Alejandro Goicochea

Tarea 2 – Kubernetes

1. Deploying Kubernetes

1.1 minikube start --nodes 1 single-node

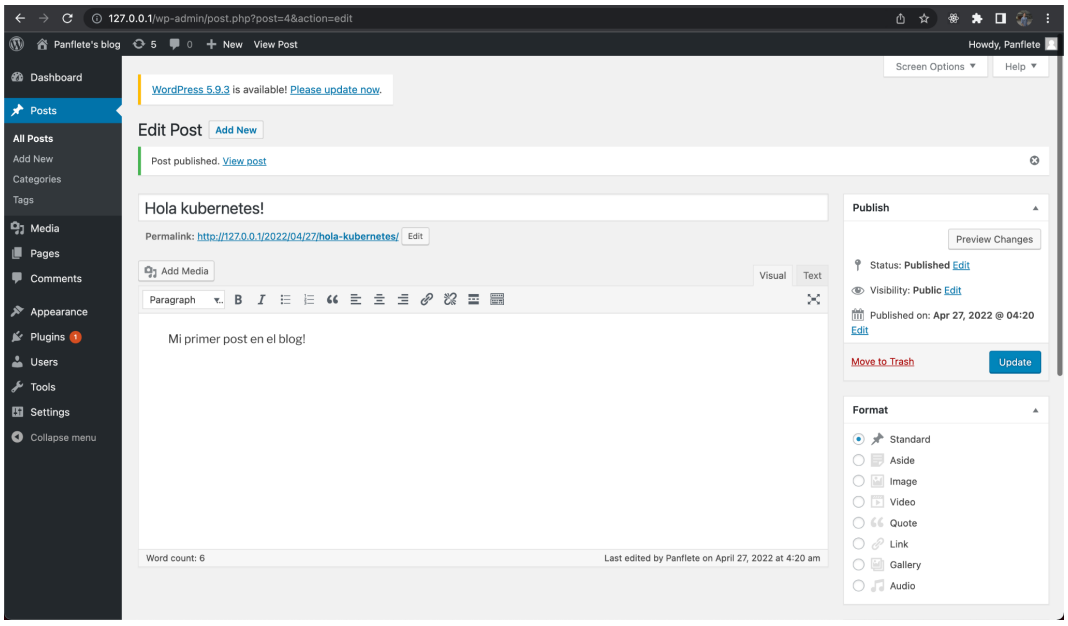
```
panflete@DESKTOP-7TU7QBB:~$ kubectl get nodes -o wide
NAME       STATUS   ROLES                  AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION                 CONTAINER-RUNTIME
minikube   Ready    control-plane,master   22s   v1.23.3   192.168.49.2   <none>        Ubuntu 20.04.2 LTS   5.4.72-microsoft-standard-WSL2   docker://20.10.12
```

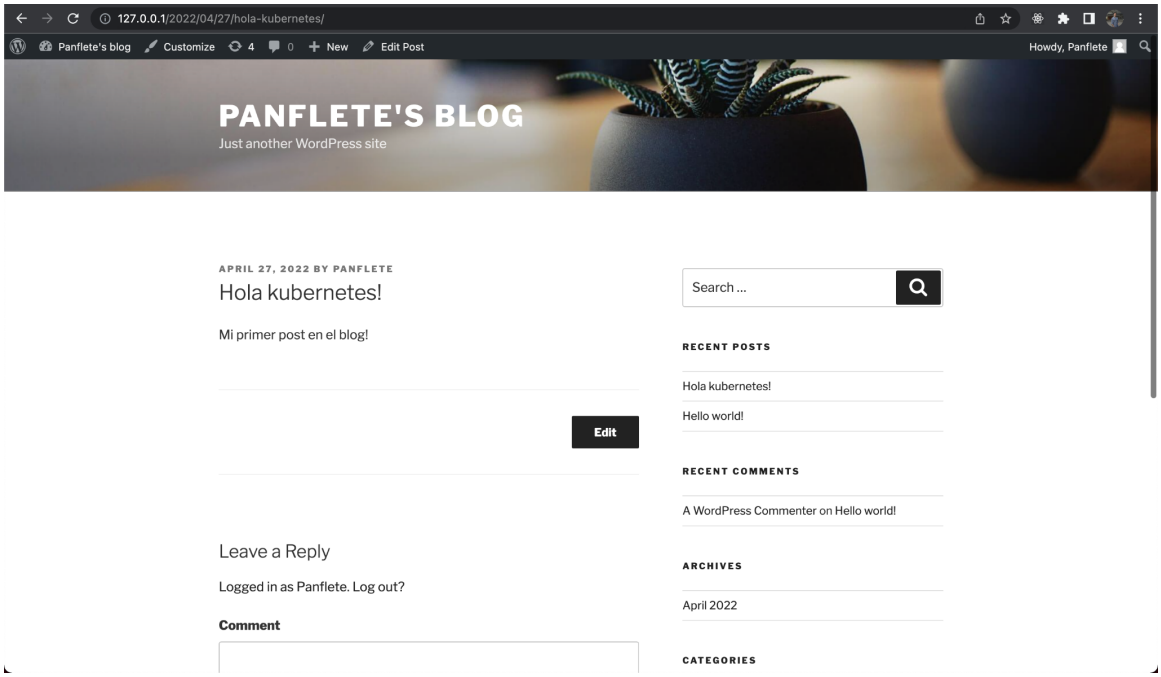
1.2 minikube start --nodes 2 multi-node

```
panflete@DESKTOP-7TU7QBB:~$ kubectl get nodes -o wide
NAME           STATUS   ROLES                  AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION                 CONTAINER-RUNTIME
minikube       Ready    control-plane,master   54s   v1.23.3   192.168.49.2   <none>        Ubuntu 20.04.2 LTS   5.4.72-microsoft-standard-WSL2   docker://20.10.12
minikube-m02   Ready    <none>                 27s   v1.23.3   192.168.49.3   <none>        Ubuntu 20.04.2 LTS   5.4.72-microsoft-standard-WSL2   docker://20.10.12
```

2. Search and select a containerized application

La aplicación consiste de un frontend hecho en wordpress y una base de datos hecha con mysql. El blog tiene una funcionalidad simple permite publicar, editar y eliminar posts en los que se puede dejar comentarios. En las siguientes imágenes se muestra como se crea un post en el blog y como se ve una vez publicado.





3. Deployment a containerized application on kubernetes

3.1

Primero se hizo un deployment básico con un solo nodo. Se utilizaron los archivos yaml en el repositorio y se deployeo a través de un kustomization file. Para hacerlo solo hace falta ejecutar el comando

kubectl apply -f ./

Este comando apunta al directorio donde se encuentra el archivo con la configuración (en este caso se encuentra en el directorio actual.) Podemos ver los pods dentro del mismo nodo en la siguiente imagen:

```
[panflete@Alejandros-MacBook-Pro cloud-tarea-2 % kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS       AGE    IP            NODE
wordpress-56bd9b8899-6dplq        1/1     Running   1 (2m5s ago)   3m8s   172.17.0.5    minikube
wordpress-56bd9b8899-96dxt        1/1     Running   1 (2m6s ago)   3m8s   172.17.0.4    minikube
wordpress-56bd9b8899-szcxp        1/1     Running   1 (2m6s ago)   3m8s   172.17.0.3    minikube
wordpress-mysql-5574486cb7-wr46b  1/1     Running   0              3m8s   172.17.0.6    minikube
```

Para el caso del ambiente multinodo, tenemos que modificar el archivo wordpress-deployment.yaml para agregar antiaffinity a los pods. Esta característica hace que los pods deciden a que nodo pertenecer basado en que pods se encuentran actualmente en ese nodo. La modificación se puede ver en la siguiente imagen:

```yaml
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions: [{ key: app, operator: In, values: [wordpress] }]
        topologyKey: "kubernetes.io"
```

Luego de ejecutar nuestro archivo de kustomization, revisamos los pods nuevamente y podemos ver que se han repartido entre los dos nodos.

```
panflete@Alejandros-MacBook-Pro Cloud % kubectl get pods -o wide
NAME                               READY   STATUS              RESTARTS   AGE   IP       NODE
wordpress-764b859d48-f2v9l         0/1     ContainerCreating   0          10s   <none>   minikube-m02
wordpress-764b859d48-m7b7d         0/1     ContainerCreating   0          11s   <none>   minikube-m02
wordpress-764b859d48-sgw4k         0/1     ContainerCreating   0          10s   <none>   minikube
wordpress-mysql-5574486cb7-9p75v   0/1     ContainerCreating   0          10s   <none>   minikube-m02
```

Los dos primeros pods se encuentran en el nodo minikube-m02 mientras que el tercero se encuentra en el nodo minikube.

Finalmente en ambos casos, para poder acceder a la aplicación, ejecutamos *minikube tunnel* Y podemos visualizar la aplicación en nuestro navegador.

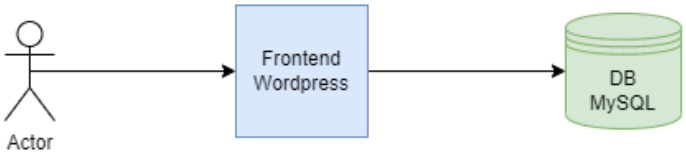
```
panflete@Alejandros-MacBook-Pro Cloud % minikube tunnel
✅  Tunnel successfully started

📌  NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...

❗  The service/ingress wordpress requires privileged ports to be exposed: [80]
🔑  sudo permission will be asked for it.
🏃  Starting tunnel for service wordpress.
```

3.2

Flujo Alto Nivel



Flujo Bajo Nivel