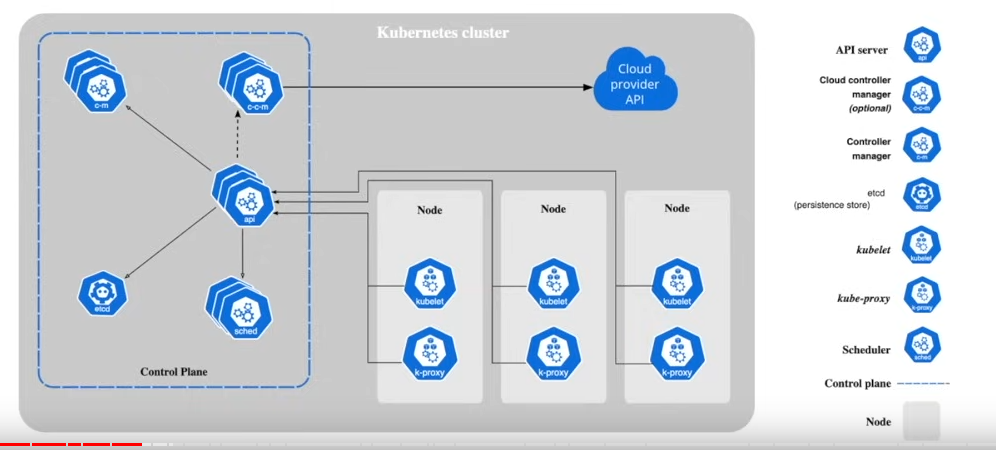
**Kubernetes**

¿Qué es? ¿Por qué existen los kubernetes? Esta tecnología es la continuación a Docker-Compose.

Para empezar a programar y poder compartir entornos utilizamos **Docker.** Esto nos da la ventaja de poder compartir nuestros entornos y máquinas virtuales con compañeros y volver a instalarlos sin complicaciones asumiendo que tendremos siempre las mismas dependencias.

¿y si necesitas levantar más de un contenedor? Para eso se intentó utilizar docker-compose. A través de YAML o JSON te permite organizar un poco mejor 5 - 10 servidores para poder levantarlos más eficientemente y te puedes gestionar algunos Scripts para utilizarlos.

¿y si tienes 1000 servidores? PARA ESO PUEDES UTILIZAR KUBERNETES 😀



Kubernetes se divide en un Control Panel (A la izquierda) y las instancias también llamadas Workers a su derecha. También puedes observar como arriba a la derecha ofrece la posibilidad de replicar tus nodos a través de un Cloud Provider (Y esta ventaja es precisamente la virtud que diferencia a un orquestador con un gestor de contenedores… permite poder ampliar tus nodos a través de AWS… Google… IBM… Etc… y no solo en local de esa máquina)

**Control Panel**

Gestiona varios componentes como:

-Etcd: para almacenar la información del estado de los nodos y cloud component.

-Schedule: Gestor del manifiesto que declaras sobre tus nodos al principio de la instalación.

-c-c-m: Servicio para conectarse a API de algún Cloud como pueda ser IBM para ampliar espacio.

-c-m: Gestor que controla los servicios del panel de control

**Nodos**

Incluye dos componentes:

-Kubelet: Servicio instalado dentro de cada nodo que comunica el control panel con el nodo para dirigirlo.

-KProxy: Recibir tráfico y gestionarlo entre los PODS para enviarlo al nodo correcto.

**¿Cómo levantar un Cluster de Kubernetes?**

-Instalar Kubectl que es el gestor de los nodos. Tan sencillo como ir a internet y descargarlo

-Instalar un contenedor de kubernetes donde estén todos los componentes necesarios del control panel y se puede hacer tanto en local con MINIKUBE o en un proveedor cloud.

-Preparar el fichero KubeConfig (En el caso de cloud descargarlo y en el caso de trabajar con minikube se instala automáticamente) Basicamente lo que hace minikube es:

**export KUBECONFIG=(Path de archivo de configuración)**

-Kubectl get nodes → ya puedes ver los nodos

**Comandos comunes**

-kubectl get

-kubectl edit

-kubectl delete

-kubectl apply → Aplicar manifiesto en nodos

-kubectl exec → Permite ejecutar comandos

-kubectl Logs

-kubectl cp → Copiar archivos de nuestra máquina al server

-kuebctl port-forward → Exponer puertos a tu máquina

-kubectl cordon / uncordon → Cordon para que ese nodo no reciba más contenedores y al contrario

-kubectl drain → Denar nodos y poder moverlos

-kubectl config get-contexts → Archivos de contextos

-kubectl describe → Te lanza eventos en ese PODS específicos.

**TRICK**: lens app (Gestor visual de kubectl. Muy buen programa)

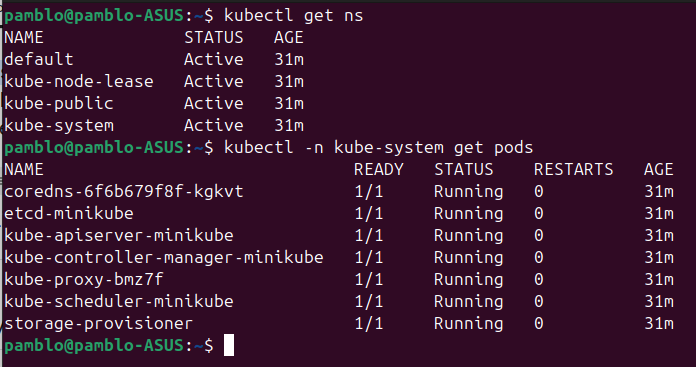
**Namespaces**

Son grupos de trabajo que están comunicados entre ellos mismos. Pero sería la primera capa de abstracción

**POD**

Un Pod es SET de contenedores. Es una agrupación de 1 o más contenedores dentro de un determinado Namespaces.

Para ver los pods que tenemos elegimos el namespaces donde queremos mirarlo y ejecutamos el comando de la siguiente manera



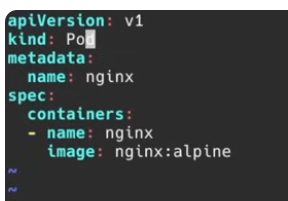
donde -n kube-system es el namespace elegido para obtener los pods. También para obtener información mas especifica podemos consultar

kubectl -n kube-system get pods -o wide

**FORMAS PARA DECLARAR PODS**

**Manifiesto de Kubernetes declarativo PODS**

Es un fichero YAML por ejemplo este:



1- Api version → Dependiendo del tipo que queremos declarar en la documentación puedes buscar que versión de API es necesario

2- kind → Eliges que estás declarando, en este caso un Pod

3- Metadata → Es para declarar etiquetas o nombre, en este caso el name es nginx

4- spec: es la especificación que se va a declarar dentro de este POD. En este caso buscamos un contenedor de docker nginx:alpine (lo busca en la image) y solo falta aplicarlo

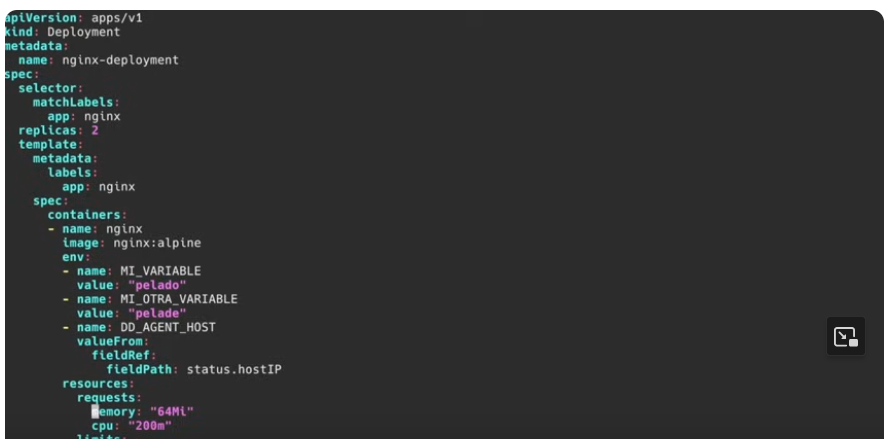
5- kubectl apply -f (Nombre del archivo)

**6- Si no especificamos -n se va al namespace Default. Por lo tanto cuando ejecutamos kubectl get pods lo podremos ver, ya que en el default no hace falta indicar namespace**

-ReadinessProbe y livenessProbe son parámetros que se pueden incluir en este manifiesto donde fijas en el Liveness value cuánto tiempo ha de esperar para comprobar que está abierto el puerto y readiness le dices cuánto ha de tardar en hacer la primera petición GET donde tu le digas

**Manifiesto de Deployments**

Este es un paso mas elevado. Un deployment es una plantilla para crear PODS. No le dices a kuberneters quiero dos PODS… Ahora le estás diciendo… Kubernetes quiero tener siempre un entorno con dos PODS para que te garantice ese servicio

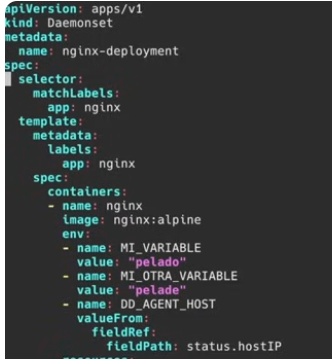


donde ahora el apiversion es apps/v1 según la documentación, kind es un “Deployment” y el primer spec hace garantizar 2 réplicas. ¿Dos réplicas de que? Dos réplicas del siguiente spec dónde estás declarando el pod de nginx que se replicará dos veces.

**Ahora aunque tu borres un pod, el manifiesto LO VOLVERÁ A REVIVIR**

**Manifiesto de DaemonSets**

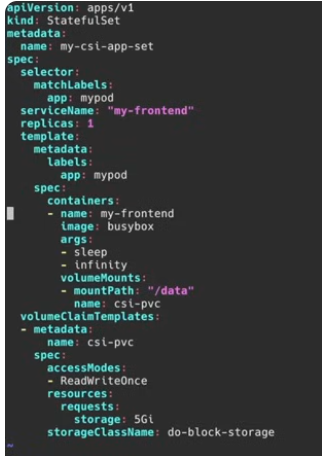
Es una forma de hacer un deploy de los PODS pero en TODOS TUS NODOS.



¿Para qué sirve esto? Esto permite desplegar una aplicación en un contenedor de logs y lanzarla directamente en todos los nodos que tienes activados.

**MANIFIESTO Statefullsets**

Es una forma de declarar PODS pero que están vinculados a volúmenes. Son pods por ejemplo para base de datos. Creas un POD de mySQL y lo vinculas a un espacio del disco, para que en el caso de que se tenga que revivir, siga apuntando a una memoria persistente



**SERVICIOS EN KUBERNETES**

Es la forma que tienes de comunicación entre pods. Hay tres tipos de servicios:

-ClusterIP – IP Fija dentro del Cluster entre todos los PODS

-NodePort – Crea un puerto en cada nodo para recibir tráfico

-Load Balancer – Crea balanceador en cloud y gestiona los pods