

### Guía Número 3 de DevOps – Kubernetes

- Prerrequisito: Tener instalado Docker, sino es así acceder a este enlace para descargar el instalador y posteriormente realizar la instalación.

[https://drive.google.com/file/d/1mbykZVf\\_4ZGKwwVAoBq4ugzmJUEy0WdS/view?usp=sharing](https://drive.google.com/file/d/1mbykZVf_4ZGKwwVAoBq4ugzmJUEy0WdS/view?usp=sharing)

- Prerrequisito: Tener instalado Minikube, sino es así acceder a este enlace para descargar el instalador y posteriormente realizar la instalación.

<https://minikube.sigs.k8s.io/docs/start/>

- Prerrequisito: Tener instalado Kubectl, sino es así acceder a este enlace para descargar el instalador y posteriormente realizar la instalación.

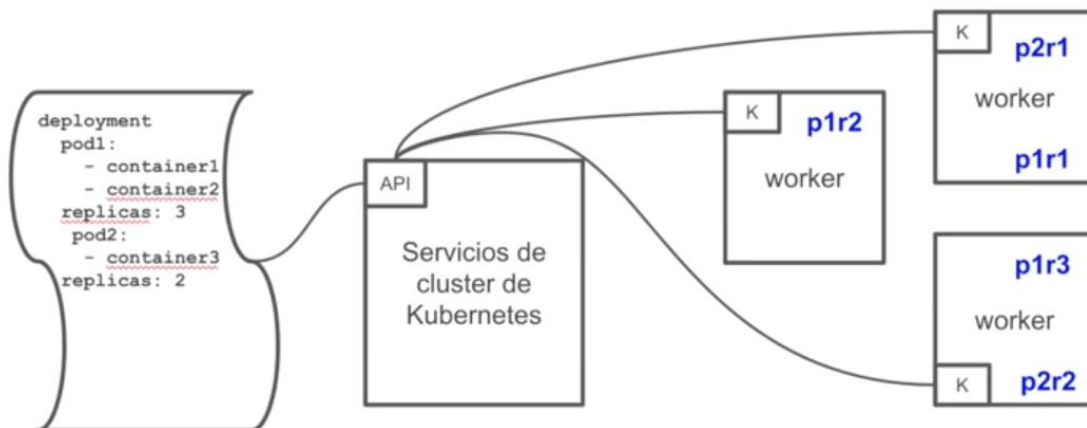
<https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>

Kubernetes como tecnología que permite orquestar los contenedores.

Kubernetes es declarativo ya que permite crear manifiestos (tipo de deployment). Estos manifiestos facilitan que kubernetes pueda crear pods con una cantidad específica de contenedores y réplicas, para que así los servicios de los cluster de kubernetes (API) puedan cumplir con las órdenes dadas. Los workers serán los espacios donde correrán nuestras aplicaciones o contenedores.

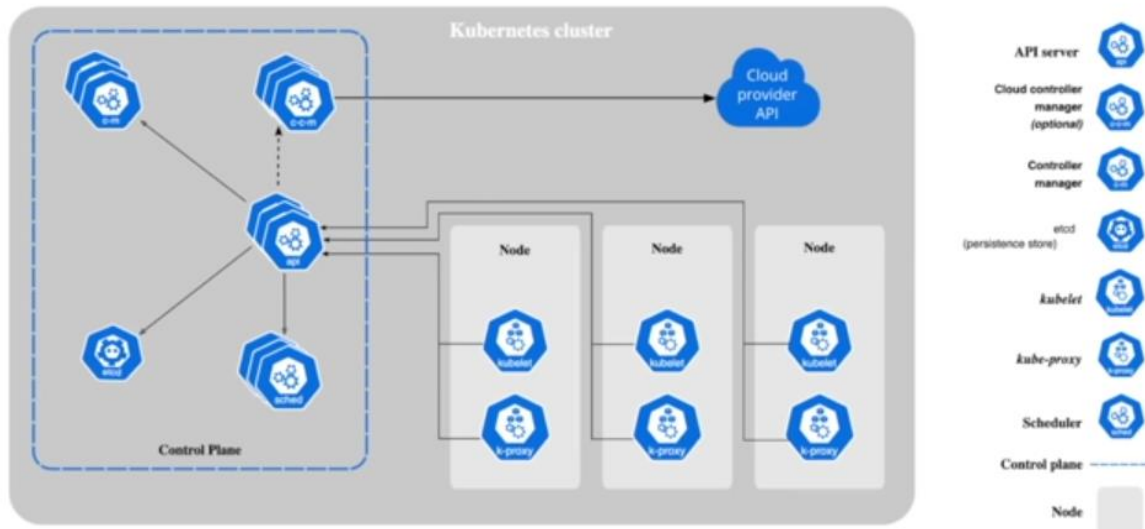
Los contenedores son manejados por Scheduler (Servicios de cluster de Kubernetes) que se encargan de moverlos de un lugar a otro y se conectan a la API de kubernetes con los workers a través de un agente que corre en cada uno de los workers que se llama Kubelet.

Kubelet es un servicio de kubernetes que permite conectar todos los workers y servicios de kubernetes entre sí, por lo cual si uno de esos worker, instancias o incluso alguno de esos contenedores se cae, kubernetes se encarga de reemplazarlo automáticamente .



Componentes:

- Control plane ⇒ Servidores de Kubernetes
- Nodos ⇒ Cada uno correrá kubelet (agente de Kubernetes) y el servicio de kube-proxy (recibe el tráfico y lo manda a los pods que requieran ese tráfico)
- Scheduler
- Cloud Controller Manager ⇒ Se conecta a la API de tu proveedor de cloud
- etcd ⇒ Base de datos que te permite guardar el estado de tu cluster de Kubernetes



1. Ejecutamos este comando para conocer la versión de kubectl que se tiene instalado, recordando que esta herramienta nos permite interactuar con el cluster.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Nicolas>kubectl version --client=true
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.5", GitCommit:"5c99e2ac2ff9a3c549d9ca665e7bc05a3e18f07e", GitTreeState:"clean", BuildDate:"2021-12-16T08:38:33Z", GoVersion:"go1.16.12", Compiler:"gc", Platform:"windows/amd64"}

C:\Users\Nicolas>

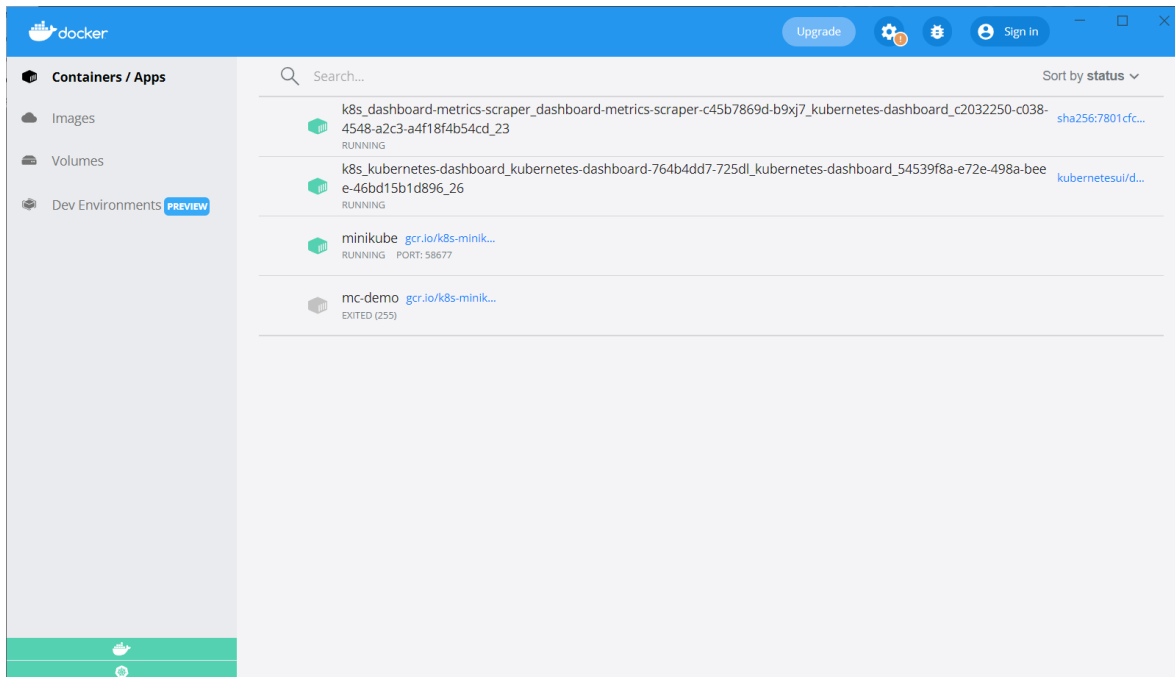
```

## 2. Iniciamos minikube.

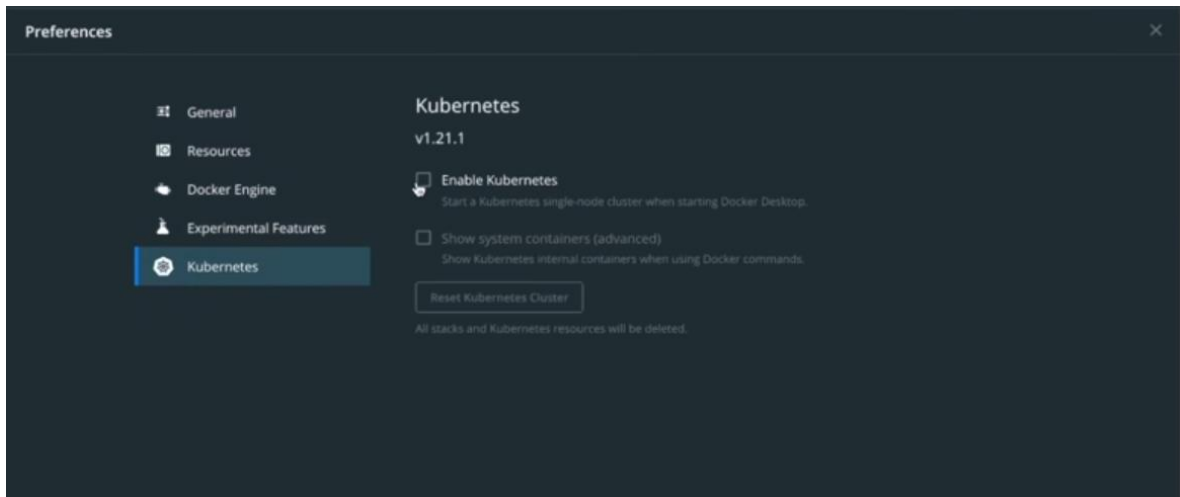
```
C:\Users\Nicolas>minikube start
* minikube v1.25.2 en Microsoft Windows 10 Home Single Language 10.0.19044 Build 19044
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Restarting existing docker container for "minikube" ...
* Preparando Kubernetes v1.23.3 en Docker 20.10.12...
  - kubelet.housekeeping-interval=5m
* 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:\Users\Nicolas>
```

## 3. Validamos que minikube este corriendo con Docker.



## 4. Verificamos que kubernetes está habilitado con docker.



5. Con este comando podremos observar los nodos de nuestro cluster de kubernetes.

```
C:\Users\Nicolas>kubectl get nodes
NAME        STATUS    ROLES                  AGE    VERSION
minikube    Ready     control-plane,master   7d4h   v1.23.3
```

6. Kubectl es nuestro cliente de kubernetes que va a permitir conectarnos a nuestros cluster y hay varios comando que podemos utilizar, el más común es el get, que sirve para obtener recursos, otros son: edit (editar un recurso), delete (eliminar recurso), apply (aplicar algún manifiesto a tu cluster de kubernetes), exec (permite ejecutar un comando dentro de un contenedor), logs, cp (copiar archivos desde nuestra máquina al contenedor) y cordon - uncordon - drain (utiles para manejar nodos).

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Nicolas>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

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 or zsh)

Other Commands:
  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).
```

7. Con este comando se obtienen los contextos que se tienen en el archivo de configuración.

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Nicolas>kubectl config get-contexts
CURRENT  NAME          CLUSTER          AUTHINFO          NAMESPACE
*        docker-desktop  docker-desktop  docker-desktop    default
mc-demo  mc-demo       mc-demo         mc-demo          default
minikube minikube      minikube        minikube          default
```

## 8. Recursos de kubernetes:

- a) Namespace ⇒ Es una división lógica de tu cluster de kubernetes, en otras palabras, permite separar tu carga/tráfico en tu cluster de kubernates. Con este comando permite obtener los namespace que tenemos.

```
C:\Users\Nicolas>kubectl get ns
NAME          STATUS  AGE
default       Active  7d4h
kube-node-lease  Active  7d4h
kube-public    Active  7d4h
kube-system    Active  7d4h
```

- b) Pod ⇒ Es un set de 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.

Con este comando podemos ver los pods que tenemos dentro del namespace de kube-system

```
C:\Users\Nicolas>kubectl -n kube-system get pods
NAME                                READY  STATUS   RESTARTS   AGE
coredns-64897985d-wlp5w            1/1    Running  1 (9m45s ago)  7d4h
etcd-minikube                      1/1    Running  1 (9m45s ago)  7d4h
kube-apiserver-minikube            1/1    Running  1 (9m45s ago)  7d4h
kube-controller-manager-minikube   1/1    Running  1 (9m45s ago)  7d4h
kube-proxy-bdr5l                   1/1    Running  1 (9m45s ago)  7d4h
kube-scheduler-minikube            1/1    Running  1 (9m45s ago)  7d4h
storage-provisioner                 1/1    Running  4 (9m8s ago)   7d4h
```

Con este comando podemos ver información adicional de cada uno de los pods listados anteriormente.

```
C:\Users\Nicolas>kubectl -n kube-system get pods -o wide
NAME                                READY  STATUS   RESTARTS   AGE  IP            NODE      NOMINATED NODE  READINESS GATES
coredns-64897985d-wlp5w            1/1    Running  1 (11m ago)  7d4h  172.17.0.2    minikube  <none>          <none>
etcd-minikube                      1/1    Running  1 (11m ago)  7d4h  192.168.49.2  minikube  <none>          <none>
kube-apiserver-minikube            1/1    Running  1 (11m ago)  7d4h  192.168.49.2  minikube  <none>          <none>
kube-controller-manager-minikube   1/1    Running  1 (11m ago)  7d4h  192.168.49.2  minikube  <none>          <none>
kube-proxy-bdr5l                   1/1    Running  1 (11m ago)  7d4h  192.168.49.2  minikube  <none>          <none>
kube-scheduler-minikube            1/1    Running  1 (11m ago)  7d4h  192.168.49.2  minikube  <none>          <none>
storage-provisioner                 1/1    Running  4 (10m ago)  7d4h  192.168.49.2  minikube  <none>          <none>
```

En este caso vamos a verificar si al eliminar un pod se crea uno automáticamente para reemplazarlo, para ello, listamos los pods que tenemos para luego eliminar alguno con el comando delete y luego volvemos a listarlos, observando que es cierto que si por algún motivo un pod falla kubernetes se encarga crear uno nuevo casi al instante de manera transparente al usuario.

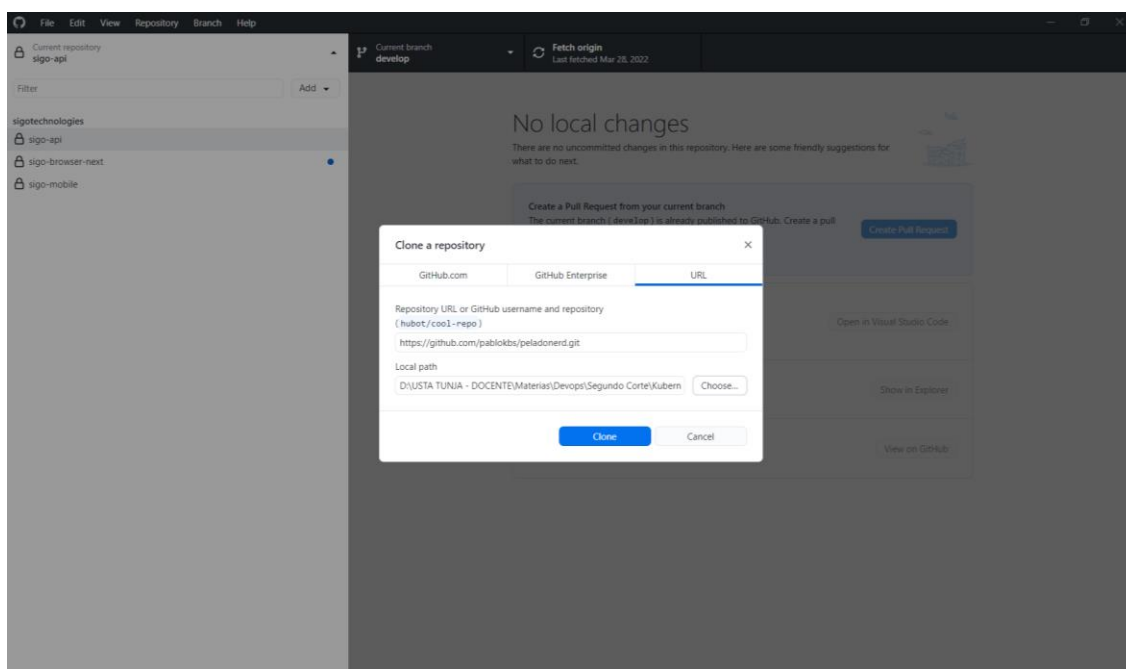
```
C:\WINDOWS\system32\cmd.exe

C:\Users\Nicolas>kubectl -n kube-system get pods
NAME                                READY   STATUS    RESTARTS   AGE
coredns-64897985d-wlp5w            1/1     Running   1 (15m ago) 7d4h
etcd-minikube                      1/1     Running   1 (15m ago) 7d4h
kube-apiserver-minikube            1/1     Running   1 (15m ago) 7d4h
kube-controller-manager-minikube    1/1     Running   1 (15m ago) 7d4h
kube-proxy-bdr5l                   1/1     Running   1 (15m ago) 7d4h
kube-scheduler-minikube            1/1     Running   1 (15m ago) 7d4h
storage-provisioner                 1/1     Running   4 (14m ago) 7d4h

C:\Users\Nicolas>kubectl -n kube-system delete pod kube-proxy-bdr5l
pod "kube-proxy-bdr5l" deleted

C:\Users\Nicolas>kubectl -n kube-system get pods
NAME                                READY   STATUS    RESTARTS   AGE
coredns-64897985d-wlp5w            1/1     Running   1 (15m ago) 7d4h
etcd-minikube                      1/1     Running   1 (15m ago) 7d4h
kube-apiserver-minikube            1/1     Running   1 (15m ago) 7d4h
kube-controller-manager-minikube    1/1     Running   1 (15m ago) 7d4h
kube-proxy-jtpw6                   1/1     Running   0          12s
kube-scheduler-minikube            1/1     Running   1 (15m ago) 7d4h
storage-provisioner                 1/1     Running   4 (15m ago) 7d4h
```

9. Para hacer las siguientes pruebas clonamos el repositorio <https://github.com/pablokbs/peladonerd>



Por terminal nos posicionamos en la siguiente ruta:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>
```

10. Con el comando `ls` listamos los archivos que hay dentro de la carpeta y accedemos mediante el comando `vim` al que se llama `01-pod.yaml`.

Los recursos de kubernetes se pueden crear basados en manifiestos que son archivos `.yaml`

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>ls
01-pod.yaml          07-hello-deployment-svc-clusterIP.yaml  13-pod-configmap.yaml
02-pod.yaml          08-hello-deployment-svc-nodePort.yaml   14-secret.yaml
03-daemonset.yaml    09-hello-deployment-svc-loadBalancer.yaml 15-pod-secret.yaml
04-deployment.yaml   10-hello-v1-v2-deployment-svc.yaml       kustomization.yaml
05-statefulset.yaml  11-hello-ingress.yaml
06-randompod.yaml    12-configmap.yaml

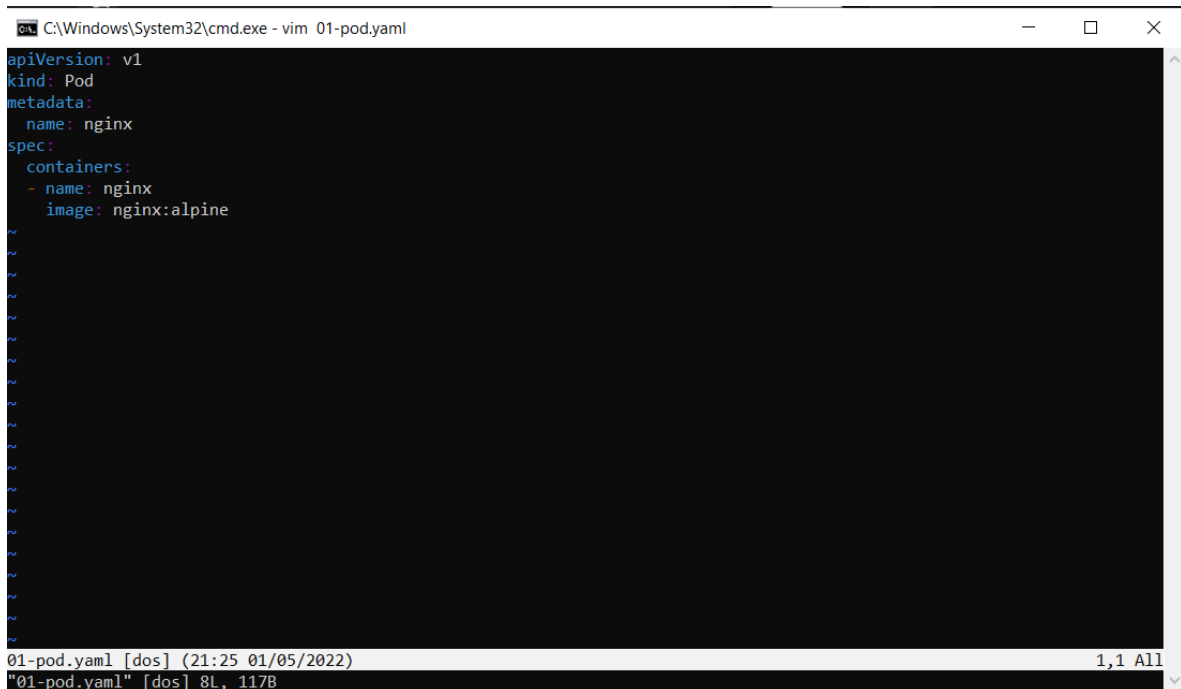
D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>vim 01-pod.yaml
```

Este archivo es un manifiesto para crear un pod que consta de varias secciones:

- a) `apiVersion` ⇒ Versión de la api de el recurso de kubernetes.
- b) `Kind` ⇒ Tipo de recurso.
- c) `metadata` ⇒ Etiquetas o nombres. En este caso se necesita el `name` que será el nombre del pod.
- d) `containers` ⇒ Contenedores que correrán dentro del pod, en este caso `nginx`.



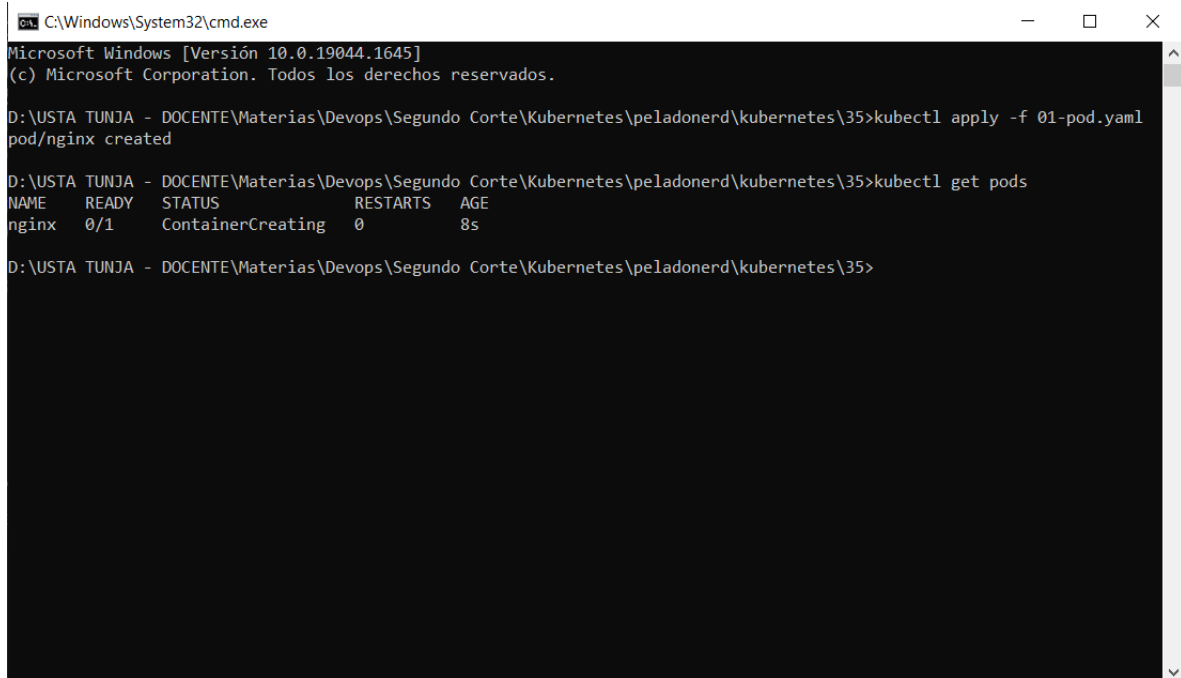
Todos los contenedores que corren dentro del pod tienen la misma ip.



```
C:\Windows\System32\cmd.exe - vim 01-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:alpine
```

01-pod.yaml [dos] (21:25 01/05/2022) 1,1 All  
"01-pod.yaml" [dos] 8L, 117B

11. Aplicamos el manifiesto de kubernetes y al obtener la lista de pods observamos que tenemos nuestro primer pod de nginx corriendo en nuestro cluster.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl apply -f 01-pod.yaml
pod/nginx created

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginx     0/1     ContainerCreating   0          8s

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>
```

12. Accedemos al pod por medio de la terminal de manera interactiva pasando el comando que queremos correr dentro de ese pod, en este caso sh. Control + d para salir.

```

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\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;
32 nginx 0:00 nginx: worker process
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 root 0:00 sh
47 root 0:00 ps fax
/#

```

13. Obtenemos los pods y eliminamos el de nginx, por último volvemos a listarlos observando que no se creo uno de respaldo automáticamente, esto sucede ya que no se estableció una orden para kubernetes en la cual se cree un nuevo pod de nginx en caso de que se borre.

```

C:\Windows\System32\cmd.exe
D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME READY STATUS RESTARTS AGE
nginx 1/1 Running 0 2m35s

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl delete pod nginx
pod "nginx" deleted

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
No resources found in default namespace.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>

```

14. Con el comando ls listamos los archivos que hay dentro de la carpeta y accedemos mediante el comando vim al que se llama 02-pod.yaml.

```

C:\Windows\System32\cmd.exe
D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>ls
01-pod.yaml      07-hello-deployment-svc-clusterIP.yaml  13-pod-configmap.yaml
02-pod.yaml      08-hello-deployment-svc-nodePort.yaml   14-secret.yaml
03-daemonset.yaml 09-hello-deployment-svc-loadBalancer.yaml 15-pod-secret.yaml
04-deployment.yaml 10-hello-v1-v2-deployment-svc.yaml      kustomization.yaml
05-statefulset.yaml 11-hello-ingress.yaml
06-randompod.yaml 12-configmap.yaml

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>vim 02-pod.yaml

```

Una de las opciones que se han agregado son las variables de entorno, que se componen de un nombre y un valor. Kubernetes tiene algo llamado downward api que son valores que se pueden heredar como la dirección ip del host de donde esta corriendo el pod.

Así mismo se agrega la sección para limitar los recursos. Existen dos formas: request (son los recursos que le vamos a garantizar a este pod que siempre va a tener disponibles) y limits (Es el límite de recursos que el pod puede utilizar).

```
C:\Windows\System32\cmd.exe - vim 02-pod.yaml
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
~
~
02-pod.yaml [dos] (21:25 01/05/2022) 1,1 All
```

15. Aplicamos el manifiesto 02-pod.yaml, obtenemos los pods y luego visualizamos el manifiesto más las variables.

- Aclaración, en este punto el comando es: `kubectl get pod nginx`

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl apply -f 02-pod.yaml
pod/nginx created

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0           10s

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods nginx
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0           29s
```

- Aclaración, en este punto el comando es: `kubectl get pod nginx -o yaml`

```

C:\Windows\System32\cmd.exe
D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods nginx -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"nginx","namespace":"default"},"spec":{"containers":[{"name":"ML_VARIABLE","value":"pelado"},"(name":"ML_OTRA_VARIABLE",
"value":"pelado"},"(name":"DD_AGENT_HOST","valueFrom":{"fieldRef":{"fieldPath":"status.hostIP}}}],"image":"nginx:alpine","livenessProbe":{"initialDelaySeconds":15,"periodSeconds":20,"tcpSocket":{"port":
80},"name":"nginx","ports":[{"containerPort":80}],"readinessProbe":{"httpGet":{"path":"/","port":80},"initialDelaySeconds":5,"periodSeconds":10},"resources":{"limits":{"cpu":"500m","memory":"128Mi"},"
requests":{"cpu":"200m","memory":"64Mi"}}}}}}
creationTimestamp: "2022-05-02T02:39:12Z"
name: nginx
namespace: default
resourceVersion: "5235"
uid: ab75d355-6e60-4fa5-abbf-d7c9954d2953
spec:
  containers:
    - env:
      - name: ML_VARIABLE
        value: pelado
      - name: ML_OTRA_VARIABLE
        value: pelado
      - 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

```

16. Eliminamos el pod de nginx.

```

C:\Windows\System32\cmd.exe
qosClass: Burstable
startTime: "2022-05-02T02:39:12Z"

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl delete pod nginx
pod "nginx" deleted

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>

```

17. Listamos nuestros archivos y abrimos el archivo llamado 04-deployment.yml que es el manifiesto de los deployment .

```

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>ls
01-pod.yaml          07-hello-deployment-svc-clusterIP.yaml  13-pod-configmap.yaml
02-pod.yaml          08-hello-deployment-svc-nodePort.yaml   14-secret.yaml
03-daemonset.yaml    09-hello-deployment-svc-loadBalancer.yaml 15-pod-secret.yaml
04-deployment.yaml   10-hello-v1-v2-deployment-svc.yaml      kustomization.yaml
05-statefulset.yaml  11-hello-ingress.yaml
06-randompod.yaml    12-configmap.yaml

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>vim 04-deployment.yaml

```

Es muy parecido al manifiesto de un pod, en este caso las réplicas son la cantidad de pods que queremos. Un deployment es un template para crear pods.

```
C:\Windows\System32\cmd.exe - vim 04-deployment.yaml
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: "pelado"
            - 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
```

18. Aplicamos el deployment y obtenemos los pods observando que ha creado 2. Al crear un deployment y no un pod observamos que se especificó que deben mantenerse 2 réplicas, por lo cual aunque borremos un pod, se creará uno nuevo para mantener este número.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl apply -f 04-deployment.yaml
deployment.apps/nginx-deployment created

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66c9c7669-qb6h5    1/1     Running   0           10s
nginx-deployment-66c9c7669-qmp5q    1/1     Running   0           10s

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66c9c7669-qb6h5    1/1     Running   0           40s
nginx-deployment-66c9c7669-qmp5q    1/1     Running   0           40s

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl delete pod nginx-deployment-66c9c7669-qb6h5
pod "nginx-deployment-66c9c7669-qb6h5" deleted

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66c9c7669-qmp5q    1/1     Running   0          106s
nginx-deployment-66c9c7669-qmshh    1/1     Running   0           13s
```

19. Ahora crearemos un pod por medio del daemonset, que al hacerle deploy estará en todos los nodos, un solo pod en un cada nodo. Para ello listamos los archivos y abrimos el que se llama 03-daemon.yaml

```
C:\Windows\System32\cmd.exe

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>ls
01-pod.yaml          07-hello-deployment-svc-clusterIP.yaml  13-pod-configmap.yaml
02-pod.yaml          08-hello-deployment-svc-nodePort.yaml    14-secret.yaml
03-daemonset.yaml    09-hello-deployment-svc-loadBalancer.yaml 15-pod-secret.yaml
04-deployment.yaml   10-hello-v1-v2-deployment-svc.yaml       kustomization.yaml
05-statefulset.yaml  11-hello-ingress.yaml
06-randompod.yaml    12-configmap.yaml

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>vim 03-daemonset.yaml
```

En este caso no tiene réplicas ya que dependerá de la cantidad de nodos que se tengan, por ejemplo, si se tienen tres nodos existirán tres pods.

```
C:\Windows\System32\cmd.exe - vim 03-daemonset.yaml

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

03-daemonset.yaml [dos] (21:25 01/05/2022) 1,1 Top
```

20. Aplicamos el manifiesto daemonset, obtenemos los pods y los listamos con la información más detallada.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1645]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl apply -f 03-daemonset.
yaml
daemonset.apps/nginx-deployment created

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66c9c7669-qmp5q    1/1     Running   0           5m41s
nginx-deployment-66c9c7669-qmshh    1/1     Running   0           4m8s
nginx-deployment-l59bq              1/1     Running   0           19s

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP             NODE       NOMINATED NODE   READINESS GATES
nginx-deployment-66c9c7669-qmp5q    1/1     Running   0           5m52s  172.17.0.3     minikube   <none>            <none>
nginx-deployment-66c9c7669-qmshh    1/1     Running   0           4m19s  172.17.0.5     minikube   <none>            <none>
nginx-deployment-l59bq              1/1     Running   0           30s    172.17.0.4     minikube   <none>            <none>
```

21. Comprobamos que aunque se elimine un pod se crea uno automáticamente ya que se tiene 3 nodos.

```
D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl delete pod nginx-deplo
yment-l59bq
pod "nginx-deployment-l59bq" deleted

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP             NODE       NOMINATED NODE   READINESS GATES
nginx-deployment-66c9c7669-qmp5q    1/1     Running   0           7m16s  172.17.0.3     minikube   <none>            <none>
nginx-deployment-66c9c7669-qmshh    1/1     Running   0           5m43s  172.17.0.5     minikube   <none>            <none>
nginx-deployment-cqndr              0/1     Running   0           5s     172.17.0.4     minikube   <none>            <none>

D:\USTA TUNJA - DOCENTE\Materias\Devops\Segundo Corte\Kubernetes\peladonerd\kubernetes\35>
```