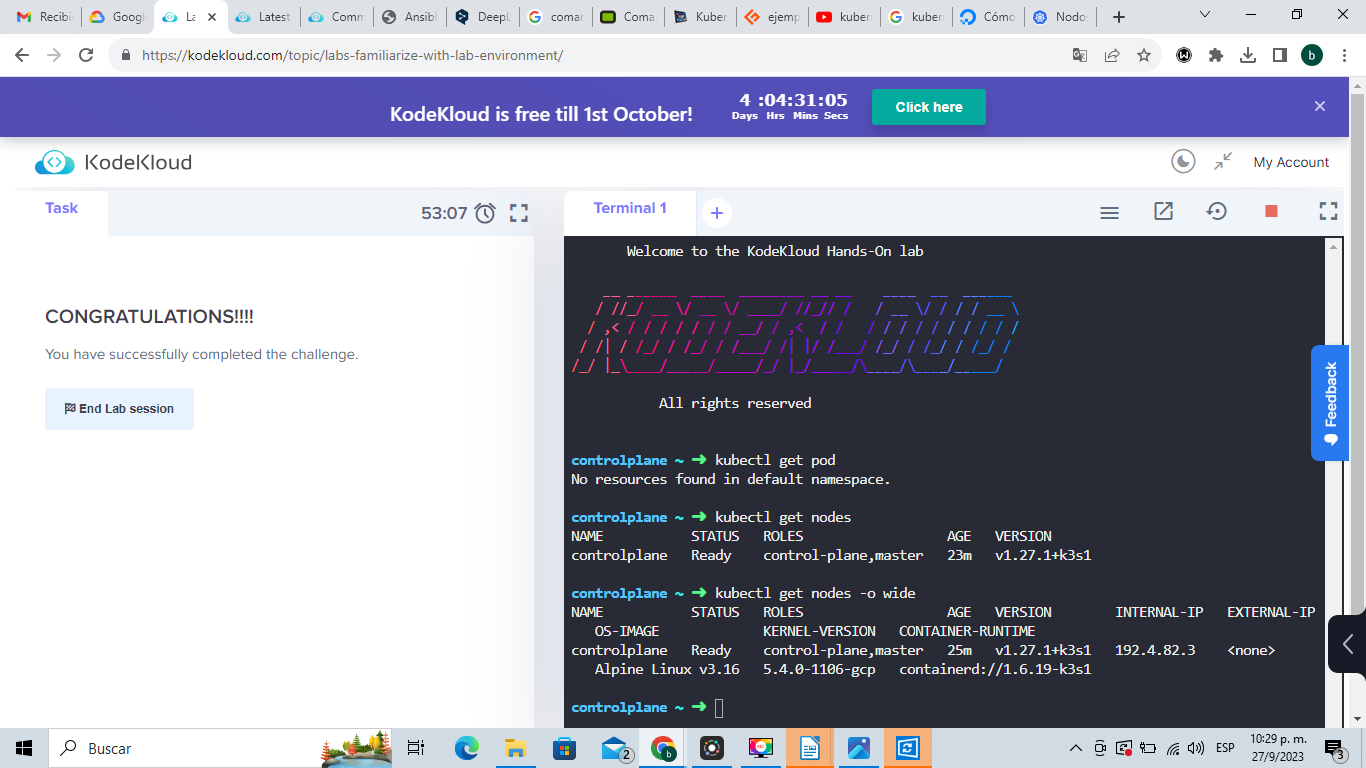
Laboratorio kubernetes

Cuál es el tipo y la versión del sistema operativo en el que se ejecutan los nodos Kubernetes?  
  
¿Cuántos nodos forman parte del clúster?

<https://kodekloud.com/community/c/kubernetes/6>

https://kodekloud.com/pages/community

c

COMANDOS

kubectl run nginx –image nginx

kubectl get pods

kubectl describe pod nginx

kube ctl get pods -o wide

**Create an NGINX Pod**

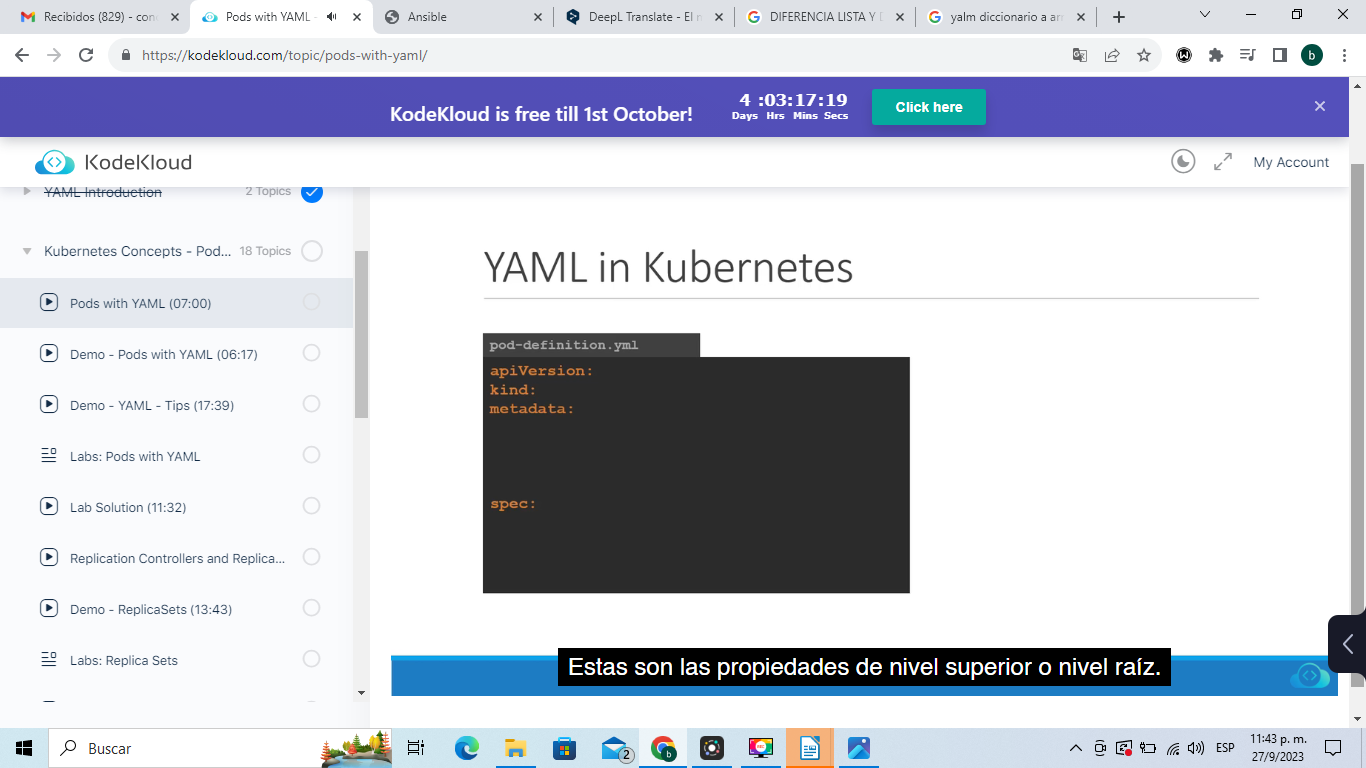
kubectl run nginx --image=nginx

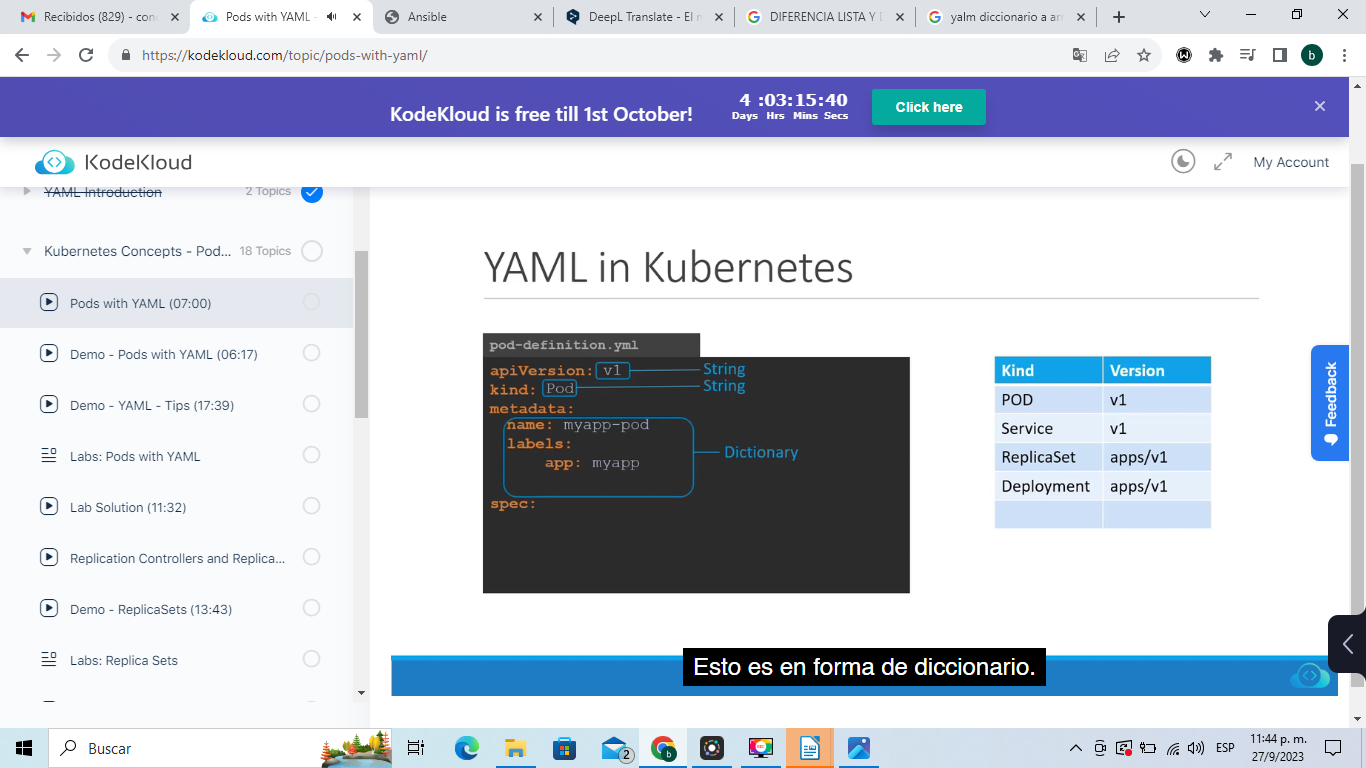
**A partir de la versión 1.18, kubectl run (sin ningún argumento como --generator ) creará un pod en lugar de un deployment.  
  
Para crear un despliegue utilizando un comando imperativo, utilice kubectl create:**

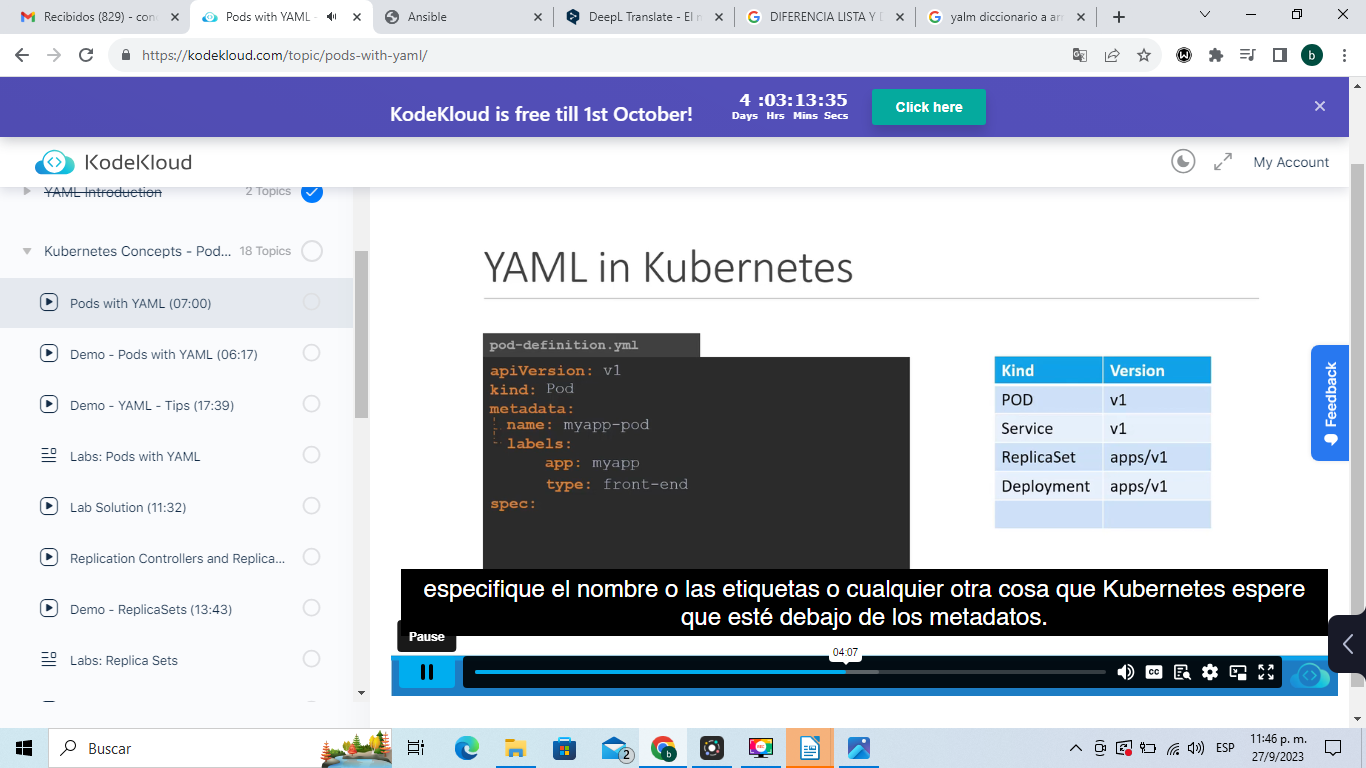
kubectl create deployment nginx --image=nginx

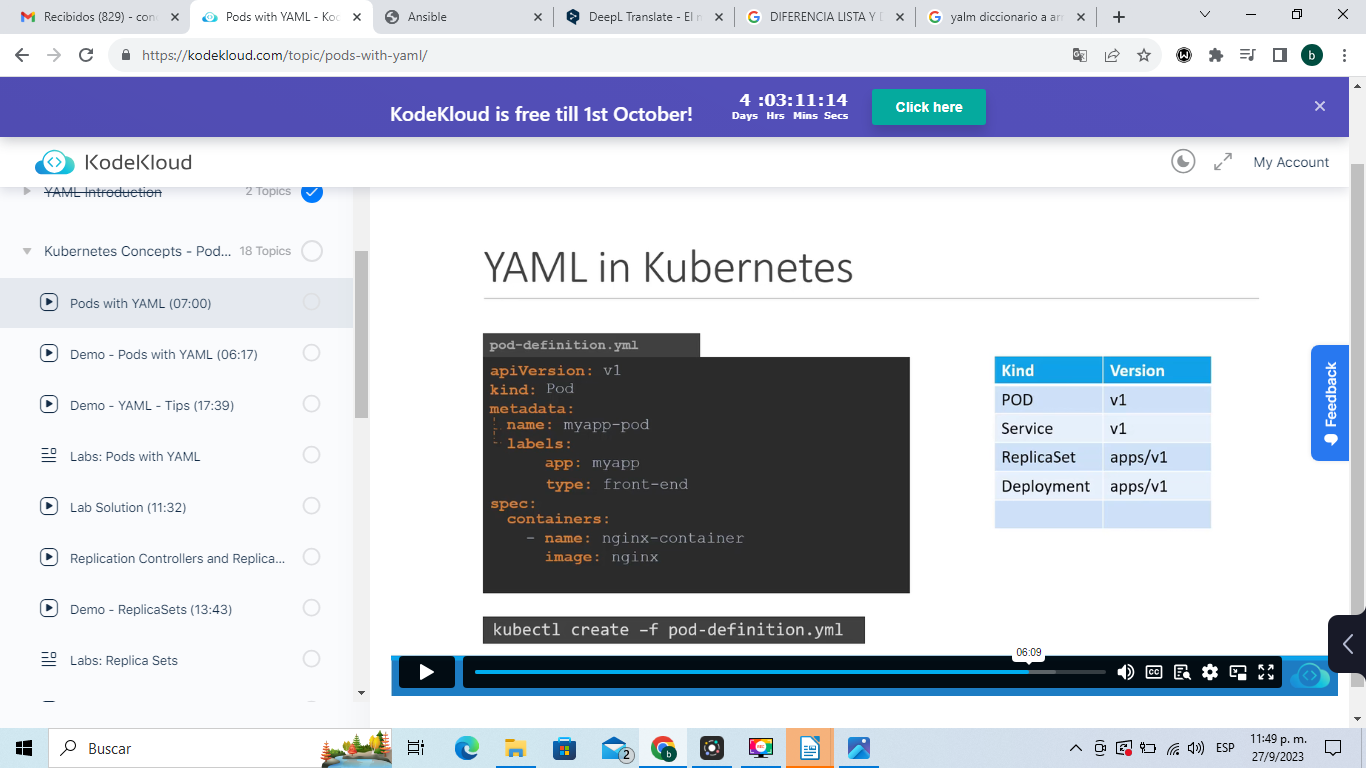
**INTRODUCCION A YALM**

**\*\***

**xx**

**x**

**xx**

**kubectl create -f pod-definition.yaml**

**kubectl apply -f pod-definition.pdf**

**Create a new pod with the nginx image.**

**kubectl run nginx –image=nginx**

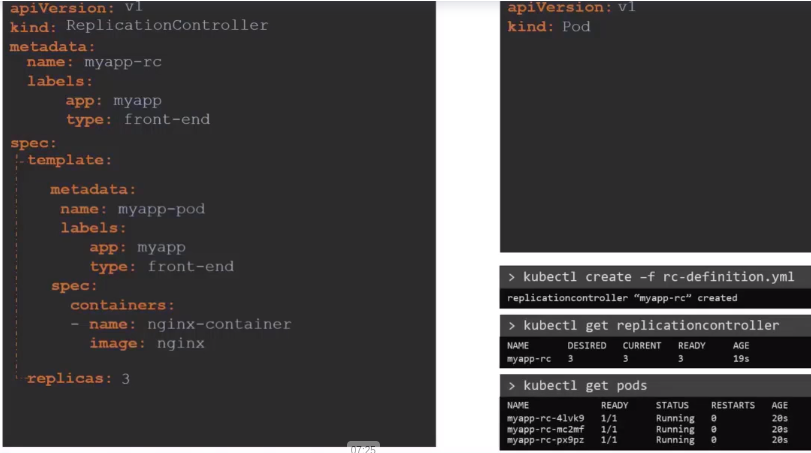
**detalle de un pod en especifico**

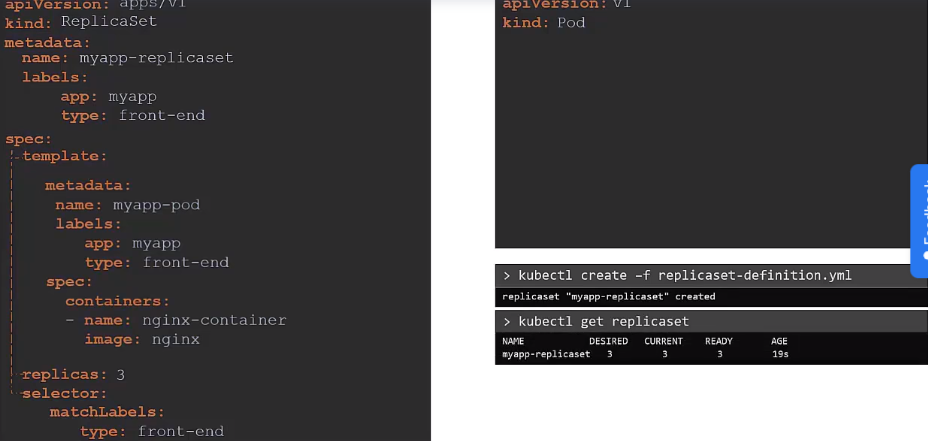
kubectl describe pod newpods-<id>

**Creando un archivo yaml**

kubectl run redis --image=redis123 --dry-run=client -o yaml > redis-definition.yaml

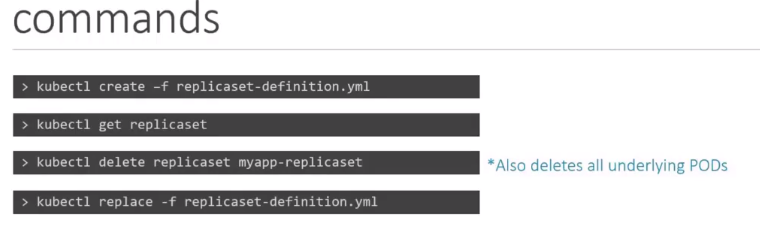
Replication controler

Replica set

d

Escalar

s

k

**comandos**

kubectl create -f archivo.yml

kubectl get replicaset archivo.yml

kubectl delete replicaset archivo.yml

kubectl replace -f archivo.yml

kubectl get rs

kubectl get replicaset

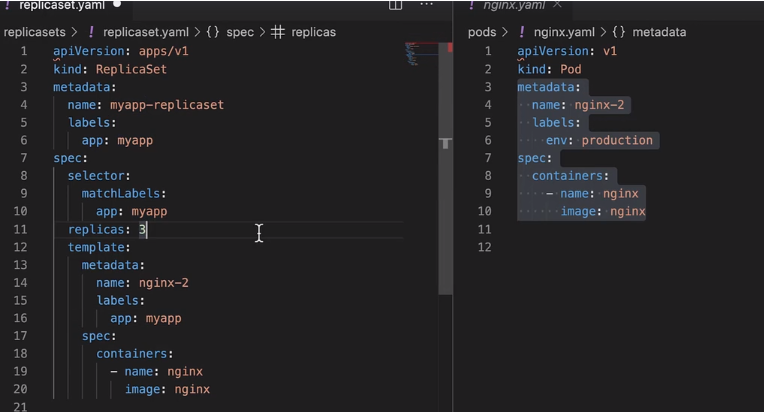
**Eliminar replicaset**

kubectl get replicaset

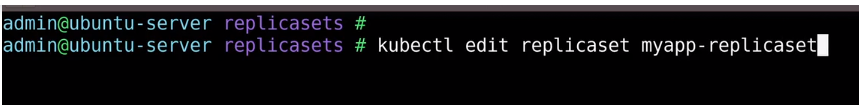
kubectl delete replicaset nombre

**Todo lo creado**

kubectl get all

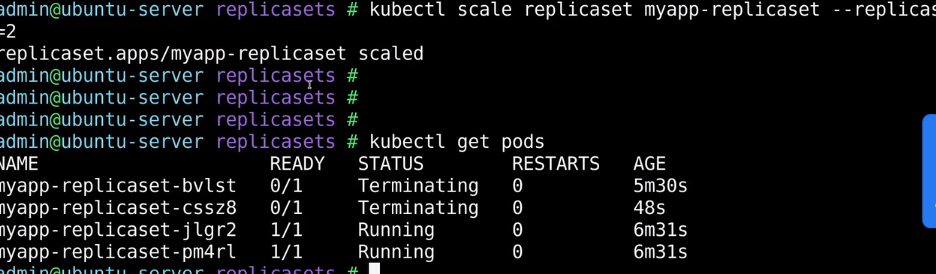
**Editar replicaset**

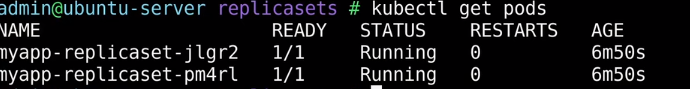
kubectl edit replicaset myapp-preplicaset

O tambien se puede hacer

kubectl scale replicaset myapp-replicaset – replicas=2

kubectl get pods



cc

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: replicaset-1

spec:

replicas: 2

selector:

matchLabels:

tier: frontend

template:

metadata:

labels:

tier: frontend

spec:

containers:

- name: nginx

image: nginx

**dos tipos de ReplicaSet**

piVersion: apps/v1

kind: ReplicaSet

metadata:

name: replicaset-2

spec:

replicas: 2

selector:

matchLabels:

tier: nginx

template:

metadata:

labels:

tier: nginx

spec:

containers:

- name: nginx

image: nginx

**Archivo de pruebas laboratorio funciona**

**apiVersion: apps/v1**

**kind: ReplicaSet**

**metadata:**

**creationTimestamp: "2023-09-29T01:35:24Z"**

**generation: 3**

**name: new-replica-set**

**namespace: default**

**resourceVersion: "2308"**

**uid: 51ec9961-5780-489f-a82b-12168b6e9381**

**spec:**

**replicas: 5**

**selector:**

**matchLabels:**

**name: busybox-pod**

**template:**

**metadata:**

**creationTimestamp: null**

**labels:**

**name: busybox-pod**

**spec:**

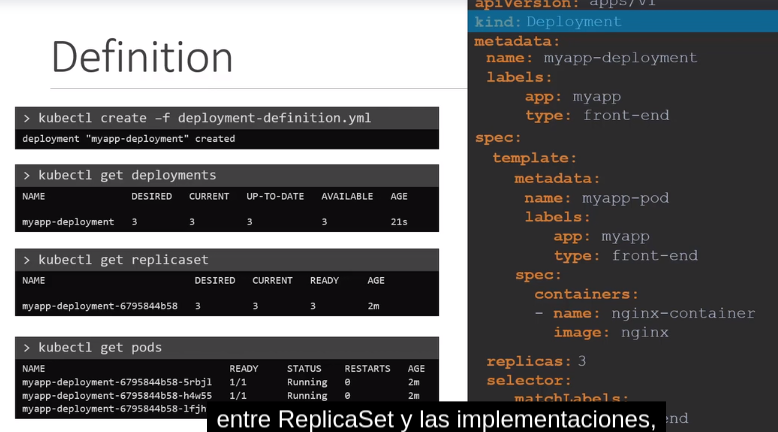
**containers:**

**- command:**

**- sh**

**- -c**

**DEPLOYMENT**



**Archivo ejemplo**

---

apiVersion: apps/v1

kind: deployment

metadata:

name: deployment-1

spec:

replicas: 2

selector:

matchLabels:

name: busybox-pod

template:

metadata:

labels:

name: busybox-pod

spec:

containers:

- name: busybox-container

image: busybox888

command:

- sh

- "-c"

- echo Hello Kubernetes! && sleep 3600

**otro ejemplo de deployment**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: httpd-frontend**

**spec:**

**replicas: 3**

**selector:**

**matchLabels:**

**name: httpd-frontend**

**template:**

**metadata:**

**labels:**

**name: httpd-frontend**

**spec:**

**containers:**

**- name: httpd-frontend**

**image: httpd:2.4-alpine**

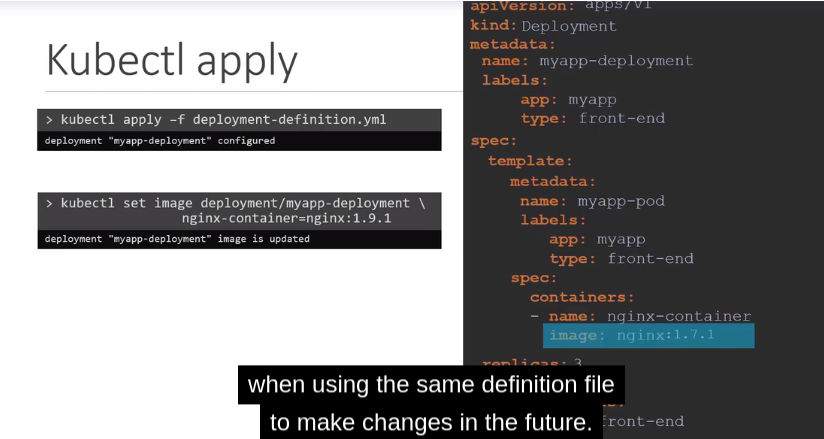
**command:**

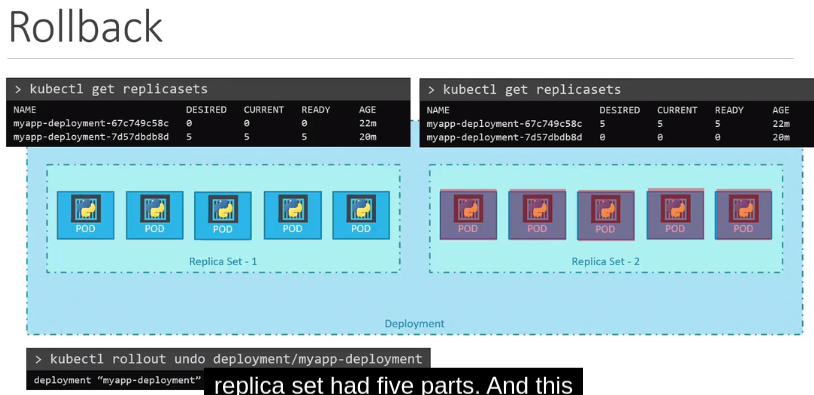
**- sh**

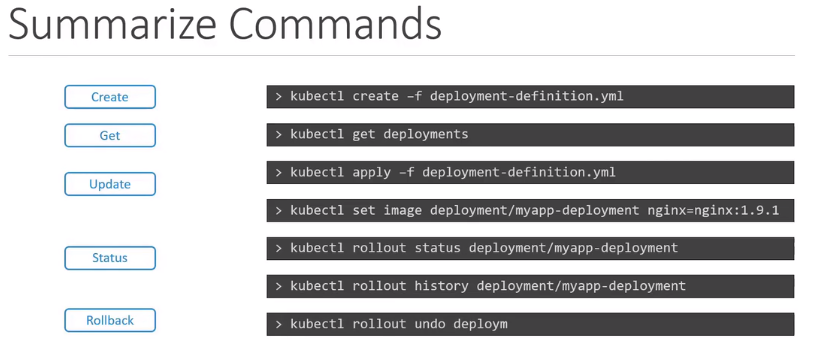
**- "-c"**

**- echo Hello Kubernetes! && sleep 3600**

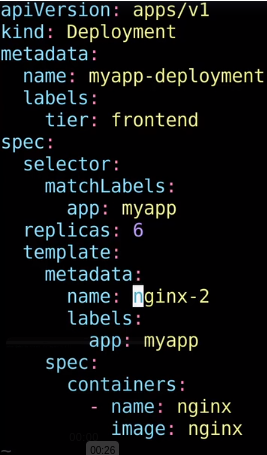
**update and rollback**



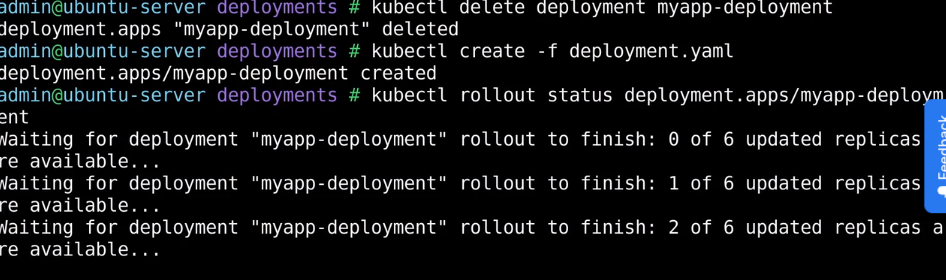


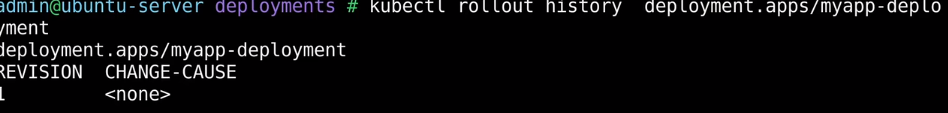
c

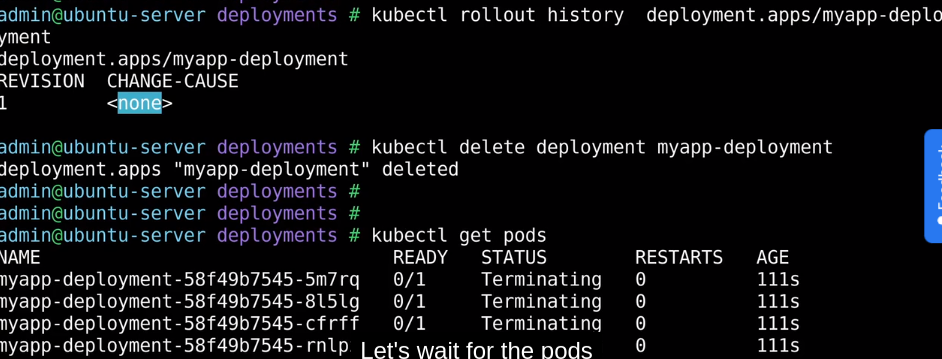
Ejemplos de roll

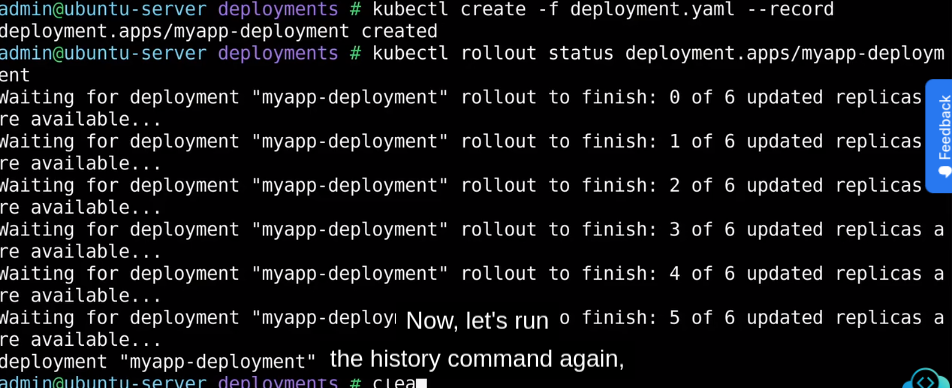


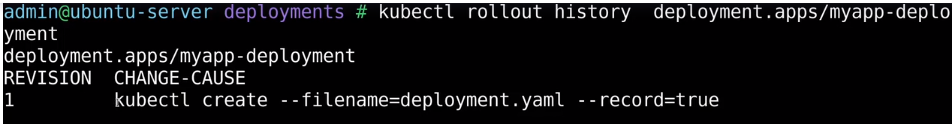






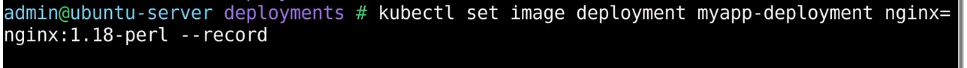


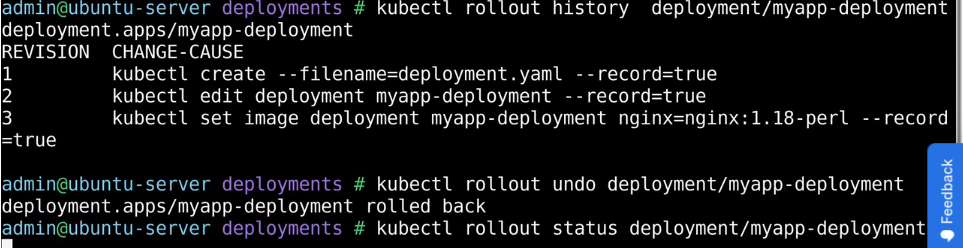


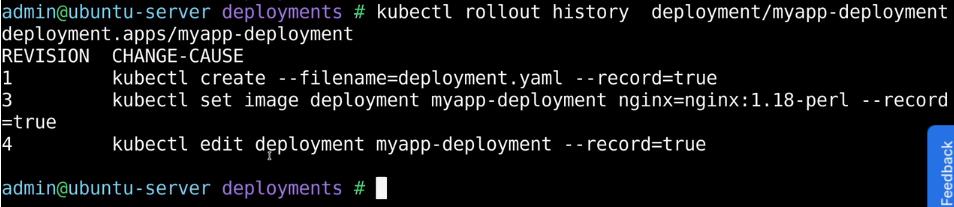




otro

cc

c

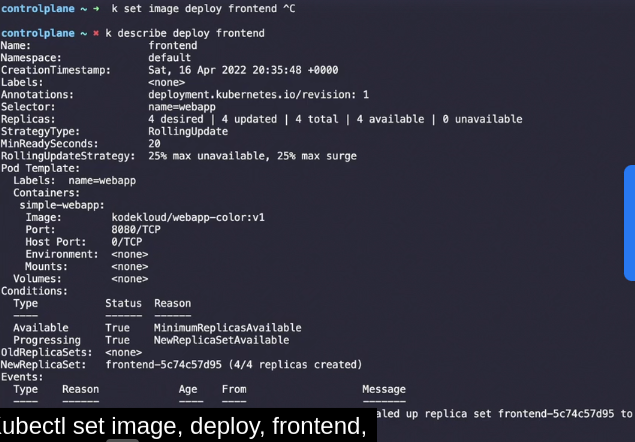
ccc

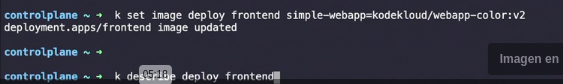
7

**Actualiar la imagen del deployment**

set image deploy fronted

describe deploy fronted

xxx

x

**Actualice la aplicación configurando la imagen en la implementación en kodekloud/webapp-color:v2**

k set image deploy frontend simple-webapp=kodekloud/webapp-color:v2

11

k edit deployment frontend

**type: recreate**

**k describe deployment frontend**

**12**

**k set image deploy frontend simple-webapp=kodekloud/webapp-color:v3**

**que es RollingUpdate y recreate de type**

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend

namespace: default

spec:

replicas: 4

selector:

matchLabels:

name: webapp

strategy:

type: Recreate

template:

metadata:

labels:

name: webapp

spec:

containers:

- image: kodekloud/webapp-color:v2

name: simple-webapp

ports:

- containerPort: 8080

protocol: TCP

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**annotations:**

**deployment.kubernetes.io/revision: "2"**

**creationTimestamp: "2023-09-29T12:25:29Z"**

**generation: 3**

**name: frontend**

**namespace: default**

**resourceVersion: "1325"**

**uid: 4448bc07-711c-4d9a-bc7a-ddde714f5736**

**spec:**

**minReadySeconds: 20**

**progressDeadlineSeconds: 600**

**replicas: 4**

**revisionHistoryLimit: 10**

**selector:**

**matchLabels:**

**name: webapp**

**strategy:**

**type: Recreate**

**template:**

**metadata:**

**creationTimestamp: null**

**labels:**

**name: webapp**

**spec:**

**containers:**

**- image: kodekloud/webapp-color:v2**

**imagePullPolicy: IfNotPresent**

**name: simple-webapp**

**ports:**

**- containerPort: 8080**

**protocol: TCP**

**resources: {}**

**terminationMessagePath: /dev/termination-log**

**terminationMessagePolicy: File**

**dnsPolicy: ClusterFirst**

**restartPolicy: Always**

**schedulerName: default-scheduler**

**securityContext: {}**

**terminationGracePeriodSeconds: 30**

**status:**

**availableReplicas: 4**

**conditions:**

**- lastTransitionTime: "2023-09-29T12:25:56Z"**

**lastUpdateTime: "2023-09-29T12:25:56Z"**

**message: Deployment has minimum availability.**

**reason: MinimumReplicasAvailable**

**status: "True"**

**type: Available**

**- lastTransitionTime: "2023-09-29T12:25:29Z"**

**lastUpdateTime: "2023-09-29T12:34:52Z"**

**message: ReplicaSet "frontend-75d9b4f9b9" has successfully progressed.**

**reason: NewReplicaSetAvailable**

**status: "True"**

**type: Progressing**

**observedGeneration: 3**

**readyReplicas: 4**

**replicas: 4**

**Como modifoco todo**

**for i in {1..35}; do**

**kubectl exec --namespace=kube-public curl -- sh -c 'test=`wget -qO- -T 2 http:>**

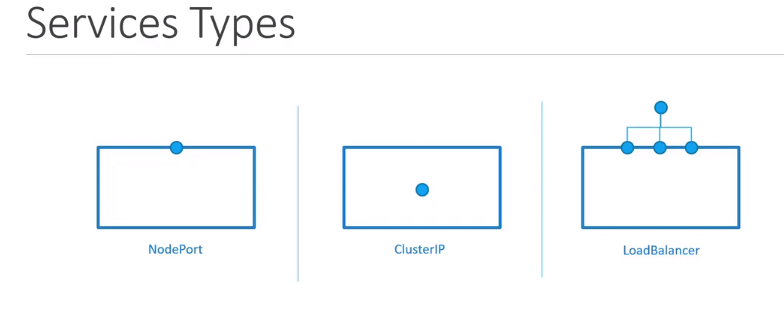
**echo ""**

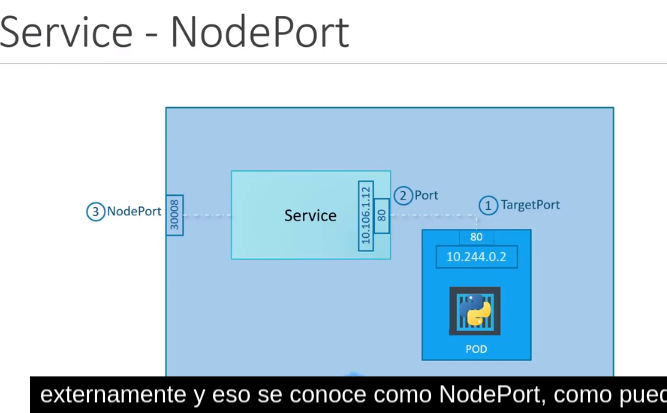
**done**

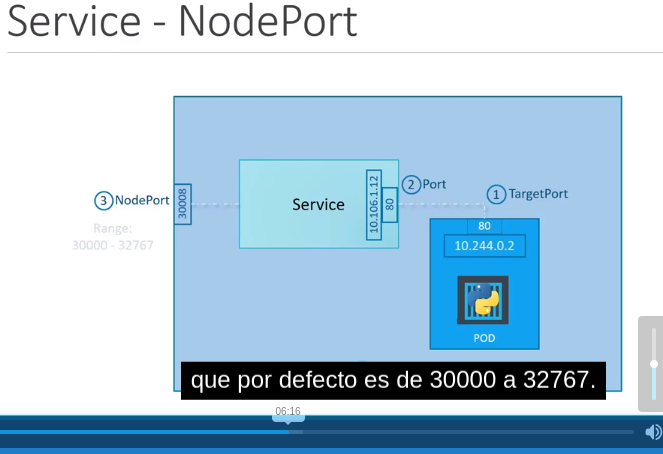
**x**

**Tipo de servicios**

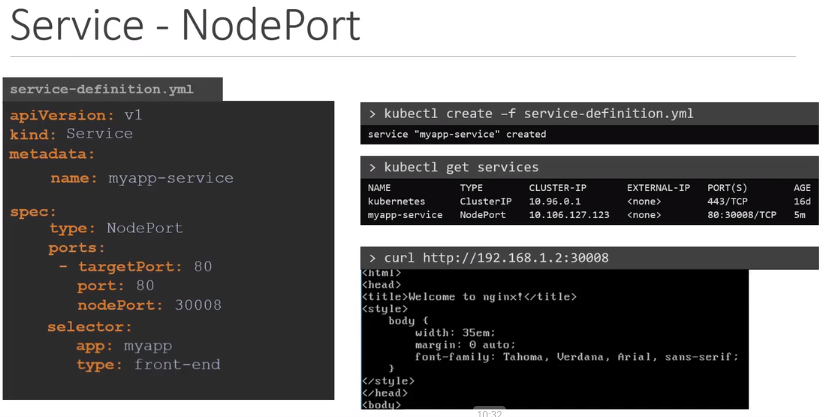
**x**

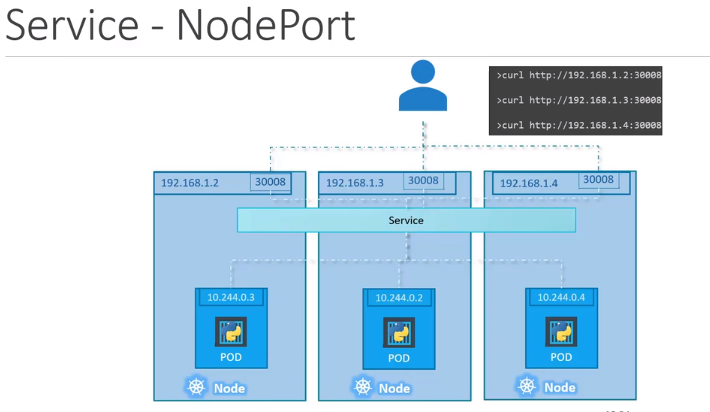
****

****

****

**El puertopor defect va desde 30000 a 32767**

**x**

**c**

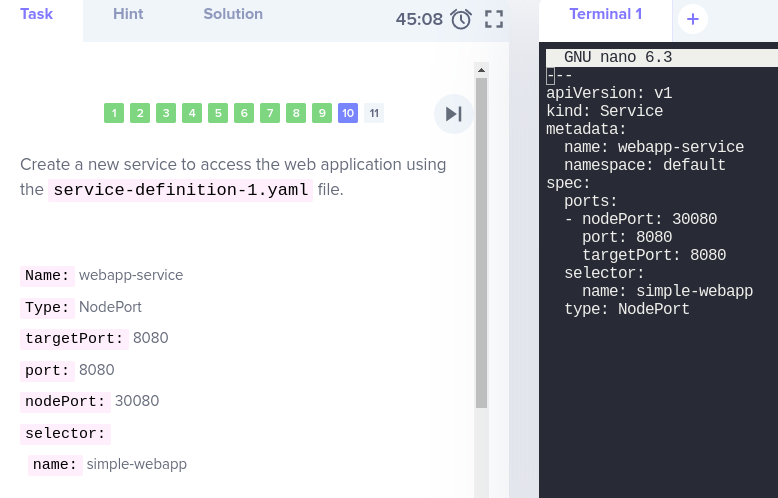
**Servicio ClusterIp**



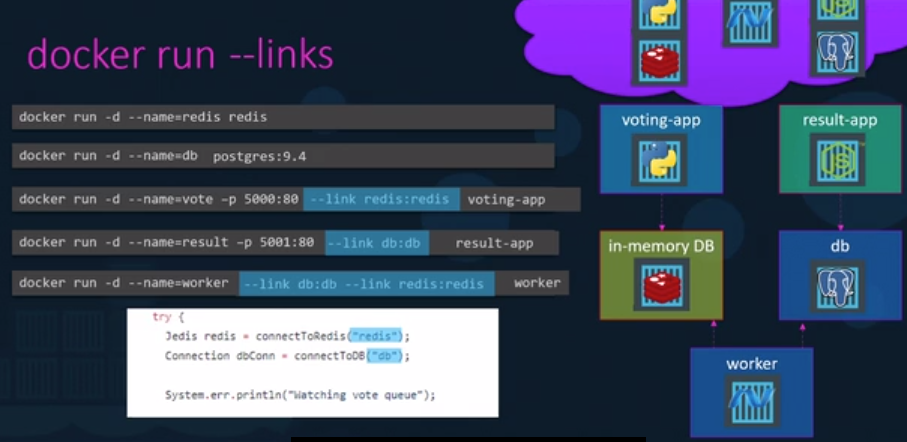
**cc**

**SERVICCE LOADBALANCE**

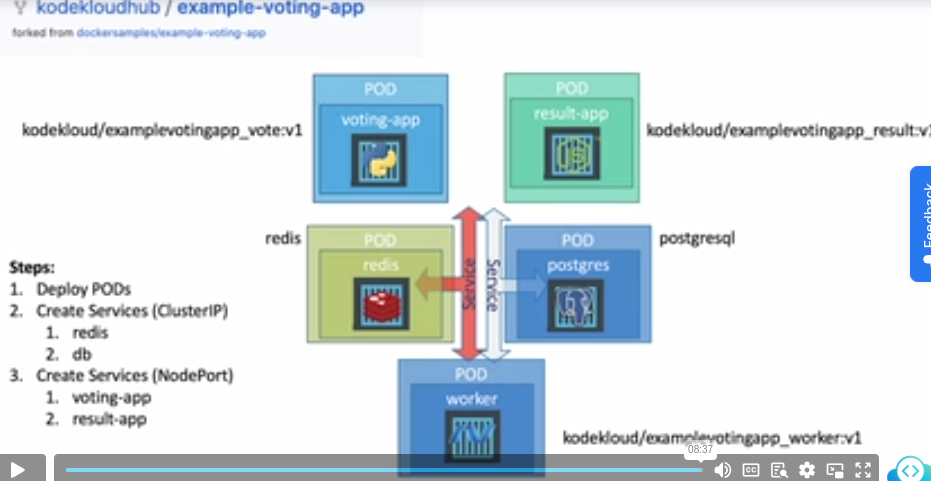
**Ejercicio**

**x**

**Microservicios arquitectura**

**x**

# eploying voting app on Kubernetes

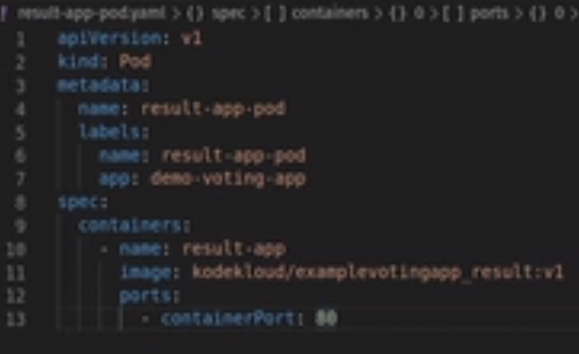
**X**

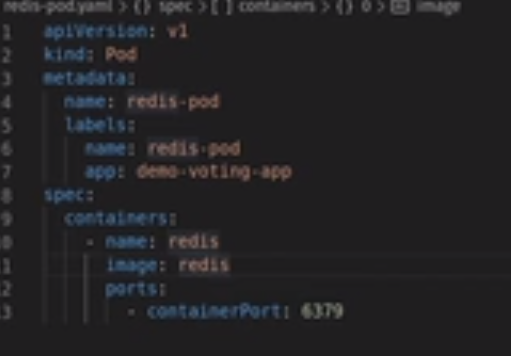
**Ejercicio demo**

**voting-app-pod.yaml**

**z**

**Archivo result-app-pod.yaml**

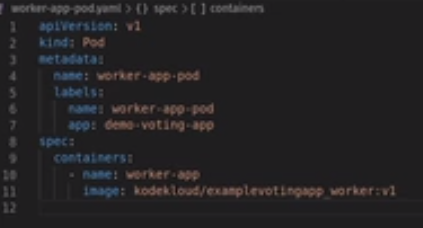
**archivo redis-pod.yaml**

****

**archivo postgres-pod.yaml**

****

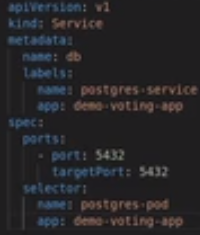
**archivo worked-app-pod.yaml**

****

**archivo redis-service.yaml**

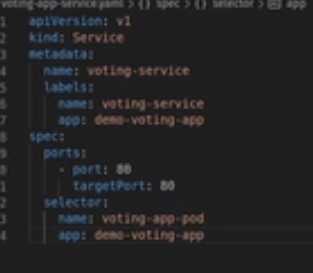
****

**archivo postgresql-servicio.yaml**

****

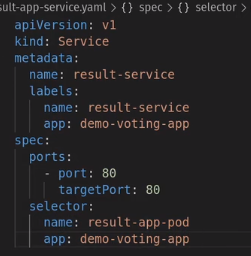
**servicio de votación**

**result-app-service.yaml**

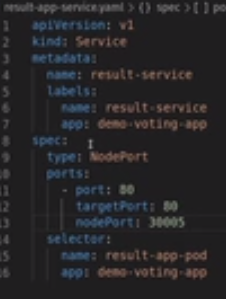
****

**Servicio app**

**result-app-service.yaml**

****

**voting-app-service.yaml**

****

**Correrlos**

k create f voting-app-pod.yaml

k create f voting-app-service.yaml

k get pod, svc

-para poder obtener la ruta-

minicube service voting-service --url

k create f redys-pod.yaml

k create f redys-service.yaml

k get pod, svc

k create f postgres-pod.yaml

k create f postgres-service.yaml

k get pod, svc

k create f worker-pod.yaml

k create f resul-app-pod.yaml

k create f postgres-service.yaml

k get pod, svc

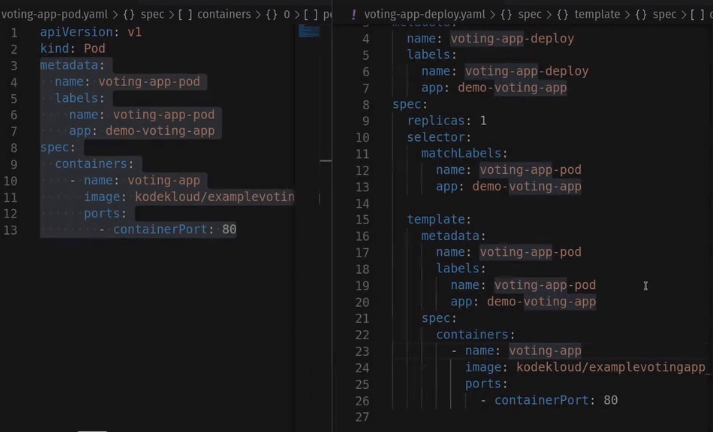
minikube service voting-service –url

minikube service result-service –url

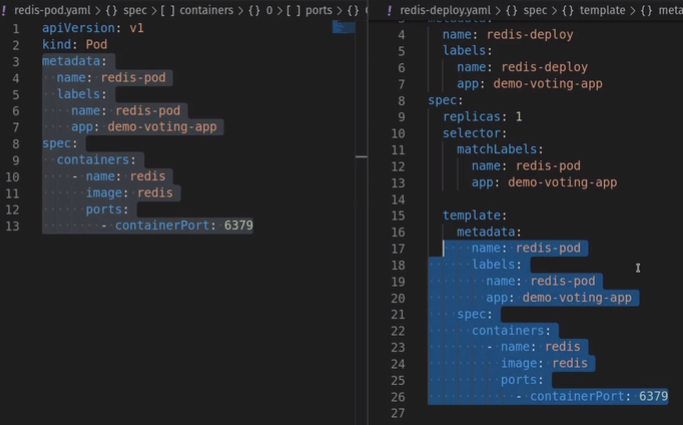
ya puede verificar por el navegador con las dos url que muestra según los 2 comandos anteriores

**el mismo ejercicio con replicaset que es mejor para que haya replicas**

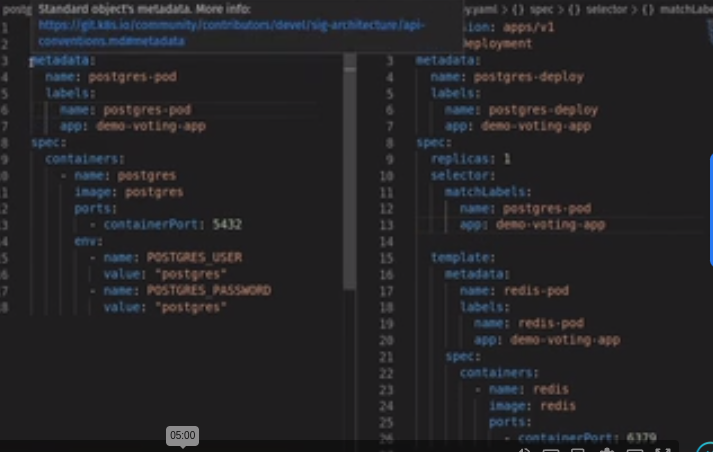
voting-app-deployment.yaml

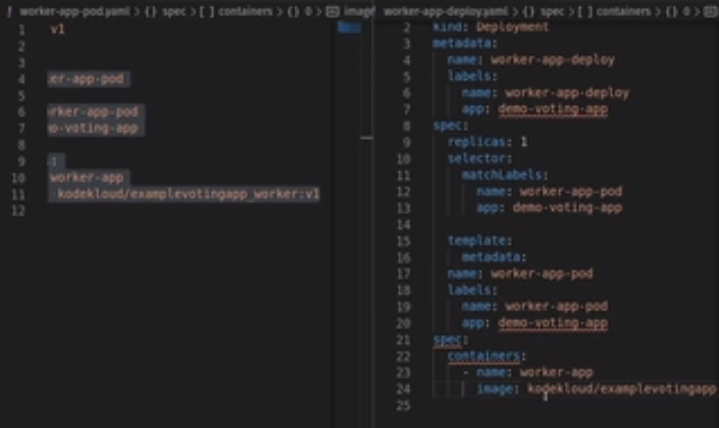


c

d

postgres

Al final colocar los valores env: que no se ven

c

nos aseguramos que no haya nada activo

comenzamos a crear los deployment

k create -f voting-app-deploy.yaml

k create -f redis-service.yaml

k create -f postgres-deploy.yaml

k create -f postgres-services.yaml

verificamos que tenemos ahasta ahora

k get deployment

k get pods

k create -f worked-app-deploy.yaml

k get pod, svc

k create -f result-app-deploy.yaml

k create -f result-services.yaml

minikube service voting-service –url

minikube service result-service --url

link del ejericio

https://github.com/kodekloudhub/example-voting-app/tree/master/k8s-specifications

**instalacion de minikube**