



**UNIVERSIDAD POLITÉCNICA DE VALENCIA**

**MÁSTER UNIVERSITARIO EN COMPUTACIÓN EN LA NUBE Y ALTAS  
PRESTACIONES**

**ASIGNATURA:  
CLOUD COMPUTING**

**PROYECTO FINAL  
BUILDING A CUSTOM KUBERNETES SCHEDULER IN PYTHON**

**CATEDRÁTICO:  
JOSÉ MANUEL BERNABEU AUBÁN**

**INTEGRANTES:  
STEFANIE CIBRIAN  
ANA PAMELA RUIZ  
WAYNER MOYA RAMIREZ**

**VALENCIA, ESPAÑA**

**DICIEMBRE, 2025**

## Environment Setup

### Create and Inspect Cluster

```
pame@PameLaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/lab-kubernetes-scheduler$ make create-cluster
kind create cluster --config kind-multi-node.yml --name sched-lab
Creating cluster "sched-lab" ...
 ✓ Ensuring node image (kindest/node:v1.34.0)
 ✓ Preparing nodes
 ✓ Writing configuration
 ✓ Starting control-plane
 ✓ Installing CNI
 ✓ Installing StorageClass
 ✓ Joining worker nodes
Set kubectl context to "kind-sched-lab"
You can now use your cluster with:

kubectl cluster-info --context kind-sched-lab

Thanks for using kind!
```

```
stefanie.cibrian@wifialu62-211 ~ % kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:65394
CoreDNS is running at https://127.0.0.1:65394/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

```
stefanie.cibrian@wifialu62-211 ~ % kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
desktop-control-plane	Ready	control-plane	5m23s	v1.31.1
desktop-worker	Ready	<none>	5m10s	v1.31.1
desktop-worker2	Ready	<none>	5m10s	v1.31.1

```
pame@PameLaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing$ kubectl -n kube-system get pods
```

NAME	READY	STATUS	RESTARTS	AGE
coredns-66bc5c9577-wl2qv	1/1	Running	0	3m57s
coredns-66bc5c9577-wsp7v	1/1	Running	0	3m57s
etcd-kind-control-plane	1/1	Running	0	4m3s
kindnet-2tsd8	1/1	Running	0	3m55s
kindnet-5hr6b	1/1	Running	0	3m55s
kindnet-r2zbr	1/1	Running	0	3m57s
kube-apiserver-kind-control-plane	1/1	Running	0	4m3s
kube-controller-manager-kind-control-plane	1/1	Running	0	4m3s
kube-proxy-lc82t	1/1	Running	0	3m55s
kube-proxy-lvd2r	1/1	Running	0	3m57s
kube-proxy-z2l9b	1/1	Running	0	3m55s
kube-scheduler-kind-control-plane	1/1	Running	0	4m3s

### Observe the Default Scheduler

#### 1. Identify the running scheduler:

```
stefanie.cibrian@wifialu61-187 py-scheduler-repo.o % kubectl -n kube-system get pods -l component=kube-scheduler
```

NAME	READY	STATUS	RESTARTS	AGE
kube-scheduler-desktop-control-plane	1/1	Running	34	13d

#### 2. Schedule a simple pod:

```
stefanie.cibrian@wifialu62-211 ~ % kubectl run test --image=nginx --restart=Never
pod/test created
```

```
stefanie.cibrian@wifialu61-187 py-scheduler-repo.o % kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS	GATES
test-pod	1/1	Running	0	3d	10.244.2.2	desktop-worker2	<none>		<none>	

### Checkpoint 1:

#### Describe the path:

**kubectl run → Pod created → Scheduler assigns Node → kubelet starts Pod.**

Tenemos nuestro archivo de pod creado y lo aplicamos con kubectl apply. Esto genera un nuevo pod que queda en estado pendiente hasta que el scheduler lo

analiza y le asigna un nodo de acuerdo a los taints. Una vez asignado, kubelet crea el pod en el nodo correspondiente.

## Implement the Polling Scheduler

```
pame@PameLaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/lab-kubernetes-scheduler$ make deploy-poll
kubectl apply -f Polling-Scheduler/rbac-deploy.yaml
serviceaccount/my-scheduler unchanged
clusterrolebinding.rbac.authorization.k8s.io/my-scheduler-binding unchanged
deployment.apps/my-scheduler configured
```

```
wayner@wifialu63-167 py-scheduler-repo.o % kubectl -n kube-system get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
coredns       2/2     2            2           16m
my-scheduler  1/1     1            1           3m53s
wayner@wifialu63-167 py-scheduler-repo.o %
```

```
wayner@wifialu63-167 py-scheduler-repo.o % kubectl apply -f test-pod.yaml
pod/test-pod created
wayner@wifialu63-167 py-scheduler-repo.o % kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE           NOMINATED NODE   READINESS GATES
test-pod  1/1     Running   0          5s    10.244.2.4    desktop-worker <none>         <none>
```

### Checkpoint 2:

**Understand the control loop:**

**Observe: list unscheduled Pods**

**Decide: pick a Node**

**Act: bind the Pod**

Al utilizar el scheduler de polling, el pod va a quedar en pendiente hasta la próxima revisión del scheduler. Una vez que el scheduler revisa los pods pendientes, elige un nodo y le asigna el pod.

## Build and Deploy

```
pame@PameLaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/lab-kubernetes-scheduler$ make kind-load
kind load docker-image my-py-scheduler-polling:latest --name sched-lab
Image: "my-py-scheduler-polling:latest" with ID "sha256:5913cf4078cb04f4b82c4a7dc497c407ad2030adceff18c5df46cc0bc0eaa57a" not yet present on node "sched-lab-worker3", loading...
Image: "my-py-scheduler-polling:latest" with ID "sha256:5913cf4078cb04f4b82c4a7dc497c407ad2030adceff18c5df46cc0bc0eaa57a" not yet present on node "sched-lab-control-plane", loading...
Image: "my-py-scheduler-polling:latest" with ID "sha256:5913cf4078cb04f4b82c4a7dc497c407ad2030adceff18c5df46cc0bc0eaa57a" not yet present on node "sched-lab-worker2", loading...
Image: "my-py-scheduler-polling:latest" with ID "sha256:5913cf4078cb04f4b82c4a7dc497c407ad2030adceff18c5df46cc0bc0eaa57a" not yet present on node "sched-lab-worker", loading...
kind load docker-image my-py-scheduler-watch:latest --name sched-lab
Image: "my-py-scheduler-watch:latest" with ID "sha256:b5da3a26ce2b4349831a72327efb8fdd12d678eabb4165476857bdb6cf7b22bd" not yet present on node "sched-lab-worker3", loading...
Image: "my-py-scheduler-watch:latest" with ID "sha256:b5da3a26ce2b4349831a72327efb8fdd12d678eabb4165476857bdb6cf7b22bd" not yet present on node "sched-lab-control-plane", loading...
Image: "my-py-scheduler-watch:latest" with ID "sha256:b5da3a26ce2b4349831a72327efb8fdd12d678eabb4165476857bdb6cf7b22bd" not yet present on node "sched-lab-worker2", loading...
Image: "my-py-scheduler-watch:latest" with ID "sha256:b5da3a26ce2b4349831a72327efb8fdd12d678eabb4165476857bdb6cf7b22bd" not yet present on node "sched-lab-worker", loading...
```

### Checkpoint 3:

**Your scheduler should log a message like:**

**Bound default/test-pod -> kind-control-plane**

```
stefanie.cibrian@wifialu61-187 py-scheduler-repo.o % kubectl logs -n kube-system deployment/my-scheduler
[polling] scheduler starting... name=my-scheduler
Bound default/test-pod -> desktop-worker2
```

### Checkpoint 4:

**Compare responsiveness and efficiency between polling and watch approaches.**

Al ser ambos muy veloces es difícil comparar cual tiene mejor rendimiento, pero sí podemos establecer, basándonos en la teoría, lo siguiente:

**Para el watch:**

- Tiempos: El cliente mantiene una persistencia en la conexión y recibe eventos que son recibidos inmediatamente al momento de la acción (el

despliegue de un pod es inmediatamente detectado por este scheduler). Lo que ayuda a minimizar la latencia simulando un estado de respuesta de tiempo real.

- Recursos: Utiliza recursos para establecer la conexión una única vez al principio

### Para el polling:

- Tiempo: El cliente chequea cambios en un intervalo de tiempo (frecuencia de polling) Por lo que puede ocurrir un pequeño delay entre que se publica el pod y este es detectado por el scheduler lo que hace al sistema menos reactivo.
- Recursos: Necesita utilizar recursos cada vez que realiza un polling para observar cambios.

### Checkpoint 5:

**Demonstrate your extended policy  
via pod logs and placement.**

```
pame@Pamelaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/py-scheduler-repo/py-scheduler-repo.o$ kubectl describe node sched-lab-worker3
Name: sched-lab-worker3
Roles: <none>
Labels: beta.kubernetes.io/arch=amd64
        beta.kubernetes.io/os=linux
        environment=production
        kubernetes.io/arch=amd64
        kubernetes.io/hostname=sched-lab-worker3
        kubernetes.io/os=linux
Annotations: node.alpha.kubernetes.io/ttl: 0
             volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Mon, 10 Nov 2025 10:48:13 +0100
Taints: environment=production:NoSchedule
Unschedulable: false
...

pame@Pamelaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/py-scheduler-repo/py-scheduler-repo.o$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS GATES
test-prod 1/1     Running   0          11s   10.244.4.3    sched-lab-worker3  <none>           <none>
test1     1/1     Running   0          7m18s 10.244.3.6    sched-lab-worker  <none>           <none>

pame@Pamelaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/py-scheduler-repo/py-scheduler-repo.o$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS GATES
test-prod 1/1     Running   0          5m19s 10.244.4.3    sched-lab-worker3  <none>           <none>
test2     1/1     Running   0          4m26s 10.244.1.4    sched-lab-worker2  <none>           <none>

pame@Pamelaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/py-scheduler-repo/py-scheduler-repo.o$ kubectl logs my-scheduler-66b6687d9-8h5zv -n kube-system
[watch] scheduler starting..name=my-scheduler
Bound by watch default/test-prod -> sched-lab-worker3
Bound by watch default/test2 -> sched-lab-worker2

pame@Pamelaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/py-scheduler-repo/py-scheduler-repo.o$ kubectl logs my-scheduler-66b6687d9-8h5zv -n kube-system
[watch] scheduler starting..name=my-scheduler
Bound by watch default/test-prod -> sched-lab-worker3
Bound by watch default/test2 -> sched-lab-worker2
Bound by watch default/test1 -> sched-lab-worker
pame@Pamelaptop:/mnt/c/Users/pamer/Documents/Master/Cloud computing/py-scheduler-repo/py-scheduler-repo.o$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS GATES
test-prod 1/1     Running   0          7m59s 10.244.4.3    sched-lab-worker3  <none>           <none>
test1     1/1     Running   0          112s   10.244.3.7    sched-lab-worker  <none>           <none>
test2     1/1     Running   0          7m6s   10.244.1.4    sched-lab-worker2  <none>           <none>
```

### Funcionamiento de taints y tolerances

La función `Choose_nodes()` es la responsable de seleccionar el nodo más adecuado para la planificación (scheduling) del Pod pendiente. Este proceso se realiza siguiendo una secuencia de filtrado y puntuación:

#### 1. Fase de Filtrado (Requisitos Obligatorios)

Evaluación de Taints y Toleraciones: Inicialmente, se verifica que el Pod pueda tolerar cualquier taint (restricción) configurado en el nodo. Un nodo es descartado inmediatamente si posee un taint que el Pod no puede tolerar.

Comprobación de Afinidad/Anti-Afinidad: Posteriormente, se evalúan las reglas de afinidad y anti-afinidad (tanto de nodo como de Pod) que el Pod pueda tener definidas. El nodo debe satisfacer estos requisitos obligatorios para poder proceder a la siguiente fase.

## 2. Fase de Puntuación (Mejor Selección)

Los nodos que superan la fase de filtrado pasan a la fase de puntuación (scoring), donde se les asigna un valor.

Criterio de Despliegue de Pods: Uno de los criterios clave en la puntuación es la distribución de la carga. Se prioriza el nodo que tenga menos Pods desplegados, favoreciendo el balanceo de carga entre los recursos disponibles.

Finalmente, el Planificador (Scheduler) elegirá el nodo con la puntuación más alta de entre todos aquellos que cumplieron con éxito los criterios de Taints, Toleraciones y Afinidad/Anti-Afinidad.