

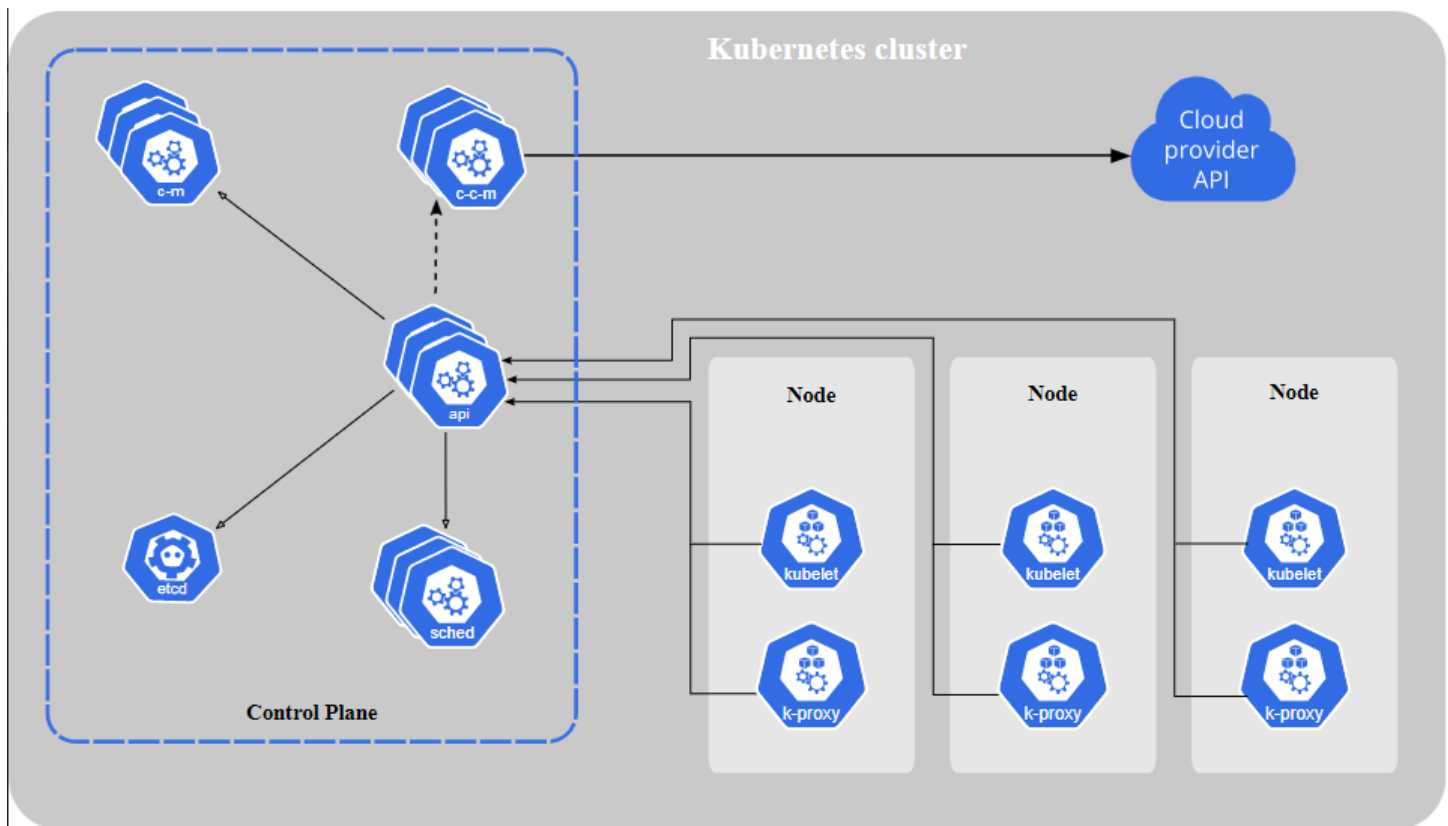
TALLER 10: DevOps Kubernetes

Ing. Luis Felipe Narvaez Gomez. E-mail: Luis.narvaez@usantoto.edu.co. Cod: 2312660. Facultad de Ingeniería de Sistemas.

Para poder desarrollar este taller se debe tener instalado el software de DOCKER, el software de MINIKUBE y tener también instalado en a PC o la VM en la que se esté trabajando el software de KUBECTL. Kubernetes es una tecnología que permite administrar los contenedores, entre esta administración está la opción de crear los manifiestos de tipo deployment, los cuales permiten crear PODS con una cantidad especifica de contenedores y replicas, haciendo que cada cluster (API) puedan cumplir con las ordenas dadas en su funcionamiento.

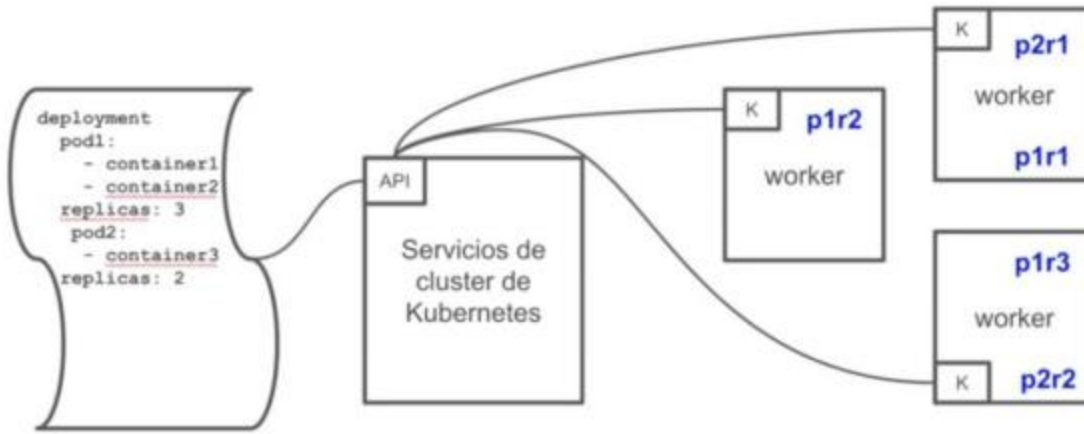
El espacio donde se correrán nuestras aplicaciones y clusters serán los WORKERS. Por parte de los contenedores, estos se manejarán por los SCHEDULER, los cuales se encargan de mover los containers de un lugar a otro, conectarlos con la API y con los respectivos workers a través de un agente llamado kubelet que corre en cada uno de los workers.

Los kubelets por su parte son un servicio de kubernetes que permiten conectar todos los workers y sus servicios de kubernetes entre sí, permitiendo que en caso de que algún worker muera, este sea reemplazado rápidamente.



Sus componentes son:

1. Control plane, Servidores de kubernetes
2. Nodos, Cada uno correrá kubelet (agente de kubernetes) y el servicio de kube - proxy
3. (recibe el tráfico y mandarlo a los pods que requieran ese tráfico)
4. Scheduler
5. Cloud Controller Manager, Se conecta a la API de tu proveedor de cloud
6. Etcd, Base de datos que te permite guardar el estado de tu cluster de kubernetes



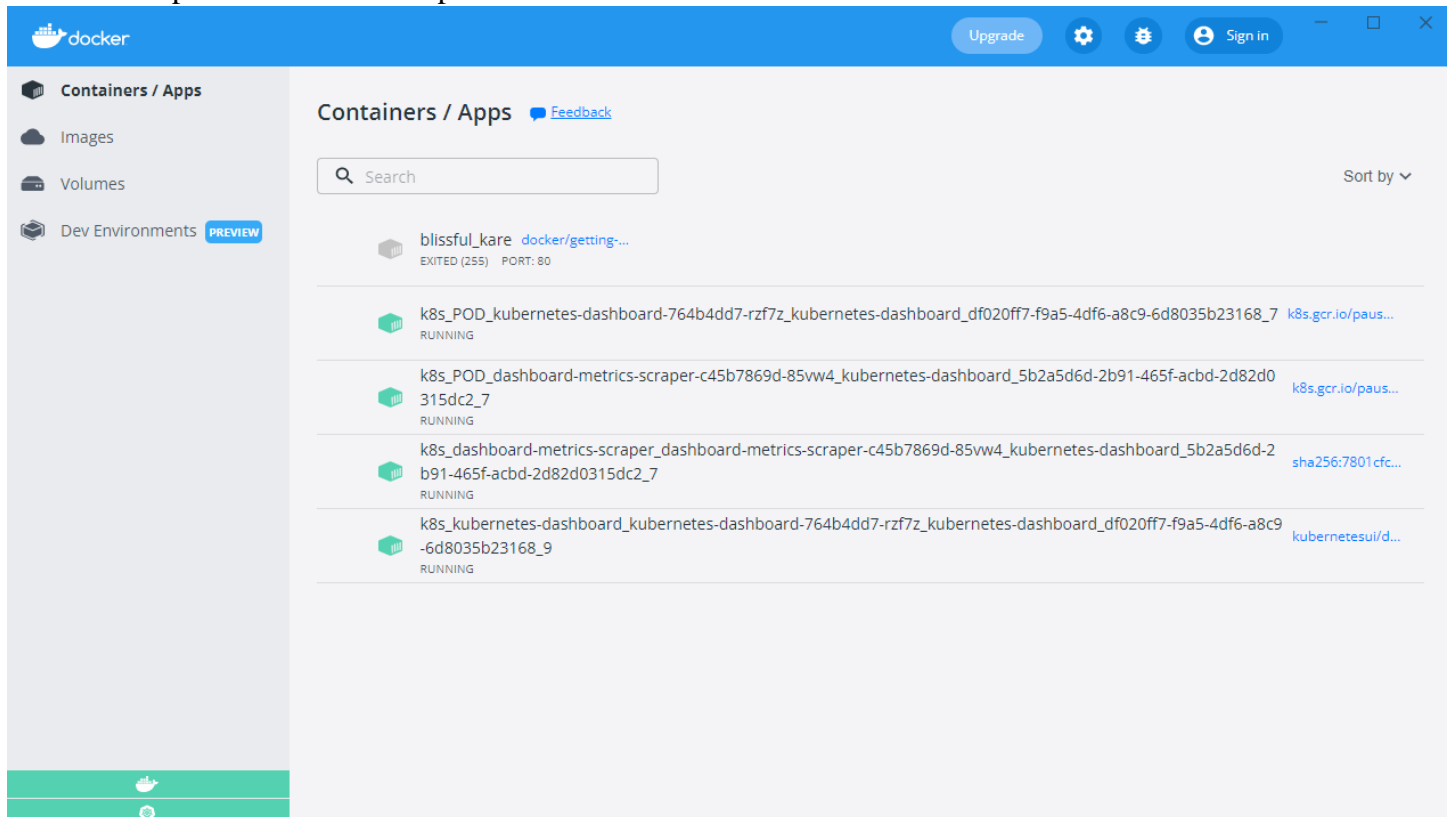
Abrimos la consola de comandos de Windows CMD como administradores, en caso de estar trabajando en Linux, será una terminal como usuario Root. En ella indicaremos conocer la versión de Kubectl, esta herramienta nos permitirá interactuar con el cluster.

```

C:\Windows\system32>kubectl version --client=true
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.6", GitCommit:"ad3338546da947756e8a88aa6822e9c1
1e7eac22", GitTreeState:"clean", BuildDate:"2022-04-14T08:49:13Z", GoVersion:"go1.17.9", Compiler:"gc", Platform:"win
dows/amd64"}
C:\Windows\system32>

```

Abrimos la aplicación de Docker para escritorio.



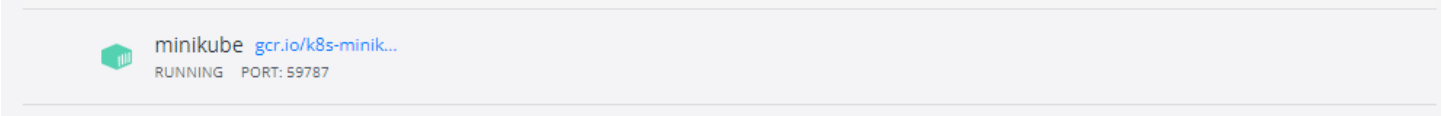
Ahora iniciamos Minikube en la CMD.

```
C:\Windows\system32>minikube start
* minikube v1.25.2 en Microsoft Windows 10 Home Single Language 10.0.19044 Build 19044
* Controlador docker seleccionado automáticamente. Otras opciones: hyperv, virtualbox, ssh
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Creando docker container (CPUs=2, Memory=4000MB) .../ E0509 09:57:13.886773 26520 kic.go:267] icacIs failed apply
ing permissions - err - [%!s(<nil>)], output - [archivo procesado: C:\Users\Ruiso Local Pc\minikube\machines\minikub
e\id_rsa
Se procesaron correctamente 1 archivos; error al procesar 0 archivos]

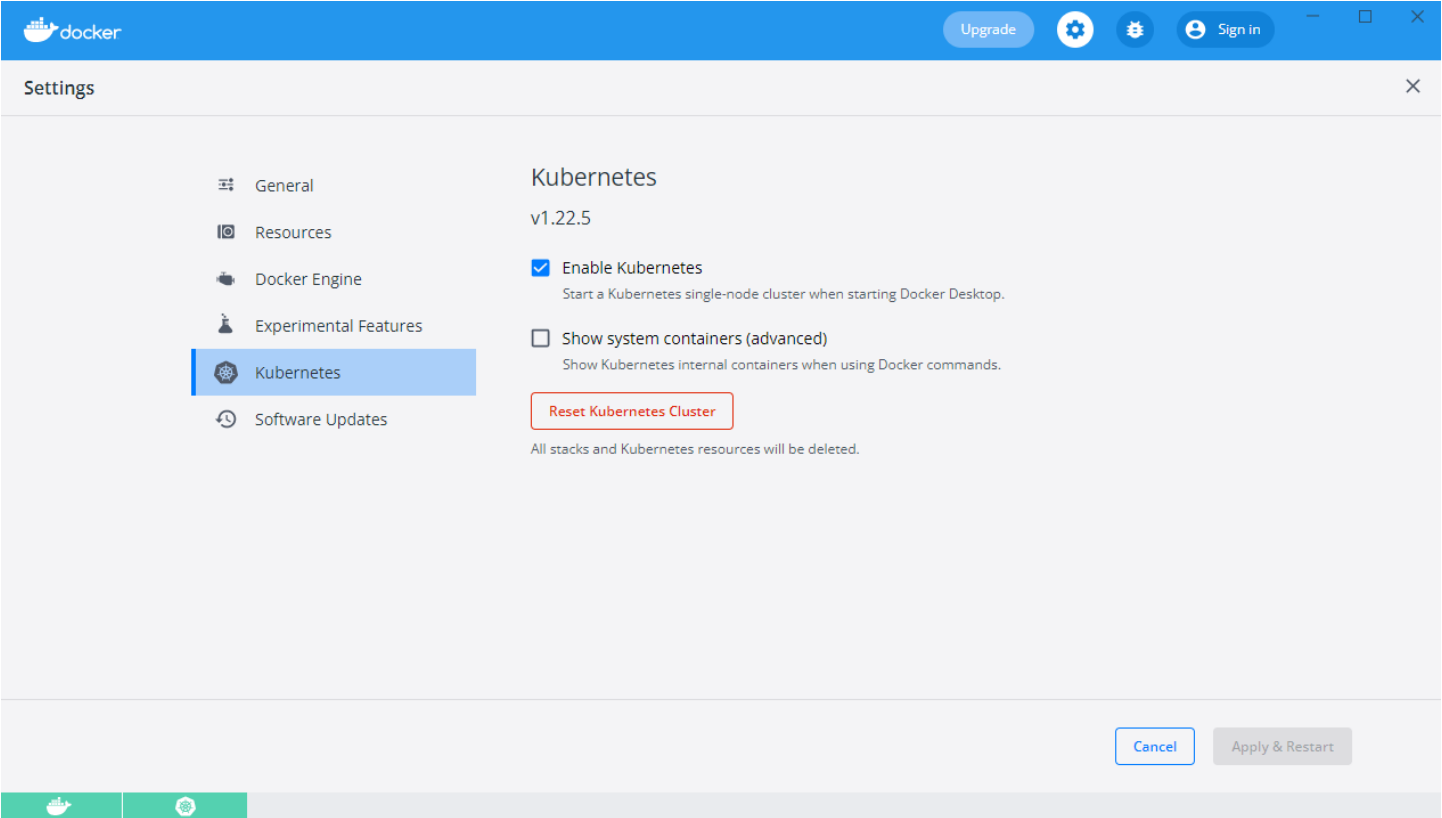
* Preparando Kubernetes v1.23.3 en Docker 20.10.12...
- kubelet.housekeeping-interval=5m
- Generando certificados y llaves
- Iniciando plano de control
- Configurando reglas RBAC...
* Verifying Kubernetes components...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Complementos habilitados: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

C:\Windows\system32>
```

Verificamos que dentro de Docker este corriendo la aplicación de Minikube.



Vamos a la sección de Settings y luego a la sección de Kubernetes, en ella verificamos que este habilitado el [Kubernetes].



Verificamos los nodos que estén en nuestro cluster de Kubernetes.

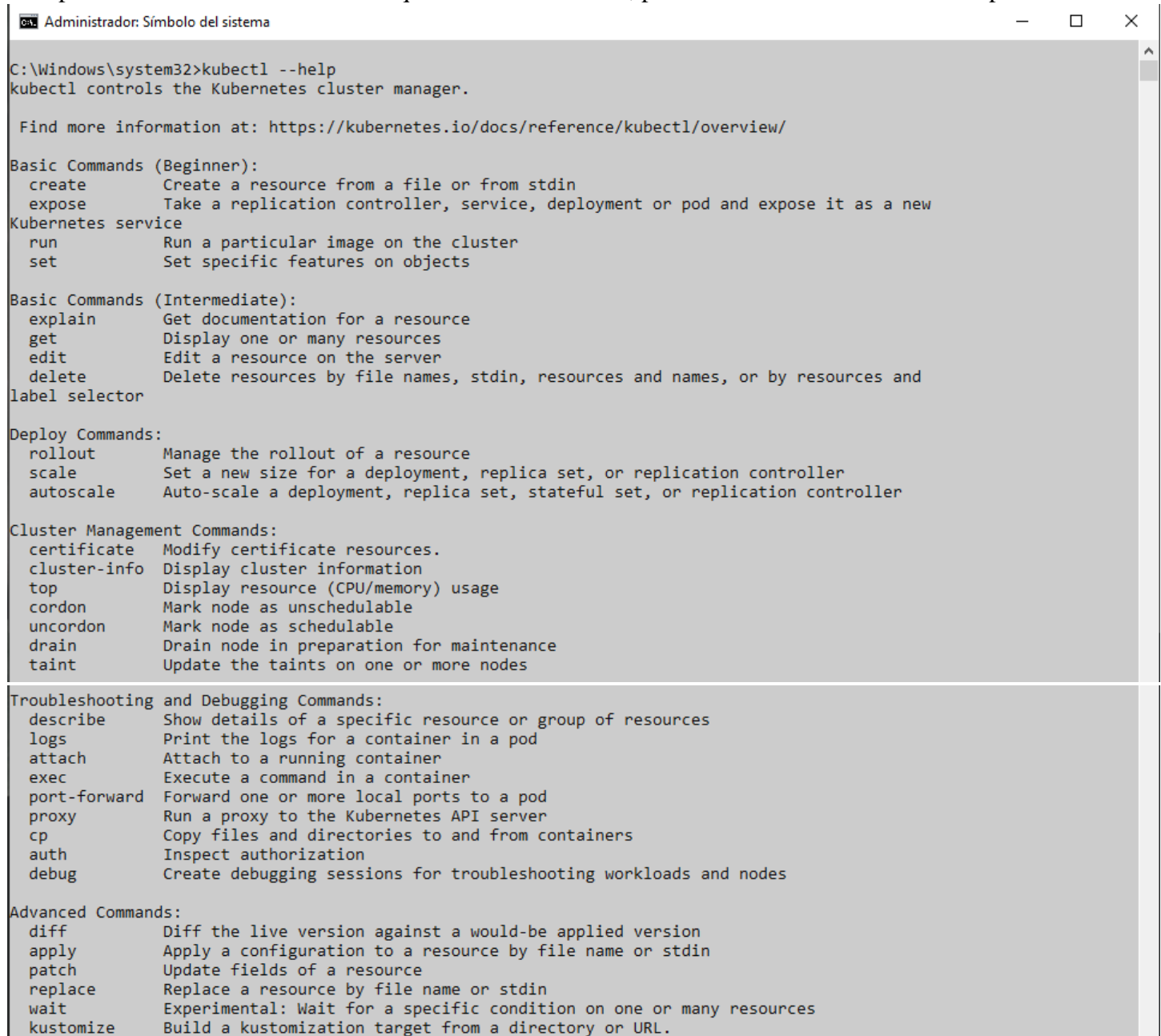
```
C:\Windows\system32>kubectl get nodes
NAME        STATUS    ROLES                  AGE    VERSION
minikube    Ready     control-plane,master   5m6s   v1.23.3

C:\Windows\system32>
```

En este caso KUBECTL es nuestro cliente de Kubernetes. Este nos va a permitir conectarnos a los diferentes clusters que tengamos y poder interactuar con ellos en base a comandos específicos. Algunos de estos comandos pueden ser:

1. Get → obtiene los recursos de cluster.
2. EDIT → edita un recurso
3. DELETE → Elimina un recurso
4. APPLY → aplica algún manifiesto a un cluster en kubernetes
5. EXEC → ejecuta un comando dentro de un contenedor
6. LOGS-CP → permite la copia de archivos de la VM al contenedor
7. CORDON-UNCORDON-DRAIN → para el manejo de nodos.

Para poder ver todas las herramientas que ofrece KUBECTL, podemos utilizar el comando –help.



```
C:\Windows\system32>kubectl --help
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create      Create a resource from a file or from stdin
  expose      Take a replication controller, service, deployment or pod and expose it as a new
Kubernetes service
  run         Run a particular image on the cluster
  set         Set specific features on objects

Basic Commands (Intermediate):
  explain     Get documentation for a resource
  get         Display one or many resources
  edit        Edit a resource on the server
  delete      Delete resources by file names, stdin, resources and names, or by resources and
label selector

Deploy Commands:
  rollout     Manage the rollout of a resource
  scale       Set a new size for a deployment, replica set, or replication controller
  autoscale   Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:
  certificate Modify certificate resources.
  cluster-info Display cluster information
  top         Display resource (CPU/memory) usage
  cordon      Mark node as unschedulable
  uncordon    Mark node as schedulable
  drain       Drain node in preparation for maintenance
  taint       Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
  describe   Show details of a specific resource or group of resources
  logs       Print the logs for a container in a pod
  attach      Attach to a running container
  exec        Execute a command in a container
  port-forward Forward one or more local ports to a pod
  proxy       Run a proxy to the Kubernetes API server
  cp          Copy files and directories to and from containers
  auth        Inspect authorization
  debug       Create debugging sessions for troubleshooting workloads and nodes

Advanced Commands:
  diff        Diff the live version against a would-be applied version
  apply       Apply a configuration to a resource by file name or stdin
  patch       Update fields of a resource
  replace     Replace a resource by file name or stdin
  wait        Experimental: Wait for a specific condition on one or many resources
  kustomize   Build a kustomization target from a directory or URL.
```

```

Settings Commands:
  label      Update the labels on a resource
  annotate   Update the annotations on a resource
  completion Output shell completion code for the specified shell (bash, zsh or fish)

Other Commands:
  alpha      Commands for features in alpha
  api-resources Print the supported API resources on the server
  api-versions Print the supported API versions on the server, in the form of "group/version"
  config     Modify kubeconfig files

plugin      Provides utilities for interacting with plugins
version     Print the client and server version information

Usage:
  kubectl [flags] [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all commands).

C:\Windows\system32>

```

Ahora veremos los archivos de configuración del cluster.

```

C:\Windows\system32>kubectl config get-contexts
CURRENT  NAME           CLUSTER          AUTHINFO          NAMESPACE
*        minikube       minikube         minikube          default
C:\Windows\system32>

```

Para ver los recursos de kubernetes, los NameSpaces, estos son una división lógica del cluster expresado en otras palabras que permite separar el tráfico del cluster de kubernetes.

```

C:\Windows\system32>kubectl get ns
NAME                STATUS   AGE
default             Active   17m
kube-node-lease     Active   17m
kube-public         Active   17m
kube-system         Active   17m
C:\Windows\system32>

```

Otro recurso de Kubectl, Pod, es un set para los contenedores que puede estar basado en uno o más contenedores, es muy probable que la mayoría de las veces tus pods corran un solo contenedor. Podemos utilizar dos versiones para listar la información de los PODS.

```

C:\Windows\system32>kubectl -n kube-system get pods
NAME                                READY   STATUS    RESTARTS   AGE
coredns-64897985d-hgk6r            1/1     Running   0           22m
etcd-minikube                      1/1     Running   0           22m
kube-apiserver-minikube             1/1     Running   0           22m
kube-controller-manager-minikube    1/1     Running   0           22m
kube-proxy-6mmm2                   1/1     Running   0           22m
kube-scheduler-minikube             1/1     Running   0           22m
storage-provisioner                 1/1     Running   2 (22m ago) 22m
C:\Windows\system32>

C:\Windows\system32>kubectl -n kube-system get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE       NOMINATED NODE   READINESS GATES
coredns-64897985d-hgk6r            1/1     Running   0           23m   172.17.0.2     minikube   <none>           <none>
etcd-minikube                      1/1     Running   0           23m   192.168.49.2   minikube   <none>           <none>
kube-apiserver-minikube             1/1     Running   0           23m   192.168.49.2   minikube   <none>           <none>
kube-controller-manager-minikube    1/1     Running   0           23m   192.168.49.2   minikube   <none>           <none>
kube-proxy-6mmm2                   1/1     Running   0           23m   192.168.49.2   minikube   <none>           <none>
kube-scheduler-minikube             1/1     Running   0           23m   192.168.49.2   minikube   <none>           <none>
storage-provisioner                 1/1     Running   2 (23m ago) 23m   192.168.49.2   minikube   <none>           <none>
C:\Windows\system32>

```

Teóricamente por la naturaleza de Kubernetes, cuando eliminamos un POD este se generara nuevamente.

```
Administrator: Símbolo del sistema

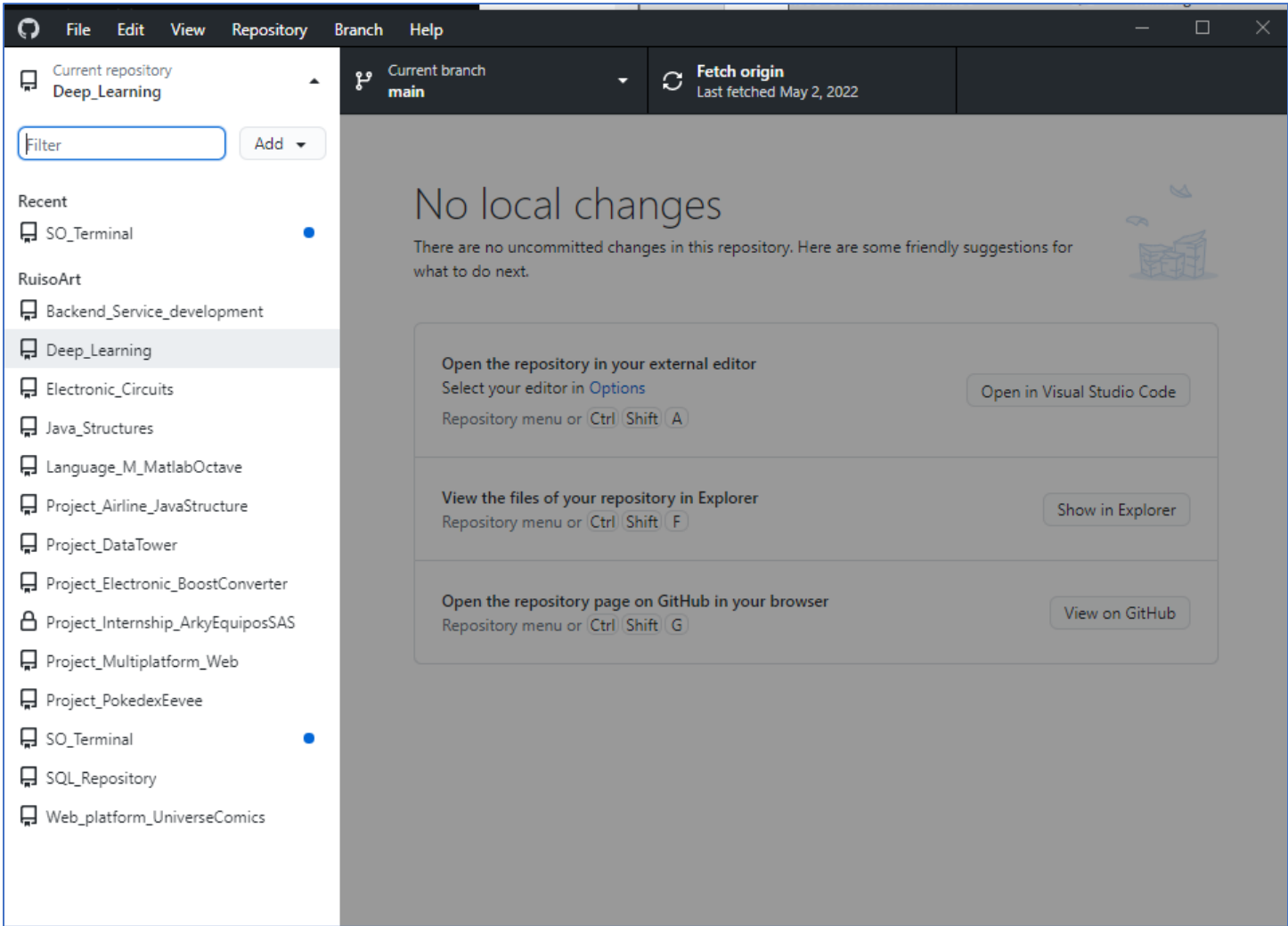
C:\Windows\system32>kubectl -n kube-system get pods
NAME                                READY   STATUS    RESTARTS   AGE
coredns-64897985d-hgk6r            1/1     Running   0           26m
etcd-minikube                      1/1     Running   0           27m
kube-apiserver-minikube            1/1     Running   0           27m
kube-controller-manager-minikube   1/1     Running   0           27m
kube-proxy-6mmm2                   1/1     Running   0           26m
kube-scheduler-minikube            1/1     Running   0           27m
storage-provisioner                1/1     Running   2 (26m ago) 27m

C:\Windows\system32>kubectl -n kube-system delete pod kube-proxy-6mmm2
pod "kube-proxy-6mmm2" deleted

C:\Windows\system32>kubectl -n kube-system get pods
NAME                                READY   STATUS    RESTARTS   AGE
coredns-64897985d-hgk6r            1/1     Running   0           28m
etcd-minikube                      1/1     Running   0           28m
kube-apiserver-minikube            1/1     Running   0           28m
kube-controller-manager-minikube   1/1     Running   0           28m
kube-proxy-4kgsh                   1/1     Running   0            6s
kube-scheduler-minikube            1/1     Running   0           28m
storage-provisioner                1/1     Running   2 (27m ago) 28m

C:\Windows\system32>
```

Como observamos, a pesar de que eliminamos el POD de proxy, este se regenero nuevamente al momento. Ahora vamos a hacer pruebas basándonos en un repositorio de github. En este paso se recomienda tener descargado alguna versión u Software de Git para PC anclado a Github o GITHUB de escritorio para llevar a cabo el CLONE.



There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Clone a repository

GitHub.com

GitHub Enterprise

URL

Repository URL or GitHub username and repository
(hubot/cool-repo)

Local path


Choose...

Clone

Cancel

File Edit View Repository Branch Help

Current repository
peladonerd

 Cloning peladonerd

Receiving objects: 5% (96/1898), 780.00 KiB | 46.00 KiB/s

File Edit View Repository Branch Help

Current repository
peladonerd

Current branch
master

Fetch origin
Last fetched just now

Changes History New

☒ 0 changed files

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor

Select your editor in [Options](#)

Repository menu or **Ctrl** **Shift** **A**

Open in Visual Studio Code

View the files of your repository in Explorer


Repository menu or **Ctrl** **Shift** **F**

Show in Explorer


Open the repository page on GitHub in your browser

Repository menu or **Ctrl** **Shift** **G**

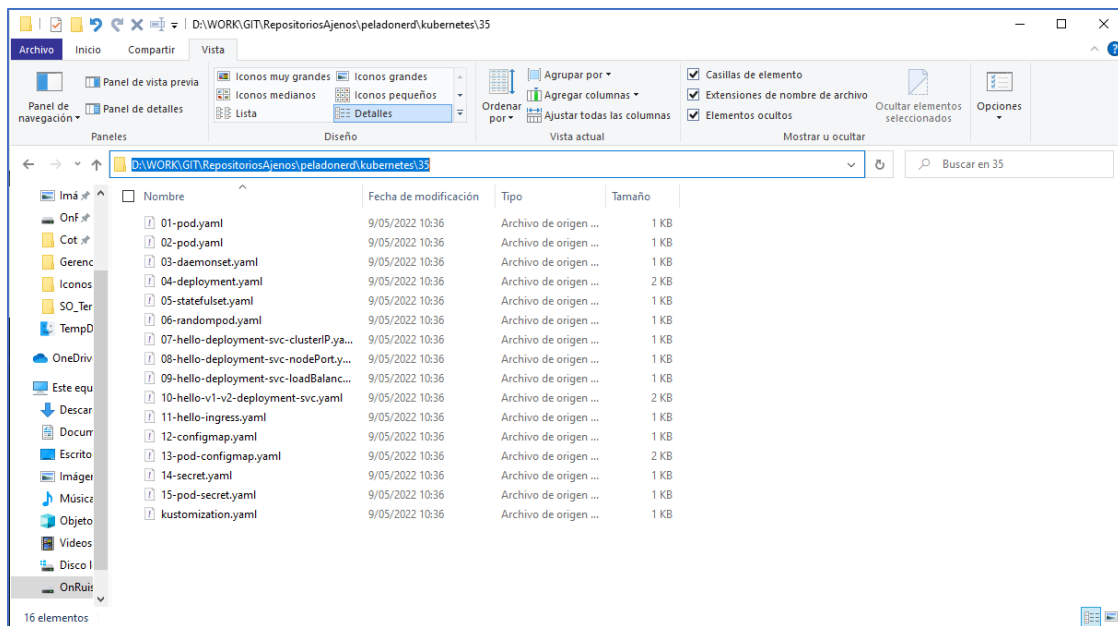
View on GitHub

 Summary (required)

Description



Commit to master



Administrador: Símbolo del sistema

```
C:\Windows\system32>D:
D:\>cd D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```

Listamos los archivos dentro del repositorio y accedemos al 01-pod.yaml, este archivo puede crear manifiestos para el POD el cual consta de varias secciones:

- apiVersion ⇒ Versión del API del recurso de kubernetes.
- Kind ⇒ Tipo de recurso.
- metadata ⇒ Etiquetas o nombres. En este caso se necesita el name que será el nombre del
- pod.
- containers ⇒ Contenedores que correrán dentro del pod, en este caso nginx.

Administrador: Símbolo del sistema

```
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>dir
El volumen de la unidad D es OnRuiso
El número de serie del volumen es: C80C-585E

Directorio de D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35

09/05/2022 10:36 <DIR> .
09/05/2022 10:36 <DIR> ..
09/05/2022 10:36      117 01-pod.yaml
09/05/2022 10:36      743 02-pod.yaml
09/05/2022 10:36    1.016 03-daemonset.yaml
09/05/2022 10:36    1.032 04-deployment.yaml
09/05/2022 10:36    668 05-statefulset.yaml
09/05/2022 10:36    153 06-randompod.yaml
09/05/2022 10:36    500 07-hello-deployment-svc-clusterIP.yaml
09/05/2022 10:36    539 08-hello-deployment-svc-nodePort.yaml
09/05/2022 10:36    522 09-hello-deployment-svc-loadBalancer.yaml
09/05/2022 10:36    1.045 10-hello-v1-v2-deployment-svc.yaml
09/05/2022 10:36    437 11-hello-ingress.yaml
09/05/2022 10:36    422 12-configmap.yaml
09/05/2022 10:36    1.063 13-pod-configmap.yaml
09/05/2022 10:36    361 14-secret.yaml
09/05/2022 10:36    460 15-pod-secret.yaml
09/05/2022 10:36    286 kustomization.yaml
        16 archivos          9.364 bytes
        2 dirs 289.418.227.712 bytes libres

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```



```
Administrador: Símbolo del sistema
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>notepad 01-pod.yaml

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>

01-pod.yaml: Bloc de notas
Archivo Edición Formato Ver Ayuda
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:alpine
```

Como observamos en el bloc de notas, evidenciamos que todos los contenedores pueden correr dentro del POD y estas tienen la misma IP. Tomaremos este archivo para nuestro cluster. Aplicamos el manifiesto de kubernetes y obtendremos la lista de PODS que observamos que tenemos nuestro primer POD de nginx corriendo en nuestro cluster.

```
Administrador: Símbolo del sistema
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl apply -f 01-pod.yaml
pod/nginx created

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     0/1     ContainerCreating   0          23s

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```

Podemos acceder al POD con el uso del comando EXEC y SH, si quisiéramos salir del POD utilizaremos la combinación de teclas [CTRL]+[D].

```
Administrador: Símbolo del sistema - kubectl exec -it nginx -- sh
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl exec -it nginx -- sh
/ #
/ #
/ # ps fax
PID   USER     TIME   COMMAND
  1  root      0:00   nginx: master process nginx -g daemon off;
 33  nginx    0:00   nginx: worker process
 34  nginx    0:00   nginx: worker process
 35  nginx    0:00   nginx: worker process
 36  nginx    0:00   nginx: worker process
 37  nginx    0:00   nginx: worker process
 38  nginx    0:00   nginx: worker process
 39  nginx    0:00   nginx: worker process
 40  nginx    0:00   nginx: worker process
 41  root      0:00   sh
 48  root      0:00   ps fax
/ #
```

En este momento, los PODs no tienen un orden para kubernetes, en el cual se cree un nuevo POD de nginx en caso de que este se borre, esto quiere decir que a diferencia del anterior POD que eliminamos y se regenero al momento, si eliminamos este POD de nginx, no se regenerara.

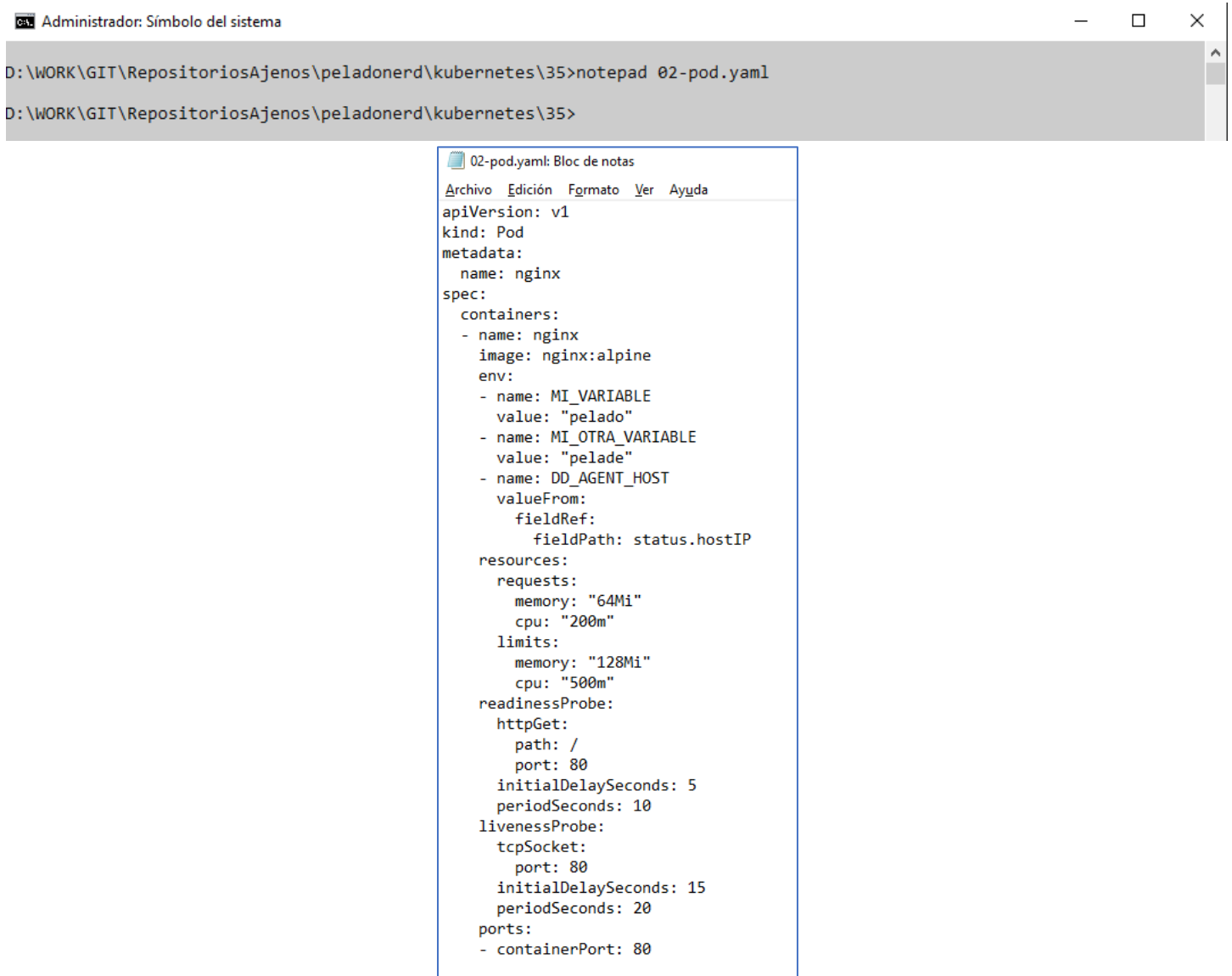
```
Administrador: Símbolo del sistema
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     1/1     Running   0          6m35s

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl delete pod nginx
pod "nginx" deleted

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
No resources found in default namespace.

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```

Ahora veamos nuevamente dentro la carpeta del repositorio, pero ahora abriremos el archivo que se denomina 02-pod.yaml.



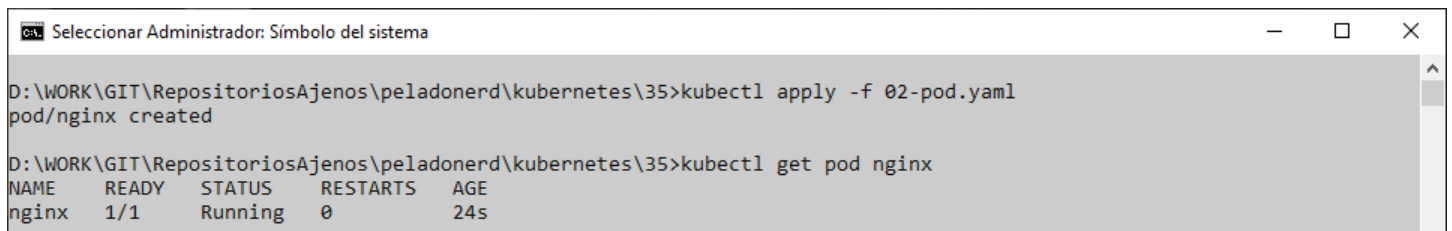
The image shows a Windows command prompt window with the title 'Administrador: Símbolo del sistema'. The command prompt shows the directory 'D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35' and the command 'notepad 02-pod.yaml'. A Notepad window titled '02-pod.yaml: Bloc de notas' is open, displaying the following YAML content:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:alpine
    env:
    - name: MI_VARIABLE
      value: "pelado"
    - name: MI_OTRA_VARIABLE
      value: "pelade"
    - name: DD_AGENT_HOST
      valueFrom:
        fieldRef:
          fieldPath: status.hostIP
  resources:
    requests:
      memory: "64Mi"
      cpu: "200m"
    limits:
      memory: "128Mi"
      cpu: "500m"
  readinessProbe:
    httpGet:
      path: /
      port: 80
    initialDelaySeconds: 5
    periodSeconds: 10
  livenessProbe:
    tcpSocket:
      port: 80
    initialDelaySeconds: 15
    periodSeconds: 20
  ports:
  - containerPort: 80
```

A simple vista podemos observar que este manifiesto tiene muchas mas variables que el anterior. Una de las opciones que se agregaron con respecto al anterior son las variables de entorno, las cuales se componen de un nombre y un valor; kubernetes tiene una característica denominada DOWNWARD API los cuales son valores que se pueden heredar como la dirección IP del host de donde está corriendo el POD.

De esta manera se agrega la sección para limitar los recursos. Existen dos maneras de hacerlo, con e comando REQUEST lo cual da los recursos que garantizan al POD siempre que estén disponibles; y el comando LIMITS el cual establece limites en los recursos para el POD.

Procedemos a aplicar el manifiesto 02-pod.yaml, observando que ahora se tienen mas variables que la anterior vez.



The image shows a Windows command prompt window with the title 'Selección Administrador: Símbolo del sistema'. The command prompt shows the directory 'D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35' and the command 'kubectl apply -f 02-pod.yaml'. The output is 'pod/nginx created'. The next command is 'kubectl get pod nginx', and the output is a table showing the pod's status:

| NAME | READY | STATUS | RESTARTS | AGE |
|-------|-------|---------|----------|-----|
| nginx | 1/1 | Running | 0 | 24s |

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubect1 get pod nginx -o yaml

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubect1.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"nginx","namespace":"default"},"spec":{"con
tainers":[{"env":[{"name":"MI_VARIABLE","value":"pelado"}, {"name":"MI_OTRA_VARIABLE","value":"pelade"}, {"name":"DD_AG
ENT_HOST","valueFrom":{"fieldRef":{"fieldPath":"status.hostIP"}}}], "image":"nginx:alpine", "livenessProbe":{"initialDe
laySeconds":15, "periodSeconds":20, "tcpSocket":{"port":80}}, "name":"nginx", "ports":[{"containerPort":80}], "readinessPr
obe":{"httpGet":{"path":"/", "port":80}, "initialDelaySeconds":5, "periodSeconds":10}, "resources":{"limits":{"cpu":"500m
", "memory":"128Mi"}, "requests":{"cpu":"200m", "memory":"64Mi"}}}]}
  creationTimestamp: "2022-05-10T00:17:35Z"
  name: nginx
  namespace: default
  resourceVersion: "3742"
  uid: 093fdc25-5392-4208-b01e-685e8fabe231
spec:
  containers:
    - env:
      - name: MI_VARIABLE
        value: pelado
      - name: MI_OTRA_VARIABLE
        value: pelade
      - name: DD_AGENT_HOST
        valueFrom:
          fieldRef:
            apiVersion: v1
            fieldPath: status.hostIP
    image: nginx:alpine
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 3
      initialDelaySeconds: 15
      periodSeconds: 20
      successThreshold: 1
      tcpSocket:
        port: 80
      timeoutSeconds: 1
    name: nginx
    ports:
      - containerPort: 80
        protocol: TCP
    readinessProbe:
      failureThreshold: 3
      httpGet:
        path: /
        port: 80
        scheme: HTTP
      initialDelaySeconds: 5
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 1
    resources:
      limits:
        cpu: 500m
        memory: 128Mi
      requests:
        cpu: 200m
        memory: 64Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
```

```
volumeMounts:
- mountPath: /var/run/secrets/kubernetes.io/serviceaccount
  name: kube-api-access-82g77
  readOnly: true
dnsPolicy: ClusterFirst
enableServiceLinks: true
nodeName: minikube
preemptionPolicy: PreemptLowerPriority
priority: 0
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
serviceAccount: default
serviceAccountName: default
terminationGracePeriodSeconds: 30
tolerations:
- effect: NoExecute
  key: node.kubernetes.io/not-ready
  operator: Exists
  tolerationSeconds: 300
- effect: NoExecute
  key: node.kubernetes.io/unreachable
  operator: Exists
  tolerationSeconds: 300
volumes:
- name: kube-api-access-82g77
  projected:
    defaultMode: 420
    sources:
    - serviceAccountToken:
        expirationSeconds: 3607
        path: token
    - configMap:
        items:
        - key: ca.crt
          path: ca.crt
        name: kube-root-ca.crt
    - downwardAPI:
        items:
        - fieldRef:
            apiVersion: v1
            fieldPath: metadata.namespace
          path: namespace
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2022-05-10T00:17:35Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2022-05-10T00:17:45Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2022-05-10T00:17:45Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2022-05-10T00:17:35Z"
    status: "True"
```

```

  type: PodScheduled
containerStatuses:
- containerID: docker://eb30848d17bda79d8c6edd5881232c552a5350a2b856bc7b363f1b7274bbfe06
  image: nginx:alpine
  imageID: docker-pullable://nginx@sha256:5a0df7fb7c8c03e4158ae9974bfb6a15da2bdfdeded4fb694367ec812325d31
  lastState: {}
  name: nginx
  ready: true
  restartCount: 0
  started: true
  state:
    running:
      startedAt: "2022-05-10T00:17:36Z"
hostIP: 192.168.49.2
phase: Running
podIP: 172.17.0.3
podIPs:
- ip: 172.17.0.3
qosClass: Burstable
startTime: "2022-05-10T00:17:35Z"

```

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>

Ahora probemos eliminarlo.

CA Administrador: Símbolo del sistema

```

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     1/1     Running   0           13m

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl delete pod nginx
pod "nginx" deleted

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
No resources found in default namespace.

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>_

```

Ahora creemos un pod con el manifiesto 04-deployment.yml

CA Administrador: Símbolo del sistema

```

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>dir
El volumen de la unidad D es OnRuiso
El número de serie del volumen es: C80C-585E

Directorio de D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35

09/05/2022  10:36    <DIR>          .
09/05/2022  10:36    <DIR>          ..
09/05/2022  10:36                117 01-pod.yaml
09/05/2022  10:36                743 02-pod.yaml
09/05/2022  10:36            1.016 03-daemonset.yaml
09/05/2022  10:36            1.032 04-deployment.yaml
09/05/2022  10:36            668 05-statefulset.yaml
09/05/2022  10:36            153 06-randompod.yaml
09/05/2022  10:36            500 07-hello-deployment-svc-clusterIP.yaml
09/05/2022  10:36            539 08-hello-deployment-svc-nodePort.yaml
09/05/2022  10:36            522 09-hello-deployment-svc-loadBalancer.yaml
09/05/2022  10:36            1.045 10-hello-v1-v2-deployment-svc.yaml
09/05/2022  10:36            437 11-hello-ingress.yaml
09/05/2022  10:36            422 12-configmap.yaml
09/05/2022  10:36            1.063 13-pod-configmap.yaml
09/05/2022  10:36            361 14-secret.yaml
09/05/2022  10:36            460 15-pod-secret.yaml
09/05/2022  10:36            286 kustomization.yaml
               16 archivos          9.364 bytes
               2 dirs 285.981.081.600 bytes libres

```

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>

CA Administrador: Símbolo del sistema

```

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>notepad 04-deployment.yml

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>

```

```
04-deployment.yaml: Bloc de notas
Archivo Edición Formato Ver Ayuda
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        env:
        - name: MI_VARIABLE
          value: "pelado"
        - name: MI_OTRA_VARIABLE
          value: "pelade"
        - name: DD_AGENT_HOST
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
      resources:
        requests:
          memory: "64Mi"
          cpu: "200m"
        limits:
          memory: "128Mi"
          cpu: "500m"
      readinessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 10
      livenessProbe:
        tcpSocket:
          port: 80
        initialDelaySeconds: 15
        periodSeconds: 20
      ports:
      - containerPort: 80
```

```
Administrador: Símbolo del sistema
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl apply -f 04-deployment.yaml
deployment.apps/nginx-deployment created

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66c9c7669-9tcr2    1/1     Running   0           22s
nginx-deployment-66c9c7669-rc4x7    1/1     Running   0           22s

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl delete pod nginx-deployment-66c9c7669-9tcr2
pod "nginx-deployment-66c9c7669-9tcr2" deleted

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66c9c7669-5k7gp     0/1     Running   0           5s
nginx-deployment-66c9c7669-rc4x7    1/1     Running   0          4m49s

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```

El pod creado por DEPLOYMENT tiene la configuración correcta para generar dos tipos de POD, en caso de que se elimine uno, se regenerará al instante para mantener este número. Ahora intentemos con un medio de tipo DAEMONSET.

```
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>dir
El volumen de la unidad D es OnRuiso
El número de serie del volumen es: C80C-585E

Directorio de D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35

09/05/2022  10:36    <DIR>          .
09/05/2022  10:36    <DIR>          ..
09/05/2022  10:36             117 01-pod.yaml
09/05/2022  10:36             743 02-pod.yaml
09/05/2022  10:36          1.016 03-daemonset.yaml
09/05/2022  10:36          1.032 04-deployment.yaml
09/05/2022  10:36             668 05-statefulset.yaml
09/05/2022  10:36             153 06-randompod.yaml
09/05/2022  10:36           500 07-hello-deployment-svc-clusterIP.yaml
09/05/2022  10:36           539 08-hello-deployment-svc-nodePort.yaml
09/05/2022  10:36           522 09-hello-deployment-svc-loadBalancer.yaml
09/05/2022  10:36          1.045 10-hello-v1-v2-deployment-svc.yaml
09/05/2022  10:36           437 11-hello-ingress.yaml
09/05/2022  10:36           422 12-configmap.yaml
09/05/2022  10:36          1.063 13-pod-configmap.yaml
09/05/2022  10:36           361 14-secret.yaml
09/05/2022  10:36           460 15-pod-secret.yaml
09/05/2022  10:36           286 kustomization.yaml
                16 archivos          9.364 bytes
                2 dirs 285.981.081.600 bytes libres

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```

```
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>notepad 03-daemonset.yaml
D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>
```

```
03-daemonset.yaml: Bloc de notas
Archivo Edición Formato Ver Ayuda
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:alpine
        env:
        - name: MI_VARIABLE
          value: "pelado"
        - name: MI_OTRA_VARIABLE
          value: "pelade"
        - name: DD_AGENT_HOST
          valueFrom:
            fieldRef:
              fieldPath: status.hostIP
      resources:
        requests:
          memory: "64Mi"
          cpu: "200m"
        limits:
          memory: "128Mi"
          cpu: "500m"
      readinessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 10
      livenessProbe:
        tcpSocket:
          port: 80
        initialDelaySeconds: 15
        periodSeconds: 20
      ports:
      - containerPort: 80
```


D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>notepad 03-daemonset.yaml

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl apply -f 03-daemonset.yaml
daemonset.apps/nginx-deployment created

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods

| NAME | READY | STATUS | RESTARTS | AGE |
|----------------------------------|-------|---------|----------|-------|
| nginx-deployment-66c9c7669-5k7gp | 1/1 | Running | 0 | 7m52s |
| nginx-deployment-66c9c7669-rc4x7 | 1/1 | Running | 0 | 12m |
| nginx-deployment-hj65p | 1/1 | Running | 0 | 13s |

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>kubectl get pods -o wide

| NAME | READY | STATUS | RESTARTS | AGE | IP | NODE | NOMINATED NODE | READINESS GATES |
|----------------------------------|-------|---------|----------|-------|------------|----------|----------------|-----------------|
| nginx-deployment-66c9c7669-5k7gp | 1/1 | Running | 0 | 8m11s | 172.17.0.5 | minikube | <none> | <none> |
| nginx-deployment-66c9c7669-rc4x7 | 1/1 | Running | 0 | 12m | 172.17.0.4 | minikube | <none> | <none> |
| nginx-deployment-hj65p | 1/1 | Running | 0 | 32s | 172.17.0.3 | minikube | <none> | <none> |

D:\WORK\GIT\RepositoriosAjenos\peladonerd\kubernetes\35>