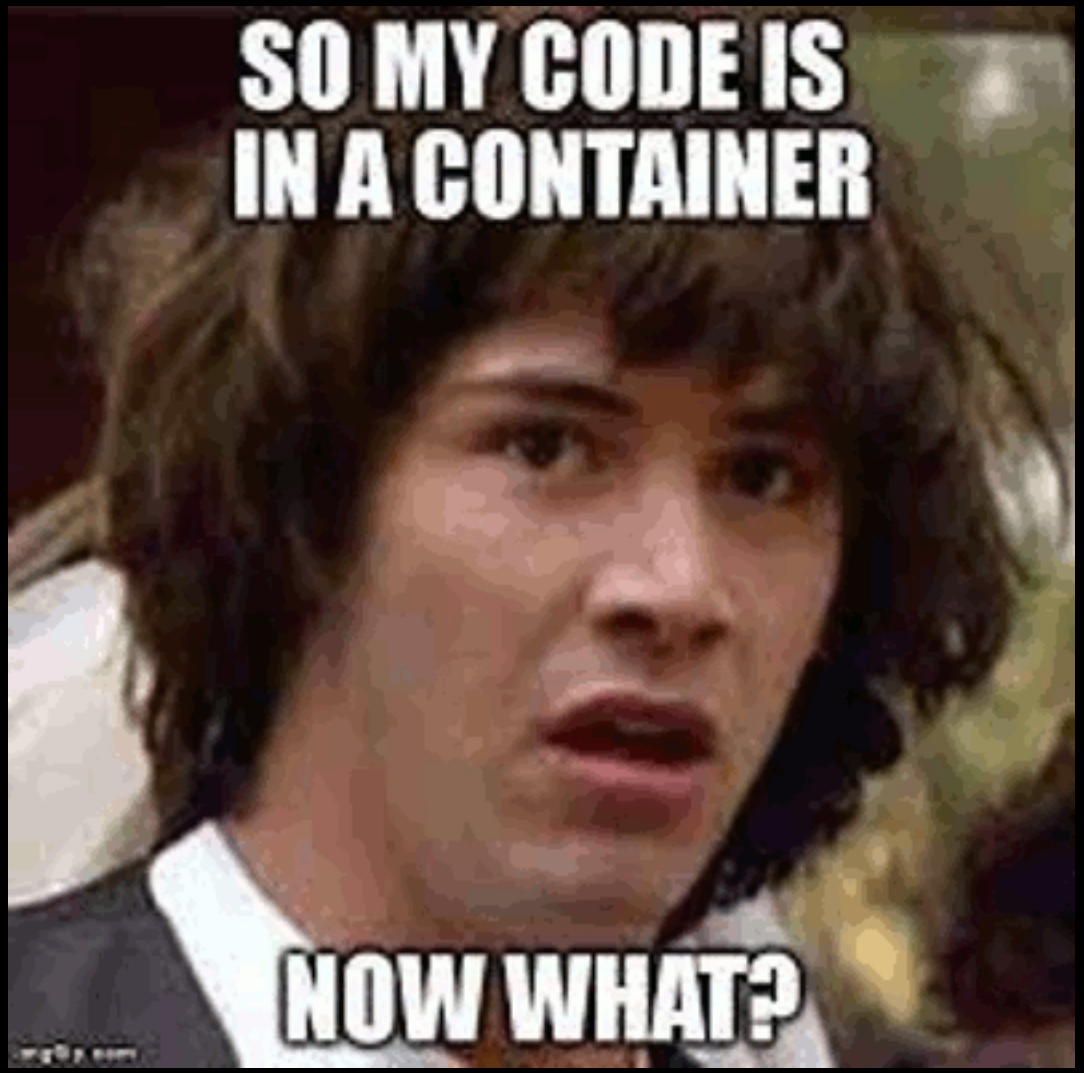

SO MY CODE IS
IN A CONTAINER
NOW WHAT?

Contenedores

- ✔ Imágenes "livianas" e inicio rápido.
- ✔ Portables.
- ✔ Económicos.
- ✔ Escalables.

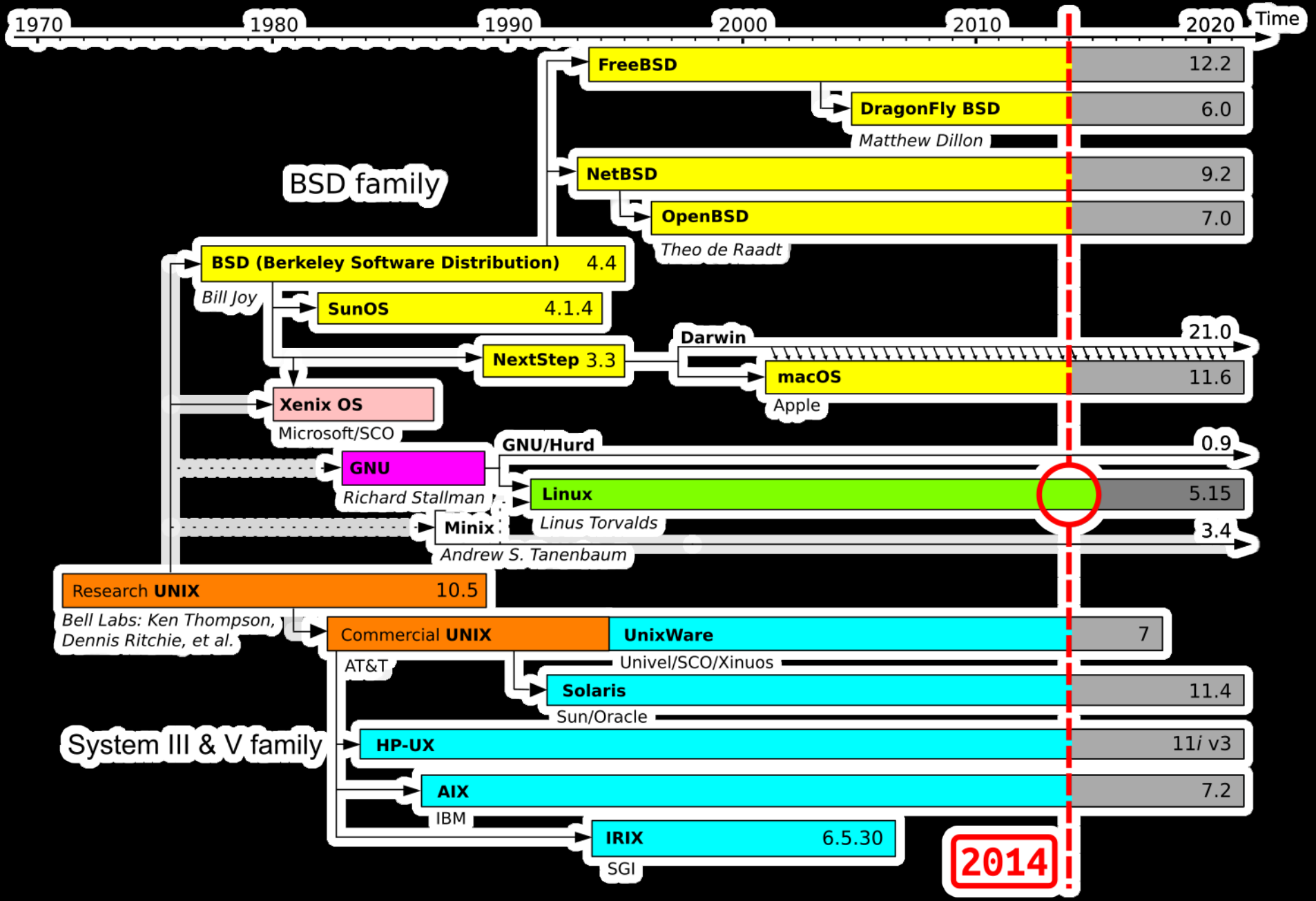
- ✖ Se pueden salir de control.
- ✖ Escasez de RRHH con conocimiento.
- ✖ No están completamente aislados.

Problemas de Administración

- Despliegue
- Escalado
- Actualizaciones
- Caída de Servidores


KUBERNETES

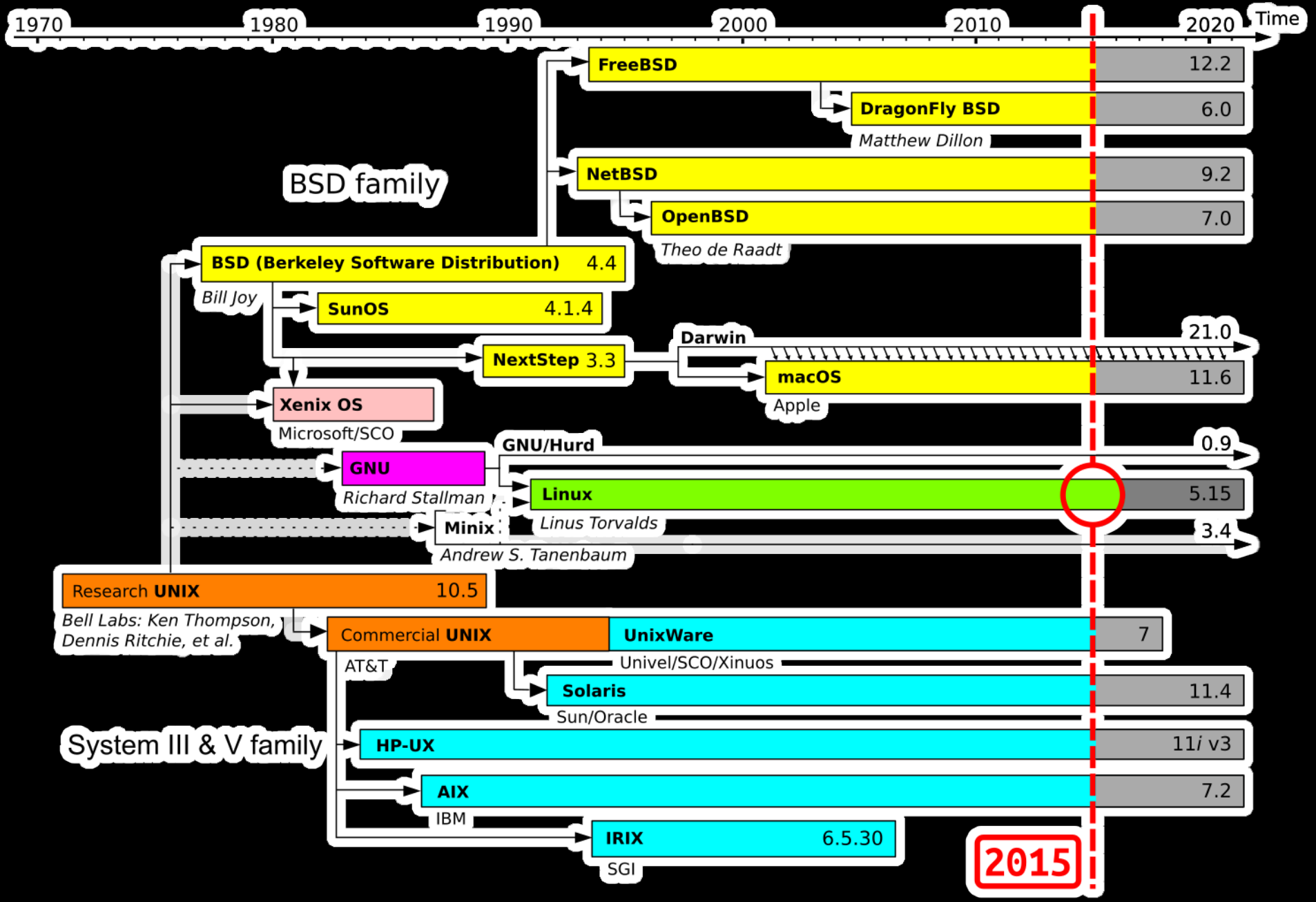

SO YOU WANT TO DO
CONTAINERS AND
KUBERNETES?
makeameme.org


1970
1980
1990
2000
2010
2020
Time
BSD family
FreeBSD
12.2
DragonFly BSD
6.0
Matthew Dillon
NetBSD
9.2
OpenBSD
7.0
Theo de Raadt
BSD (Berkeley Software Distribution)
4.4
Bill Joy
SunOS
4.1.4
NextStep 3.3
Darwin
21.0
macOS
11.6
Apple
Xenix OS
Microsoft/SCO
GNU/Hurd
0.9
GNU
Richard Stallman
Linux
5.15
Linus Torvalds
Minix
3.4
Andrew S. Tanenbaum
Research UNIX
10.5
Bell Labs: Ken Thompson,
Dennis Ritchie, et al.
Commercial UNIX
AT&T
UnixWare
7
Univel/SCO/Xinuos
Solaris
11.4
Sun/Oracle
System III & V family
HP-UX
11i v3
AIX
7.2
IBM
IRIX
6.5.30
SGI
2014

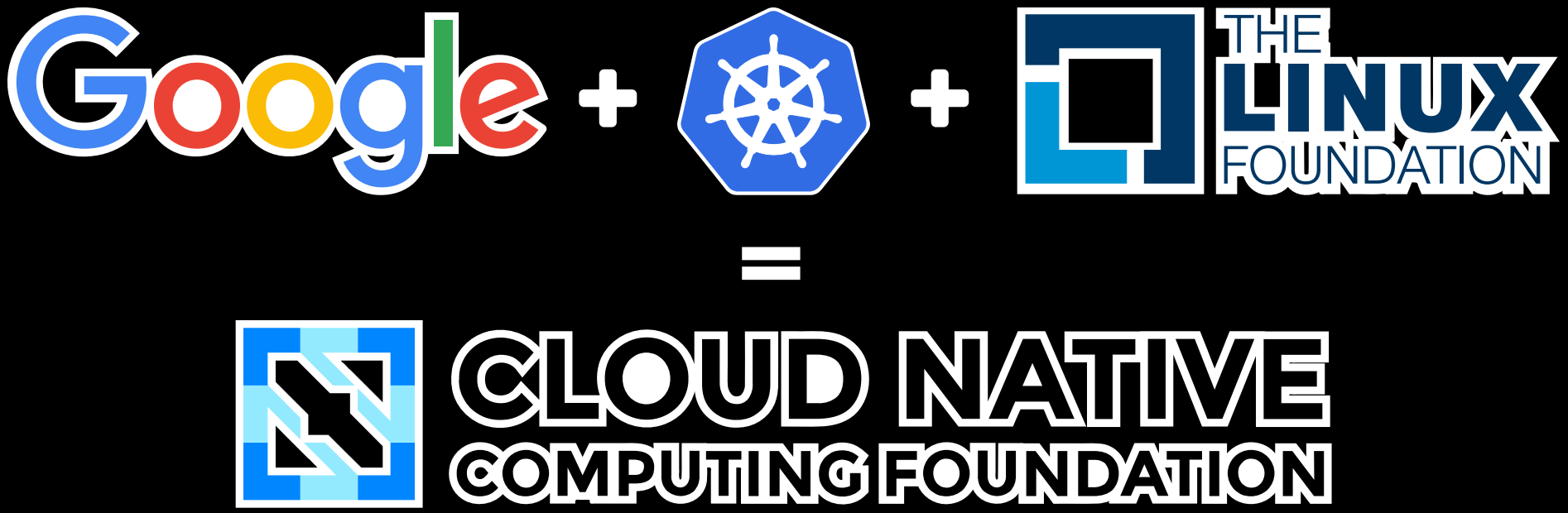
2014 - Kubernetes / K8s (Google)

Sistema de código abierto para orquestar contenedores. Kubernetes permite automatizar el despliegue, escalado y la administración de software.

El nombre "kubernetes" proviene del griego antiguo y significa "timonel". (wikipedia) (comic)


1970
1980
1990
2000
2010
2020
Time
BSD family
FreeBSD
12.2
DragonFly BSD
6.0
Matthew Dillon
NetBSD
9.2
OpenBSD
7.0
Theo de Raadt
BSD (Berkeley Software Distribution)
4.4
Bill Joy
SunOS
4.1.4
NextStep 3.3
Darwin
21.0
macOS
11.6
Apple
Xenix OS
Microsoft/SCO
GNU/Hurd
0.9
GNU
Richard Stallman
Linux
5.15
Linus Torvalds
Minix
3.4
Andrew S. Tanenbaum
Research UNIX
10.5
Bell Labs: Ken Thompson,
Dennis Ritchie, et al.
Commercial UNIX
UnixWare
7
Univel/SCO/Xinuos
AT&T
Solaris
11.4
Sun/Oracle
System III & V family
HP-UX
11i v3
AIX
7.2
IBM
IRIX
6.5.30
SGI
2015

2015 - CNCF



La CNCF fue anunciada junto a Kubernetes 1.0, la cual Google contribuyó a la Linux Foundation como tecnología semilla. Entre los miembros fundacionales encontramos a Google, CoreOS, Mesosphere, Red Hat, Twitter, Huawei, Intel, Cisco, IBM, Docker, Univa y Vmwarel. Al día de hoy la CNCF está compuesta por más de 450 miembros. (wikipedia)

containerd es un estándar industrial del núcleo del motor de ejecución de los contenedores. Maneja todo el ciclo de vida de los contenedores. En 2015, Docker donó la especificación OCI junto a una implementación de referencia llamada runC.

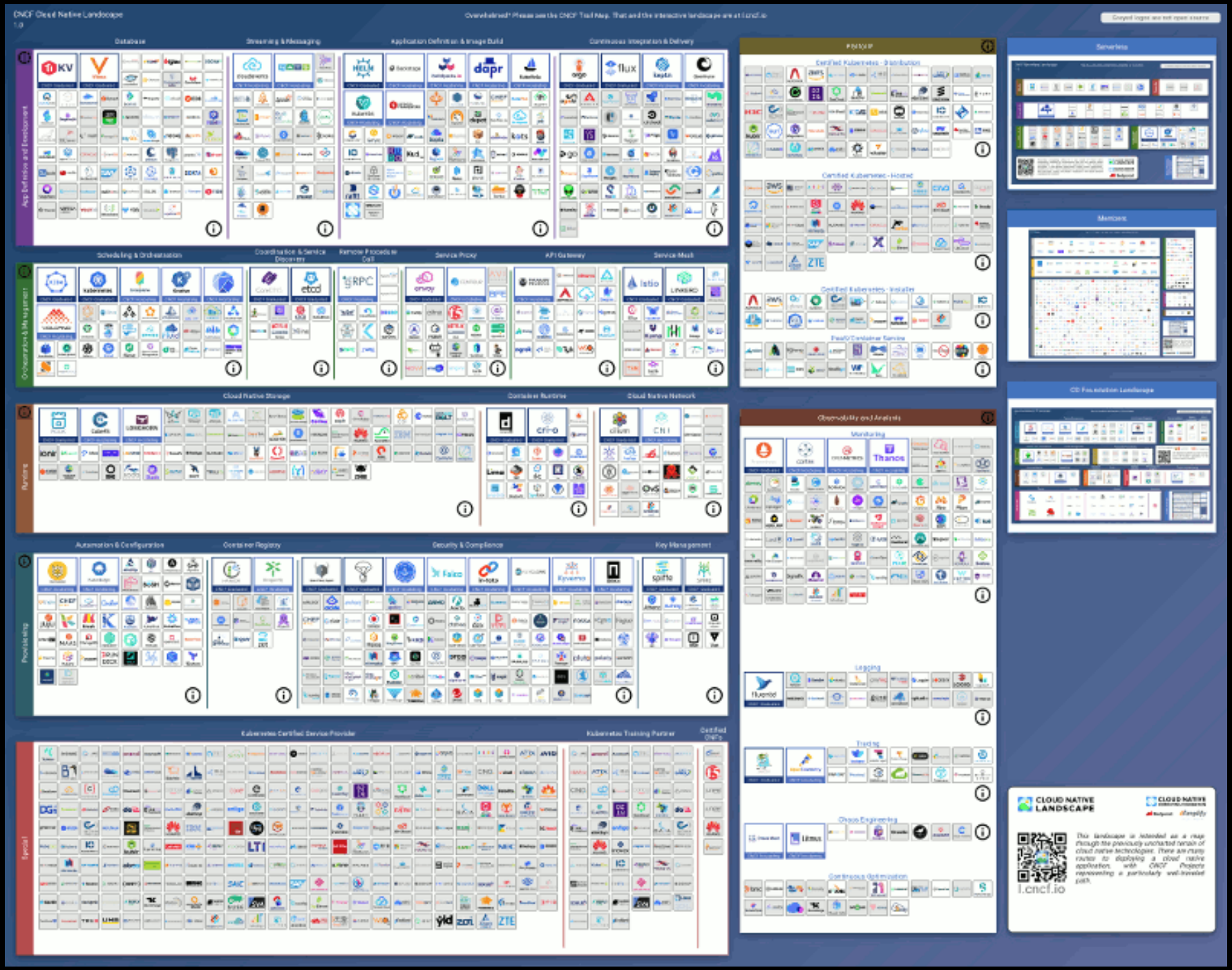
CoreDNS es un servidor DNS que permite encadenar complementos.

etcd es un almacén de clave-valor distribuido, proveyendo un método para almacenar datos a lo largo de un grupo (cluster) de máquinas.

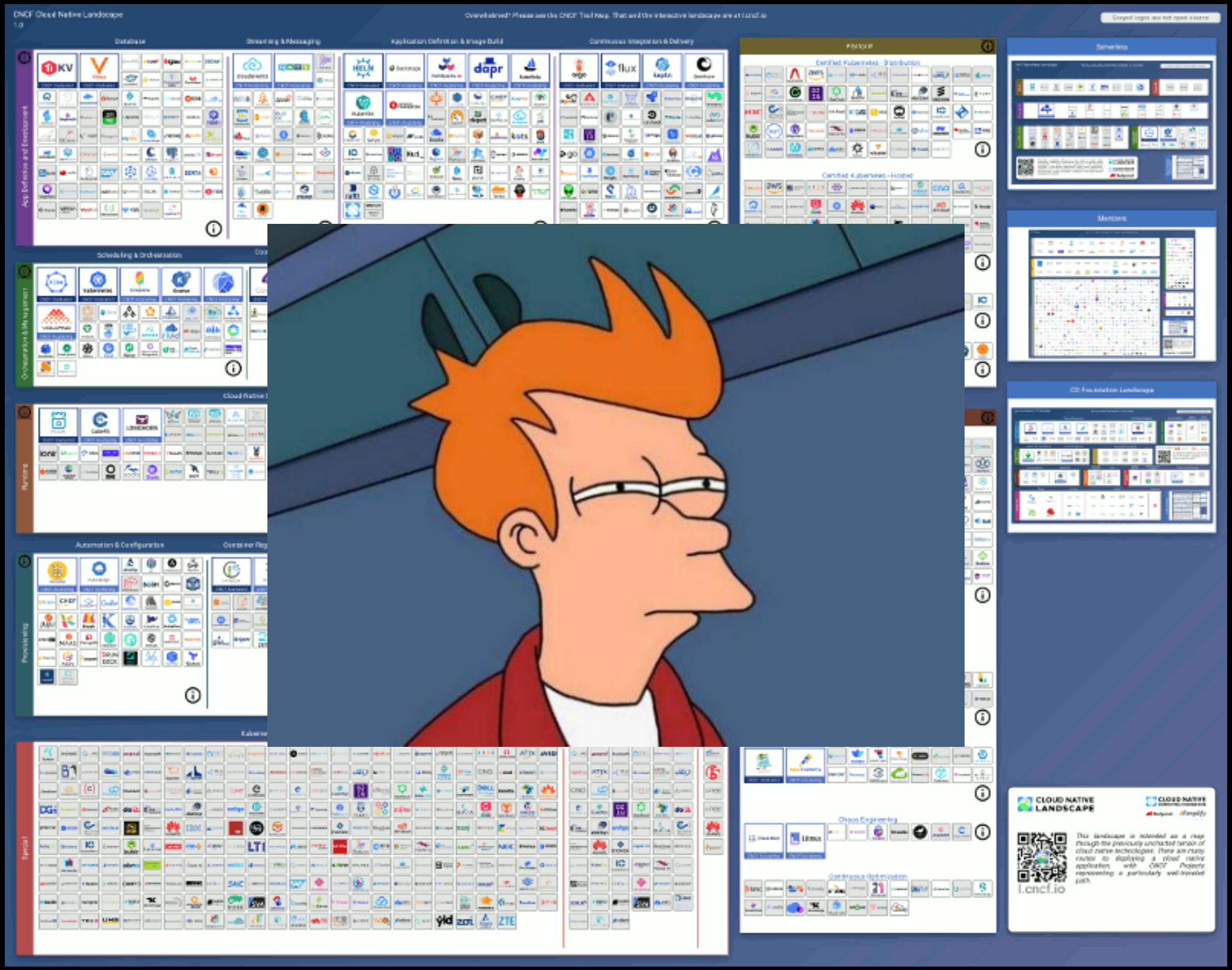
Helm es un administrador de paquetes que ayuda a los desarrolladores a "administrar y desplegar las aplicaciones en un cluster de k8s de manera fácil".

Linkerd agrega funciones de observabilidad, seguridad y confiabilidad a las aplicaciones al agregarlas a la plataforma en lugar de a la capa de aplicación. También cuenta con un "pequeño proxy" para maximizar la velocidad y seguridad del plano de datos.

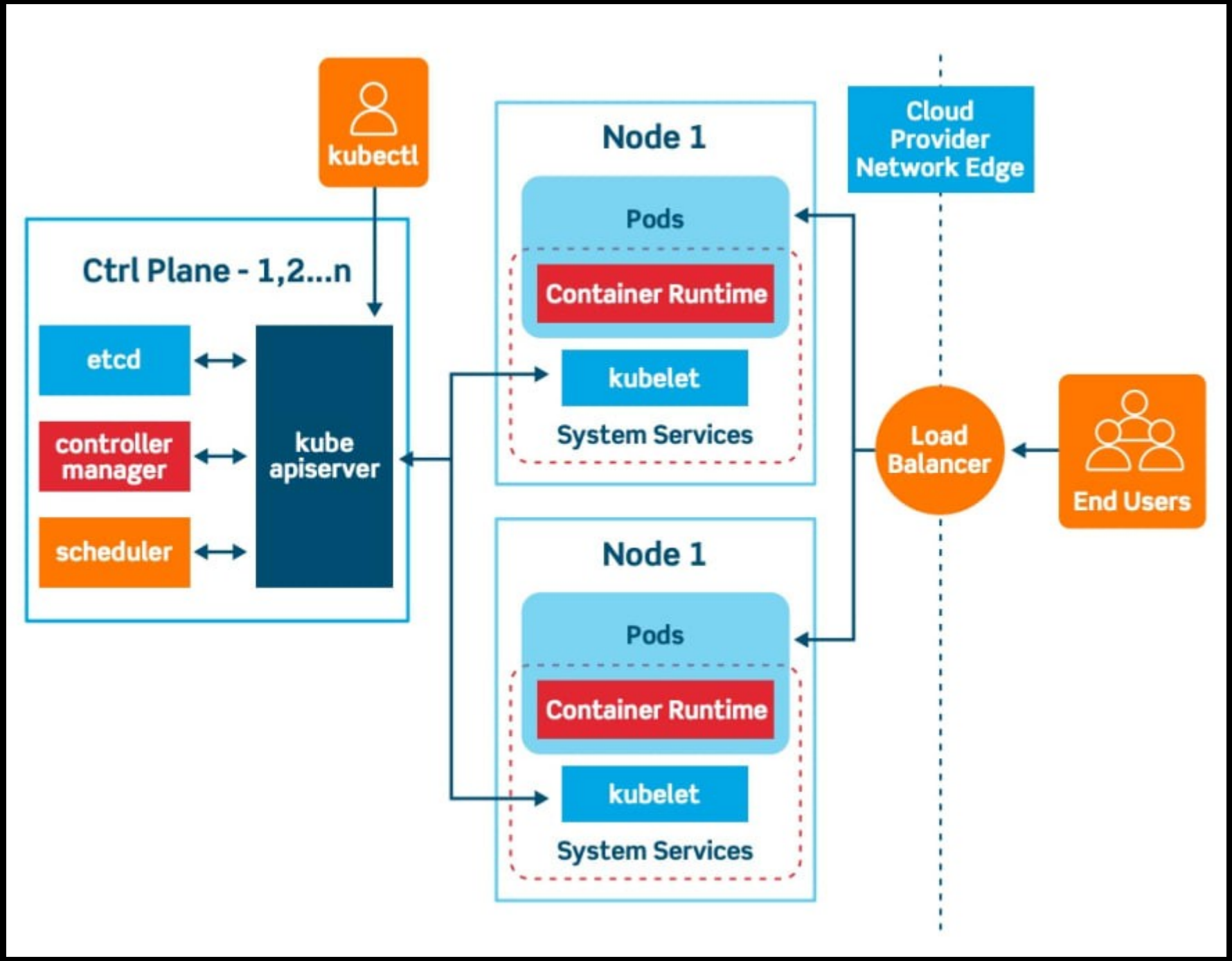

DISTRIBUTED SYSTEMS
HOW HARD CAN IT BE?

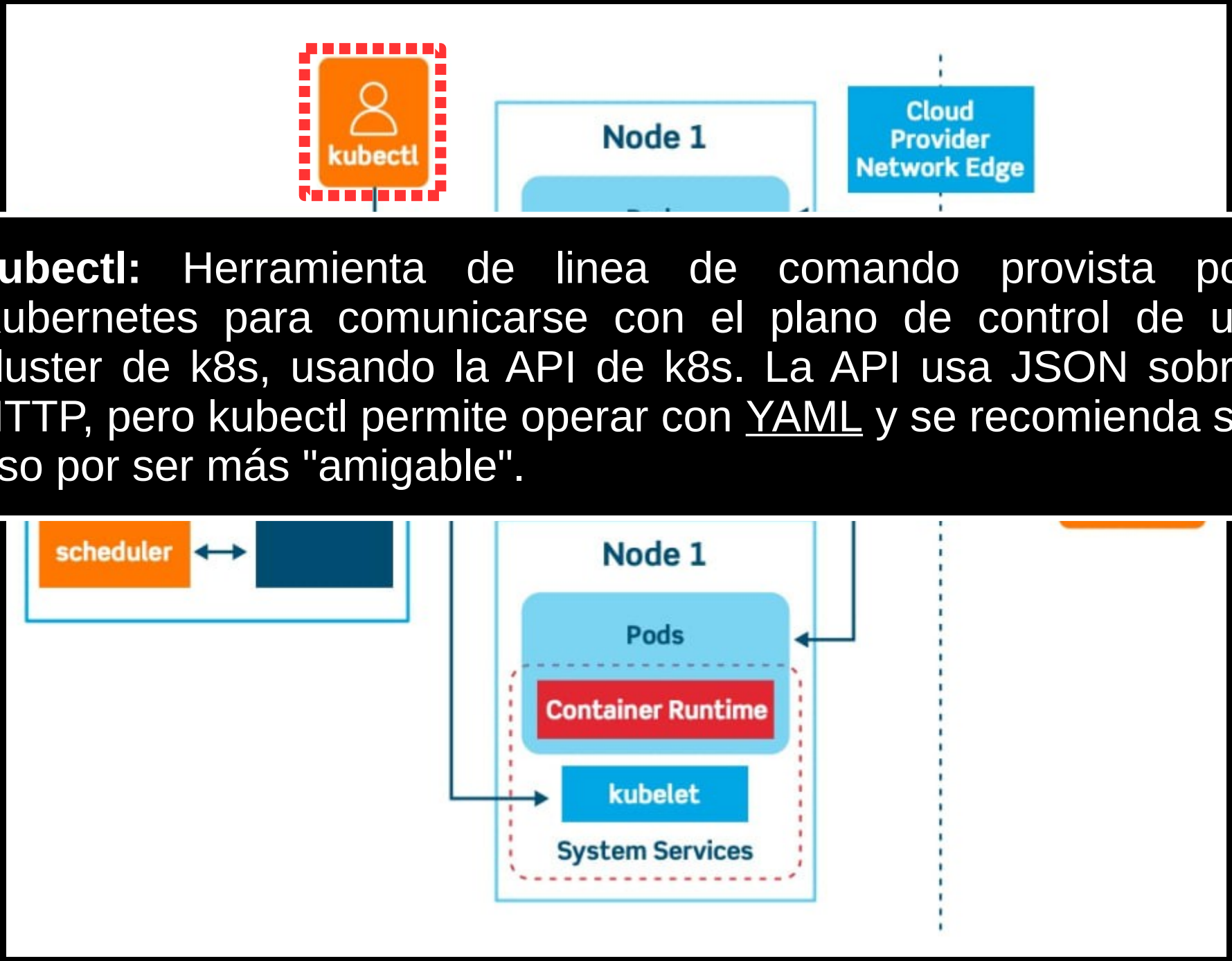


https://landscape.cncf.io/

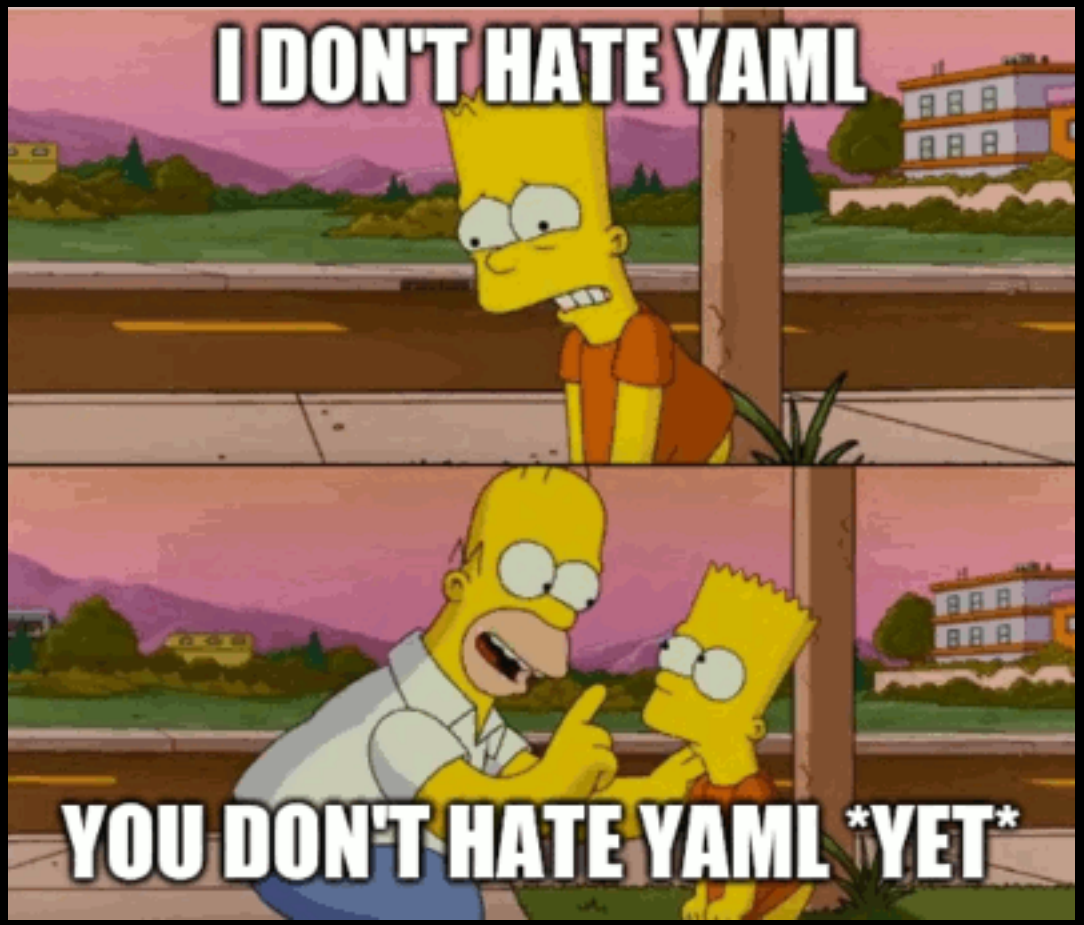


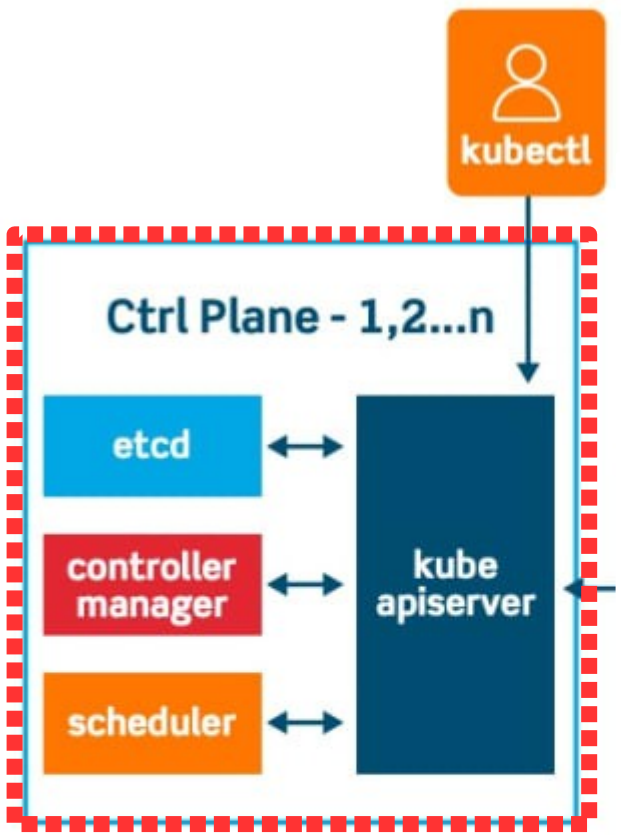
https://landscape.cncf.io/

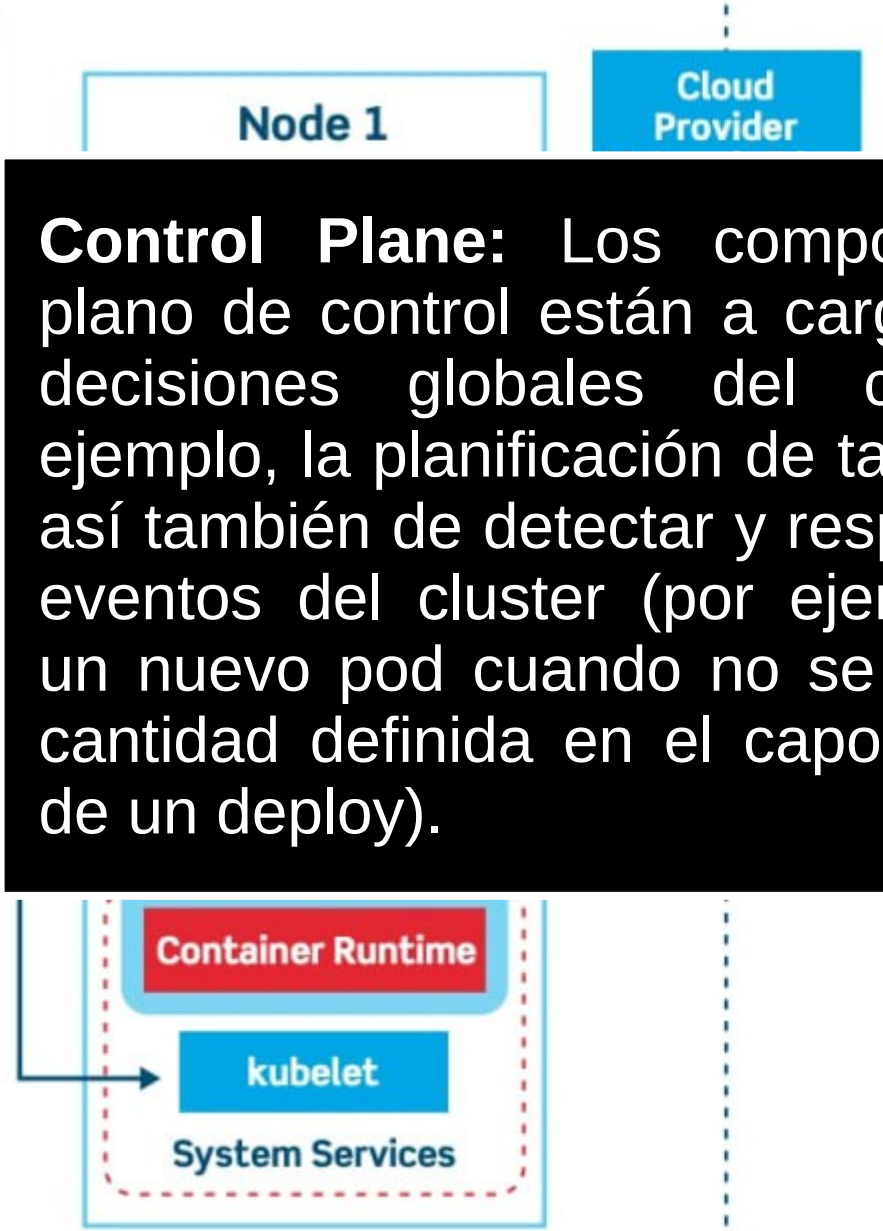

kubectl
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
Node 1
Pods
Container Runtime
kubelet
System Services
Node 1
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users



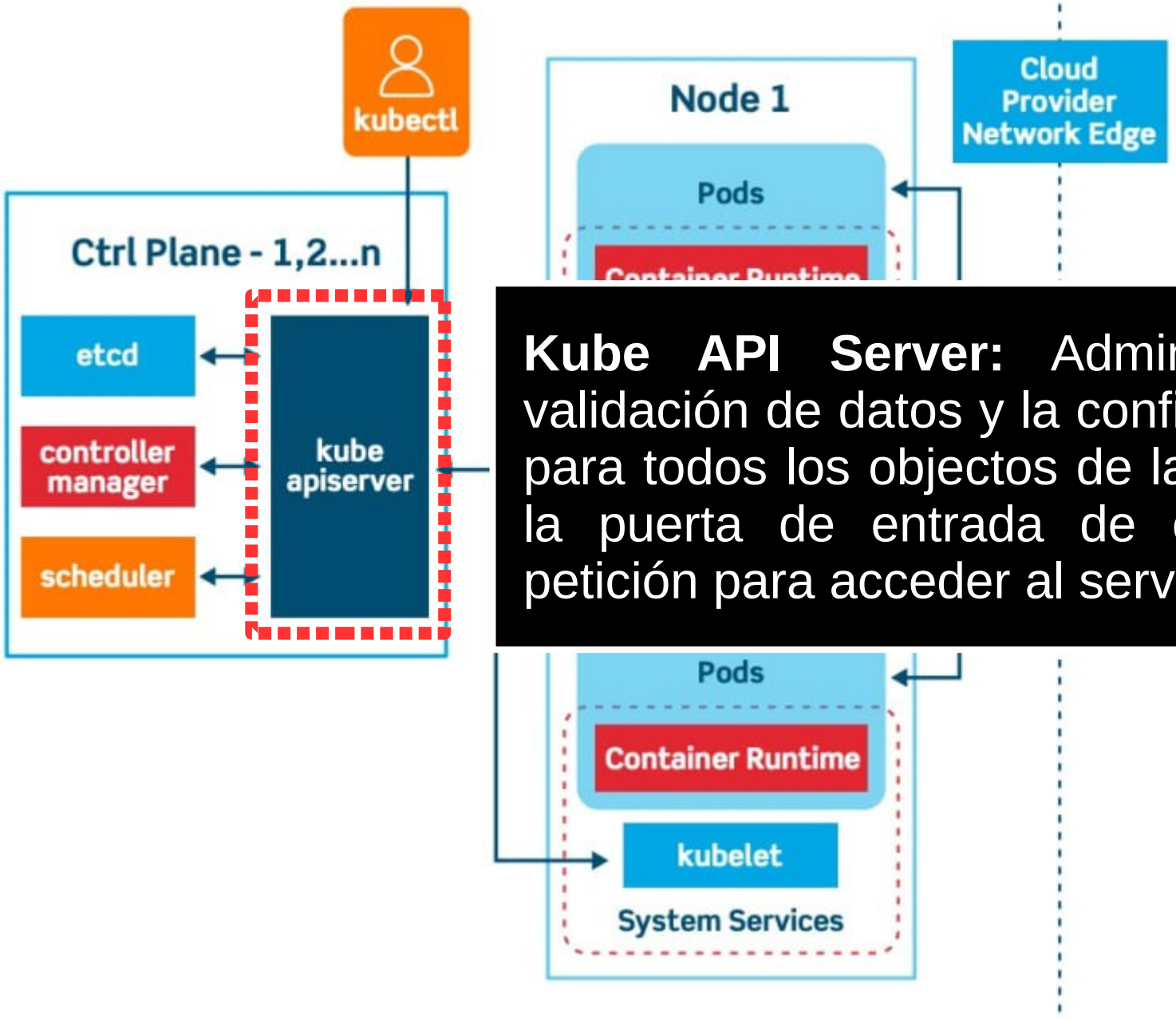
kubectl: Herramienta de linea de comando provista por Kubernetes para comunicarse con el plano de control de un cluster de k8s, usando la API de k8s. La API usa JSON sobre HTTP, pero kubectl permite operar con YAML y se recomienda su uso por ser más "amigable".


I DON'T HATE YAML
YOU DON'T HATE YAML *YET*

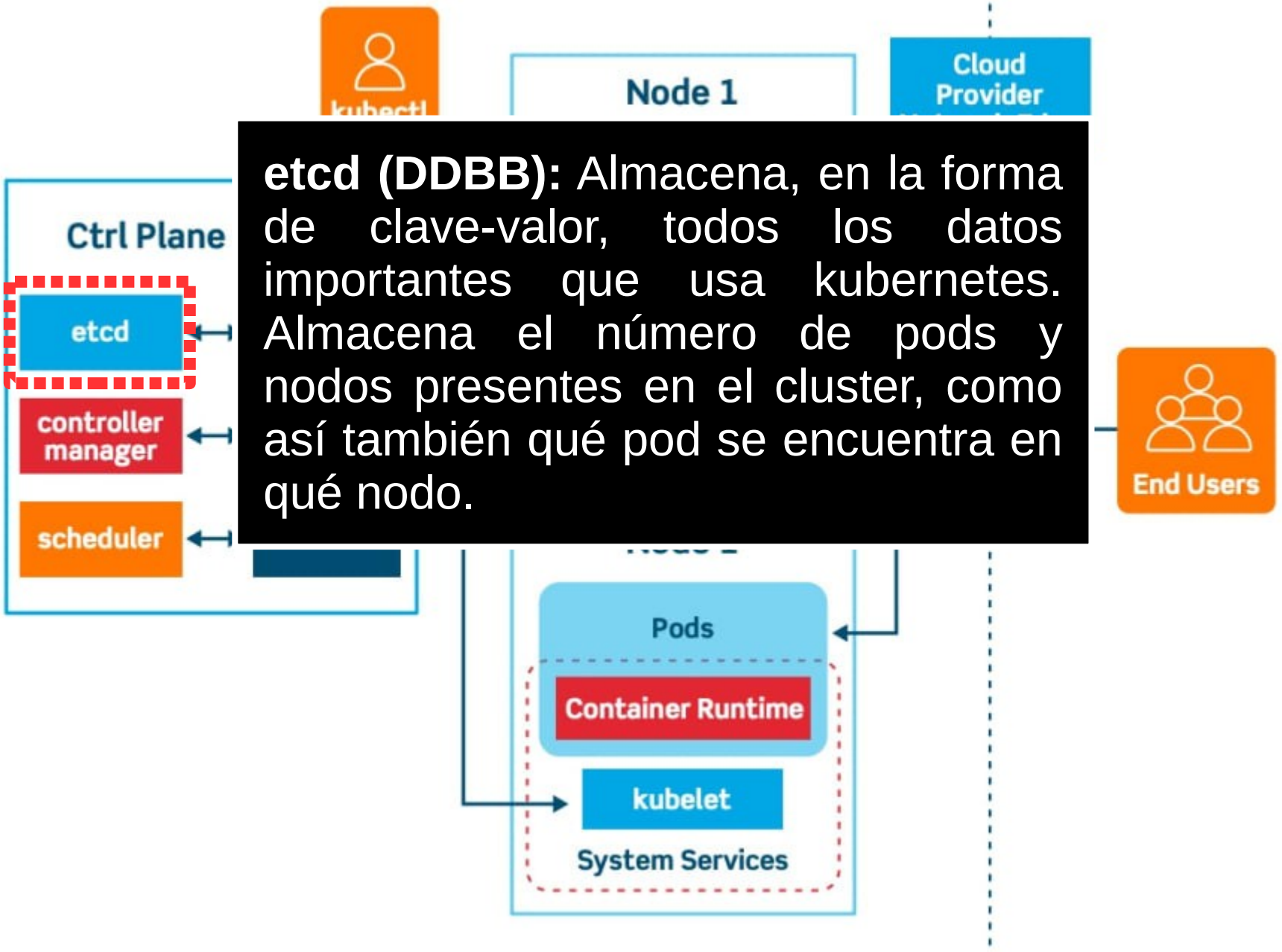




Control Plane: Los componentes del plano de control están a cargo de tomar decisiones globales del cluster (por ejemplo, la planificación de tareas), como así también de detectar y responder a los eventos del cluster (por ejemplo, iniciar un nuevo pod cuando no se satisface la cantidad definida en el capo de replicas de un deploy).



Kube API Server: Administra la validación de datos y la configuración para todos los objectos de la API. Es la puerta de entrada de cualquier petición para acceder al servidor.

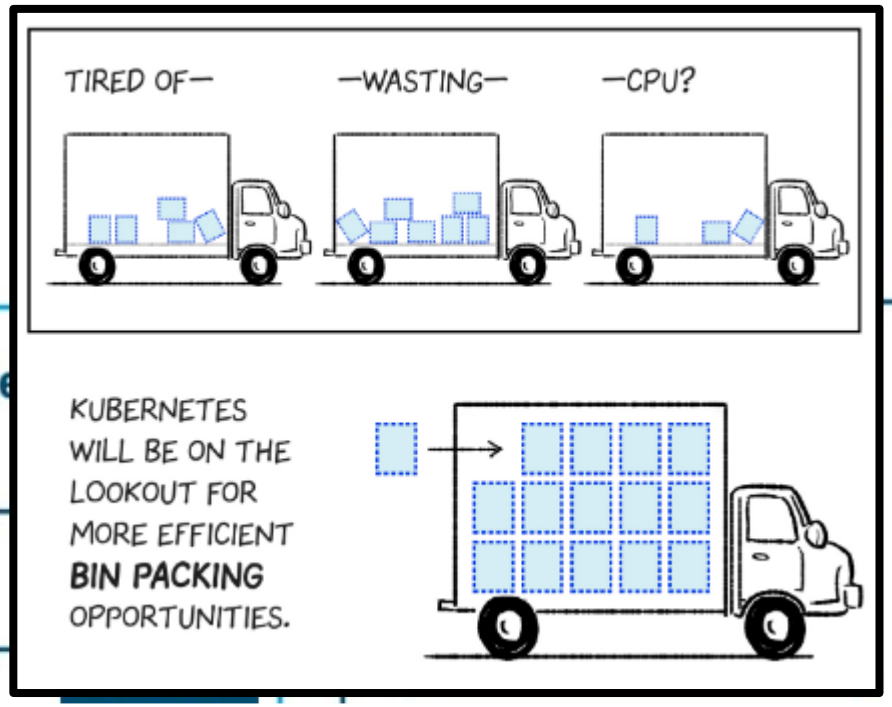


etcd (DDBB): Almacena, en la forma de clave-valor, todos los datos importantes que usa kubernetes. Almacena el número de pods y nodos presentes en el cluster, como así también qué pod se encuentra en qué nodo.

Controller Manager: Actuá como el cerebro de K8s, ya que la lógica principal se ejecuta en éste lugar. Su principal responsabilidad es la administración del ciclo de vida, ya que se asegura que todos los componentes funcionen correctamente. Existen diferentes tipos de controladores (controllers). Sin ser exhaustiva, a continuación se listan algunos ejemplos:

- ✔ Node Controller: Responsable de identificar y responder cuando se cae un nodo.
- ✔ Job Controller: Observa los objectos de trabajo (job) que representan las tareas únicas (crea los pods para ejecutar y completar éstas tareas).
- ✔ EndpointSlice Controller: Se encarga de popular los objectos que proveen un enlace entre los Servicios y los Pods.
- ✔ Replication Controller: Monitorea el estado de los pods. Se encarga de asegurar que el número deseado de pods se encuentre presente en el cluster.

Lógicamente, cada controlador es un proceso separado.



Cloud Provider Network Edge

Ctrl Plane

etcd

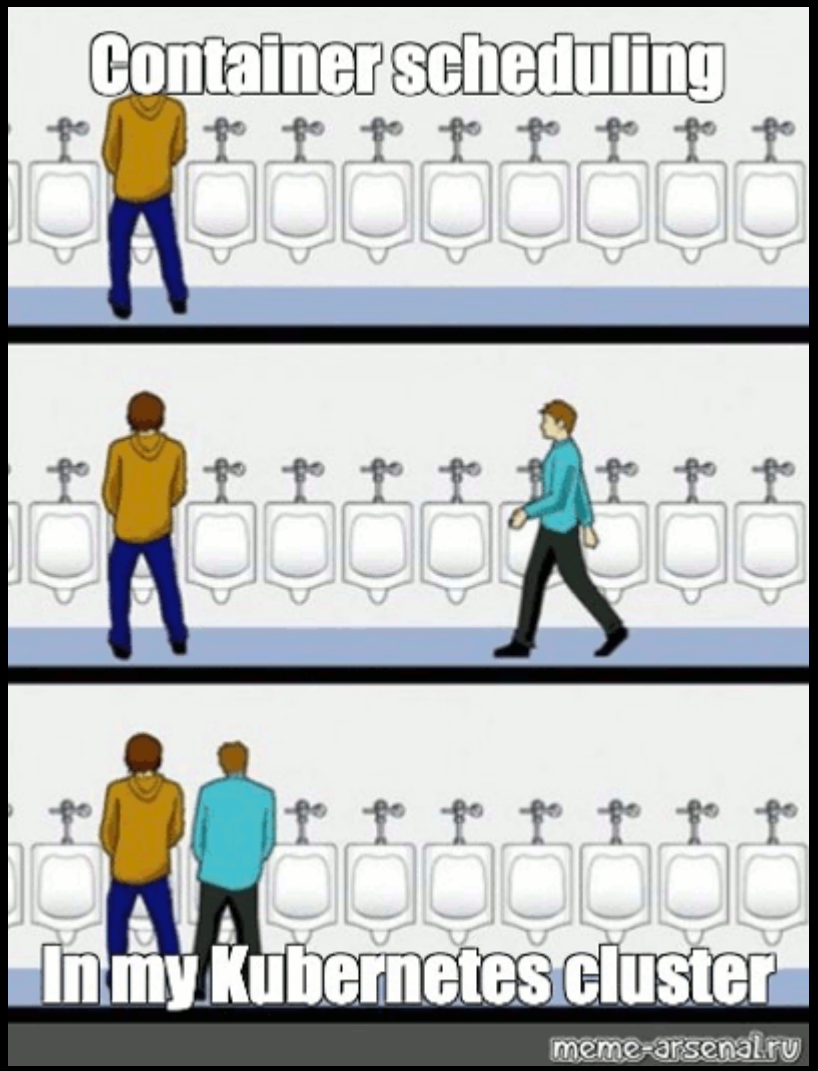
controller manager

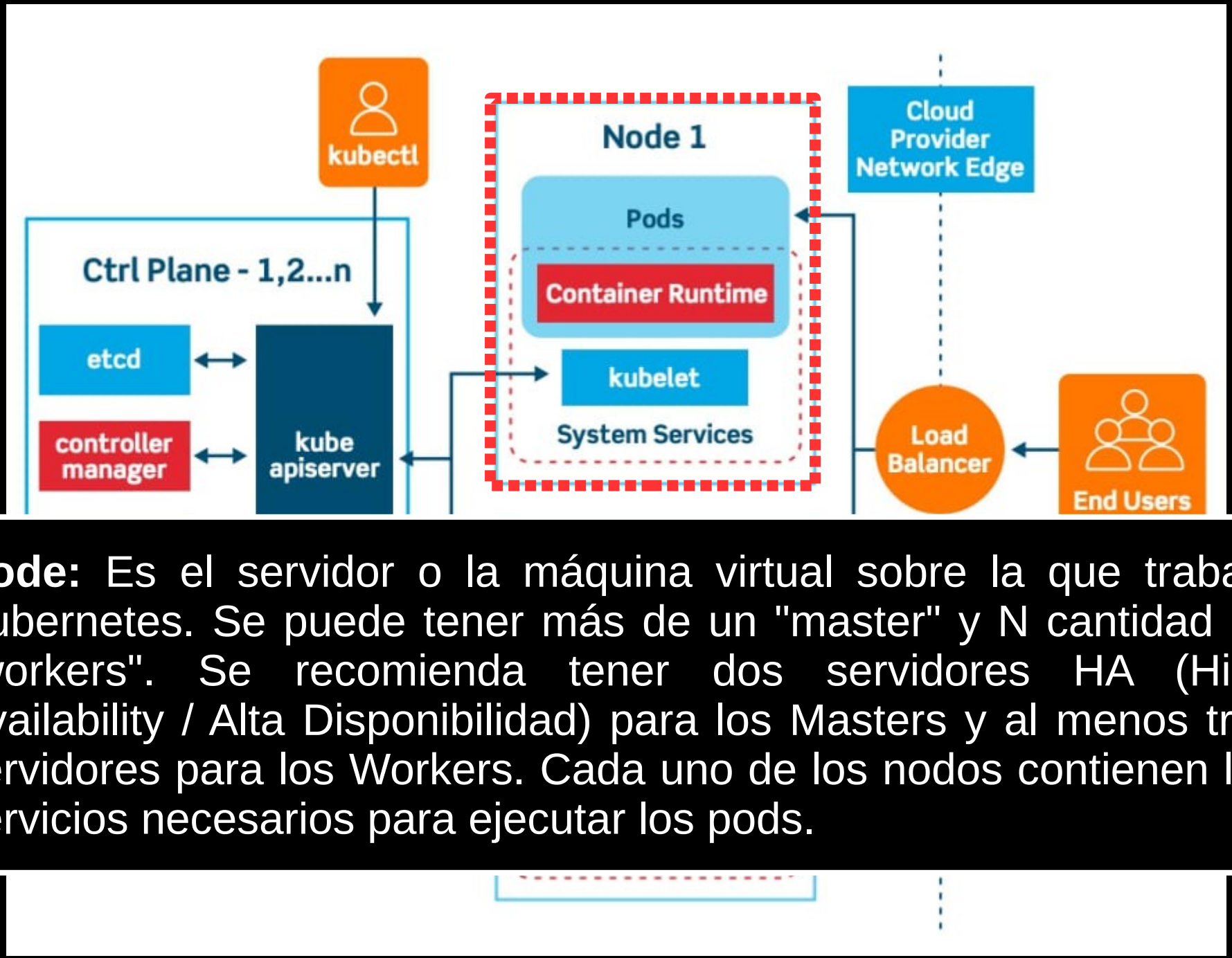
scheduler

Load Balancer

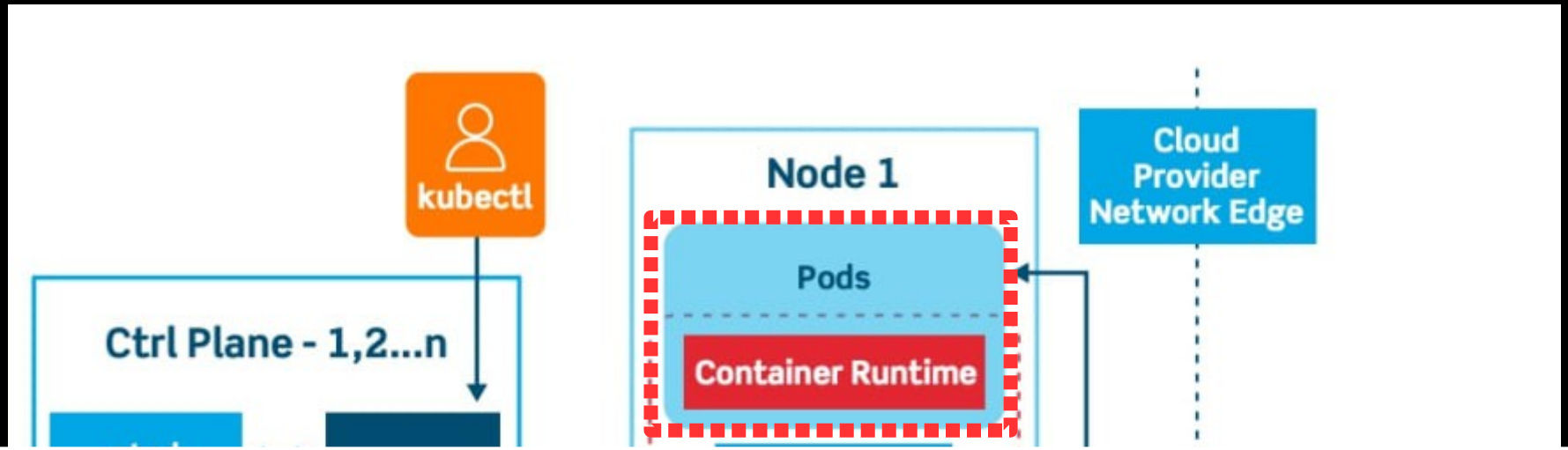
System Services

Scheduler: Es responsable de decidir exactamente donde se ejecutará un pod. Determina al nodo más adecuado para ejecutar los pods, teniendo en cuenta los requerimientos de los pods y los recursos de los nodos.

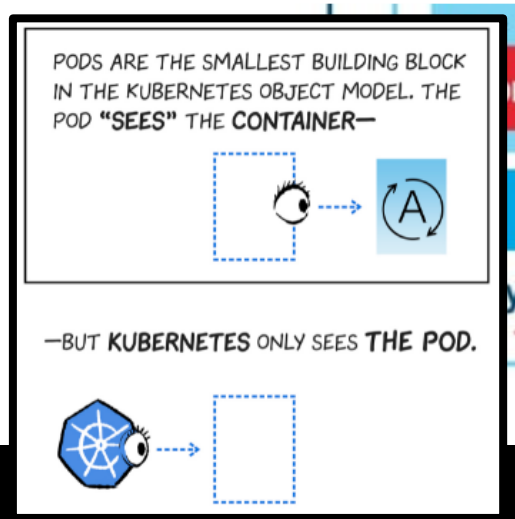

Container scheduling
In my Kubernetes cluster
meme-arsenal.ru

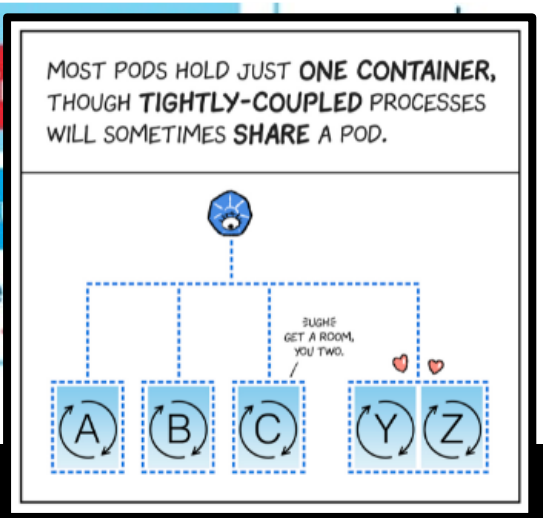


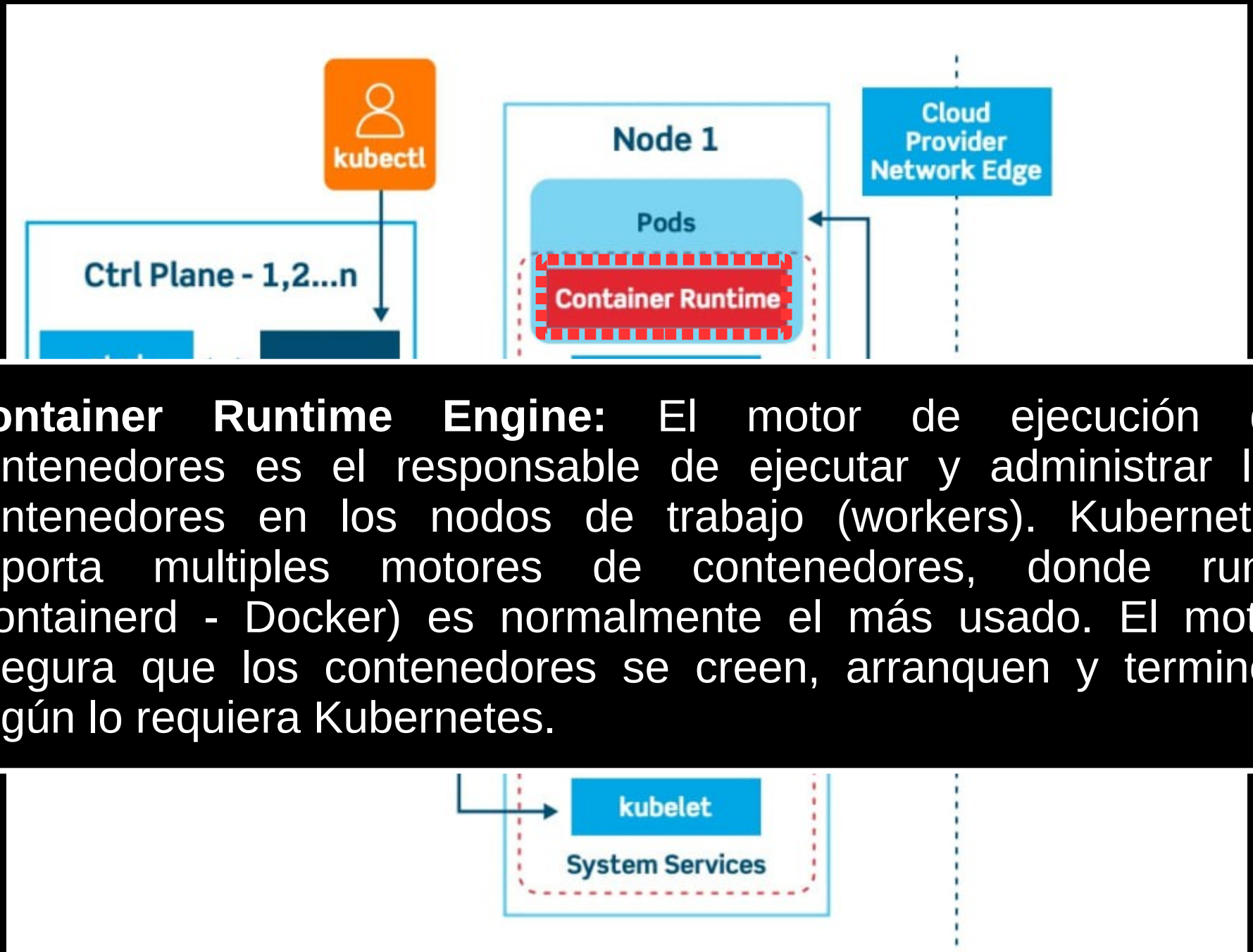
Node: Es el servidor o la máquina virtual sobre la que trabaja Kubernetes. Se puede tener más de un "master" y N cantidad de "workers". Se recomienda tener dos servidores HA (High Availability / Alta Disponibilidad) para los Masters y al menos tres servidores para los Workers. Cada uno de los nodos contienen los servicios necesarios para ejecutar los pods.



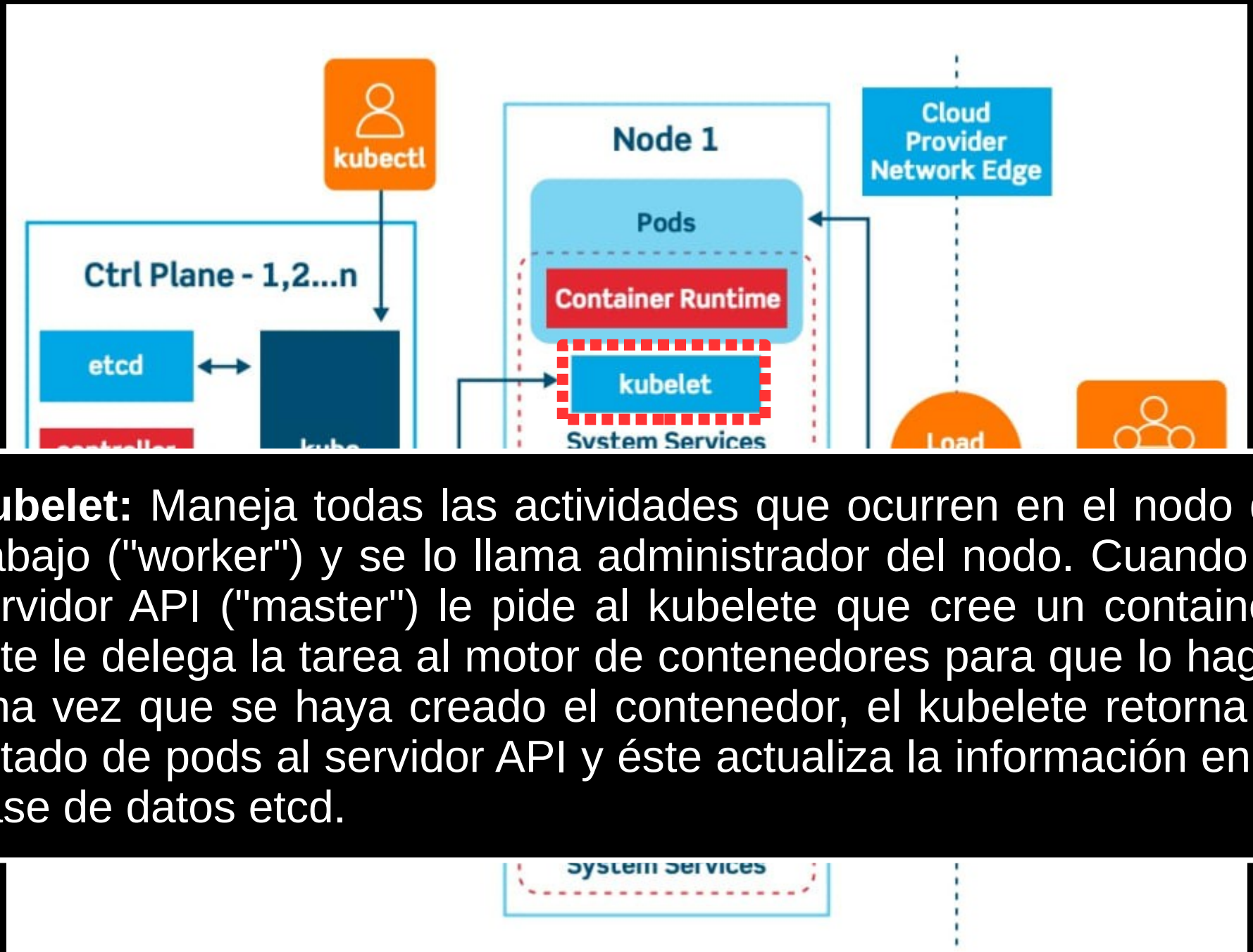
Pod: Es la unidad más pequeña que se puede desplegar en Kubernetes. Dentro de los pods se ejecutan nuestras aplicaciones. Un pod es el envoltorio (wrapper) de los contenedores. Dentro de un pod se pueden desplegar multiples contenedores que usen el mismo almacenamiento, recursos y red.



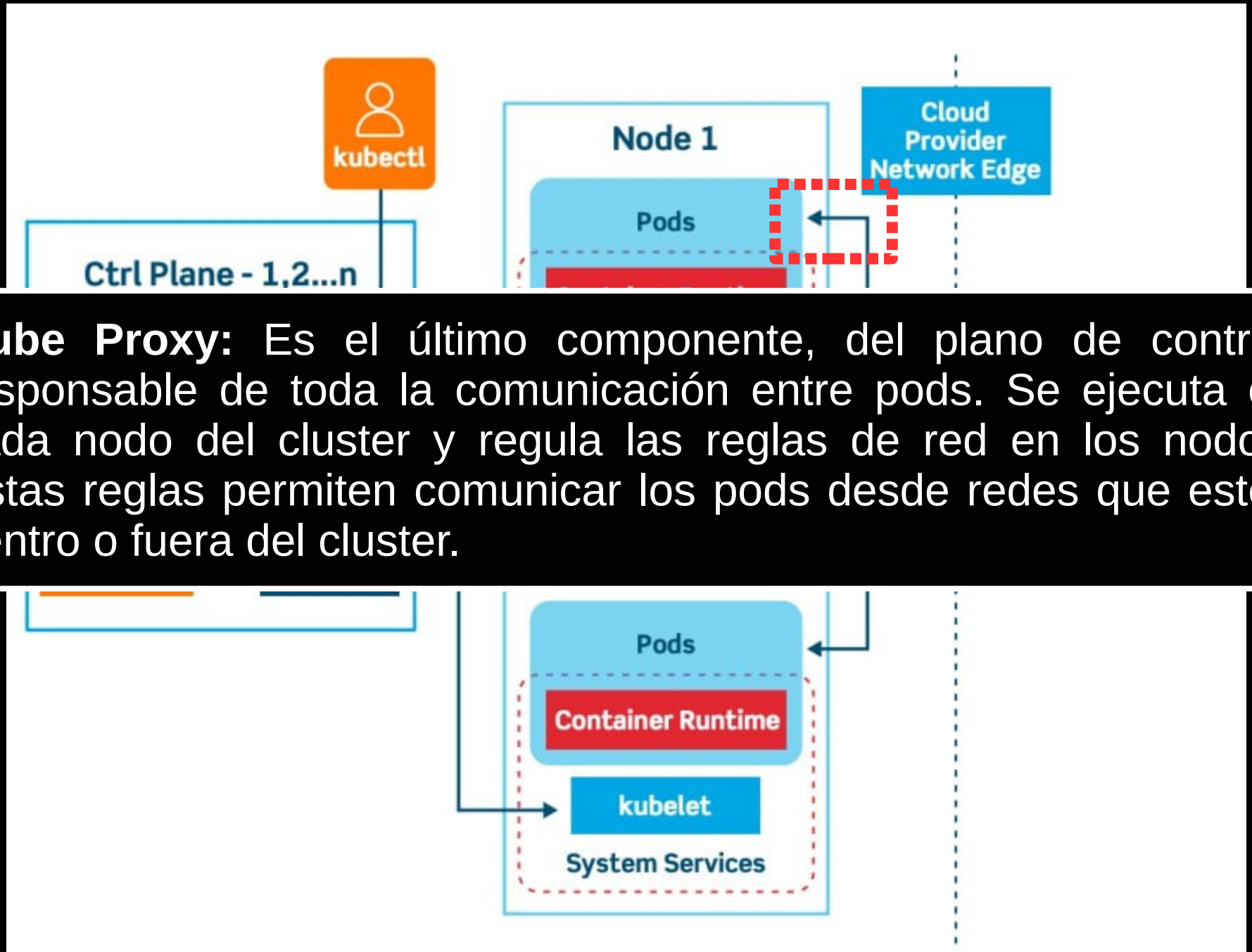




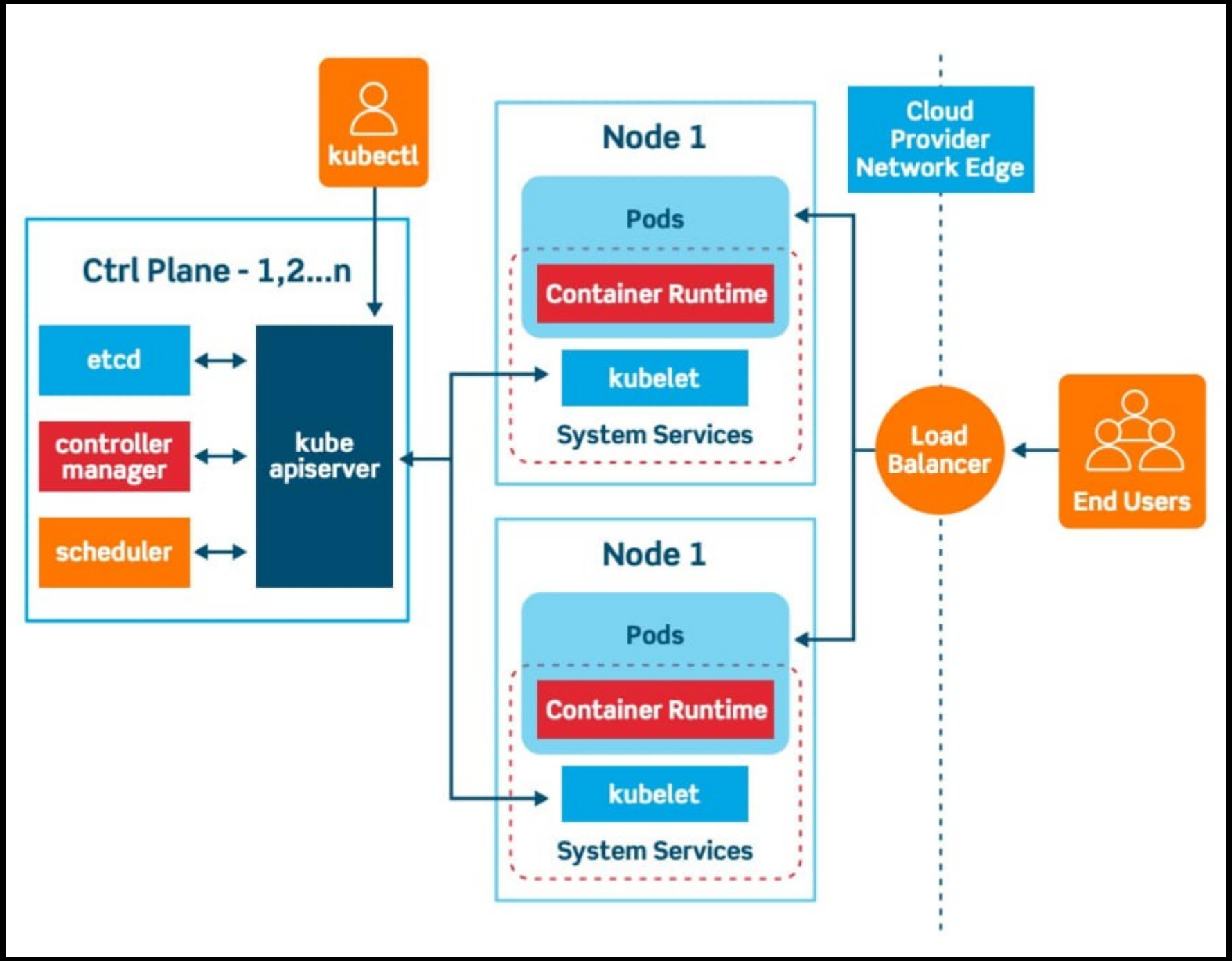
Container Runtime Engine: El motor de ejecución de contenedores es el responsable de ejecutar y administrar los contenedores en los nodos de trabajo (workers). Kubernetes soporta multiples motores de contenedores, donde runC (containerd - Docker) es normalmente el más usado. El motor asegura que los contenedores se creen, arranquen y terminen según lo requiera Kubernetes.



Kubelet: Maneja todas las actividades que ocurren en el nodo de trabajo ("worker") y se lo llama administrador del nodo. Cuando el servidor API ("master") le pide al kubelete que cree un container, éste le delega la tarea al motor de contenedores para que lo haga. Una vez que se haya creado el contenedor, el kubelete retorna el estado de pods al servidor API y éste actualiza la información en la base de datos etcd.



Kube Proxy: Es el último componente, del plano de control, responsable de toda la comunicación entre pods. Se ejecuta en cada nodo del cluster y regula las reglas de red en los nodos. Éstas reglas permiten comunicar los pods desde redes que estén dentro o fuera del cluster.


kubectl
Ctrl Plane - 1,2...n
etcd
controller manager
scheduler
kube apiserver
Node 1
Pods
Container Runtime
kubelet
System Services
Node 1
Pods
Container Runtime
kubelet
System Services
Cloud Provider Network Edge
Load Balancer
End Users

Entorno Local

- Docker Desktop
- Podman Desktop
- Minikube
- Kind
- MicroK8s
- k3d


YOU WENT FULL
KUBERNETES ON YOUR DESKTOP
NEVER GO FULL
KUBERNETES ON YOUR DESKTOP
imgflip.com

Herramientas

- k3d (k3s + dind)
- kubectl
- kompose

- k3d demo: https://github.com/k3d-io/k3d-demo/.
- k3s: k8s optimizado para usar menos recursos.
- dind: Docker in Docker.


WE HEARD YOU LIKE DOCKER
SO WE PUT A DOCKER IN YOUR DOCKER


SAY DOCKER
ONE MORE TIME

```
docker --version # >= v20.10.5
> Docker version 20.10.7, build f0df350
runc --version # >= v1.0.0-rc93
> runc version 1.0.0-rc95
> commit: b9ee9c6314599f1b4a7f497e1f1f856fe433d3b7
> spec: 1.0.2-dev
> go: go1.13.15
> libseccomp: 2.5.1
k3d version
> k3d version v5.6.0
> k3s version v1.27.4-k3s1 (default)
kubectl version --client=true
> Client Version: v1.28.3
> Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
kompose version
> 1.31.2 (a92241f79)
# Test k3d cluster
k3d cluster create edi2
kubectl cluster-info
kubectl get nodes
k3d cluster list
k3d cluster delete edi2
```

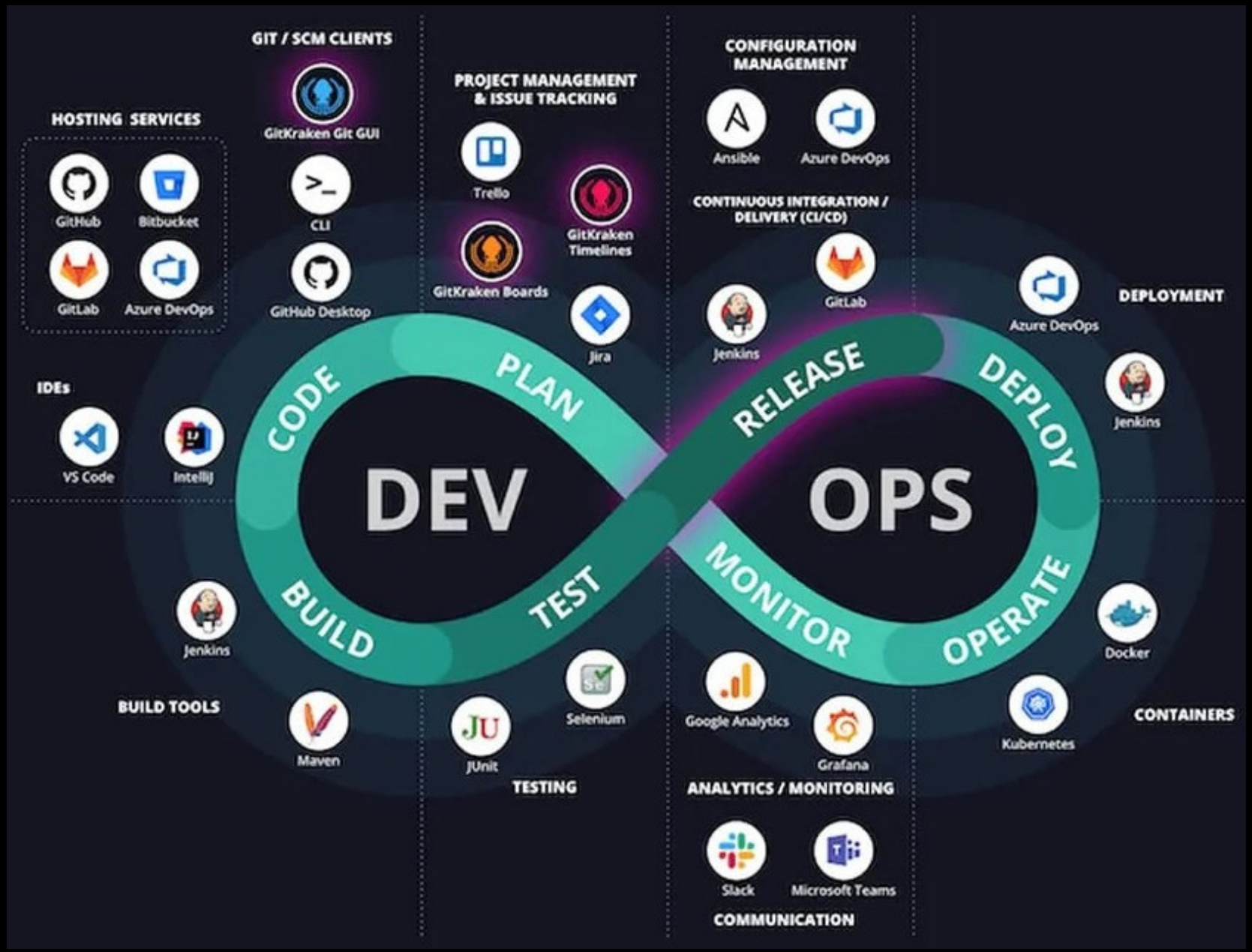
Usar el proyecto VOTAR (cliente)

```
cat docker-compose.yml
> version: '3'
> services:
>     edi2-server:
>         image: edi2/server:0.1
>         ports:
>             - '9080:80'
>     edi2-client:
>         image: edi2/client:0.1
>         ports:
>             - '9090:80'
>         environment:
>             - RESTAPI_HOST=localhost
>             - RESTAPI_PORT=9080
mkdir k8s
kompose convert --out k8s/
> INFO Kubernetes file "k8s/edi2-client-service.yaml" created
> INFO Kubernetes file "k8s/edi2-server-service.yaml" created
> INFO Kubernetes file "k8s/edi2-client-deployment.yaml" created
> INFO Kubernetes file "k8s/edi2-server-deployment.yaml" created
```

```
# Tag images ('latest' bug)
# - https://github.com/k3d-io/k3d/issues/920
# - https://github.com/kubernetes/kubernetes/issues/47775
docker tag edi2/server:latest edi2/server:0.1
docker tag edi2/client:latest edi2/client:0.1
# Create a cluster
k3d cluster create edi2cluster \
    --api-port 6550 \
    --port 9080:9080 \
    --port 9090:9090 \
    --wait
# Load VOTAR images into the cluster
k3d image import -c edi2cluster edi2/server:0.1
k3d image import -c edi2cluster edi2/client:0.1
# Check uploaded images
docker exec k3d-edi2cluster-server-0 ctr image list | grep edi
# Apply k8s objects
kubectl apply -f k8s/
# Check service
curl localhost:9090 | less
```


IT WORKED, MARTY!
IT WORKED!

	docker	kubernetes
Purpose	Development	Production
Configuration File Format	YAML	YAML
Container Orchestration	Local	Cluster
Scalability	Limited	Unlimited
Features	Basic	Advanced
Ease of Use	Simple	Complex


GIT / SCM CLIENTS
GitKraken Git GUI
CLI
GitHub Desktop
HOSTING SERVICES
GitHub
Bitbucket
GitLab
Azure DevOps
PROJECT MANAGEMENT & ISSUE TRACKING
Trello
GitKraken Timelines
GitKraken Boards
Jira
CONFIGURATION MANAGEMENT
Ansible
Azure DevOps
CONTINUOUS INTEGRATION / DELIVERY (CI/CD)
GitLab
Jenkins
DEPLOYMENT
Azure DevOps
Jenkins
IDEs
VS Code
Intellij
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
MONITOR
OPERATE
DEV
OPS
Jenkins
BUILD TOOLS
Maven
JUnit
Selenium
TESTING
Google Analytics
Grafana
ANALYTICS / MONITORING
Docker
Kubernetes
CONTAINERS
Slack
Microsoft Teams
COMMUNICATION

Learning Kubernetes