SysOps/DevOps Polska

Bezpieczeństwo Kubernetesa

Mariusz Dalewski

Zadanie 4: Instalacja Kubernetesa - kind bez domyślnego cni

Omówienie

Zadanie ma celu zaprezentowanie nieprodukcyjnej metody instalacji Kubernetesa przy pomocy pakietu kind.

Wymagania początkowe

- System operacyjny Linux z działającym dockerem.

Ćwiczenie

1. Logujemy się do systemu, w który będziemy uruchamiać Kubernetesa.
2. Instalujemy `kind` (zgodnie z instrukcją na stronie https://kind.sigs.k8s.io/docs/user/quick-start/#installation), np.:

- Linux - binary - wchodzimy na adres https://github.com/kubernetes-sigs/kind/releases i pobieramy plik `kind-linux-amd64` z ostatniego wydania, zapisujemy go jako `kind` oraz zmieniamy mu uprawnienia `chmod 755 kind`
- Linux - LinuxBrew - `brew install kind`
- macOS - MacPorts - `sudo port selfupdate && sudo port install kind`
- macOS - Homebrew - `brew install kind`.

3. Jeżeli na danym użytkowniku nie posiadamy uprawnień do dockera (np.: `docker ps`) to podnosimy uprawnienia do użytkownika root: `sudo -i`.
4. Pobieramy z repozytorium https://github.com/so-do/kubernetes plik `cluster.yaml` (katalog **bezpieczenstwo-k8s/day2**).
5. Wykonujemy polecenie `kind create cluster --config cluster.yaml`.

Oczekiwany rezultat:

```
sodo@sodo:~# kind create cluster --config ../cluster.yaml
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.25.3) 🖼
 ✓ Preparing nodes 📦 📦 📦 📦 📦 📦
 ✓ Configuring the external load balancer ⚖️
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing StorageClass 💾
 ✓ Joining more control-plane nodes 🎮
 ✓ Joining worker nodes 🚜
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Have a question, bug, or feature request? Let us know!
https://kind.sigs.k8s.io/#community 🙂
```

6. Instalujemy narzędzie `kubectl` zgodnie z manualem - https://kubernetes.io/docs/tasks/tools/install-kubectl/ (Google: kubectl).

7. Weryfikujemy działanie klastra komendami `kubectl version`, `kubectl get pods -A` oraz `kubectl get nodes`.

Oczekiwany rezultat:

```
sodo@sodo:~# kubectl version
WARNING: This version information is deprecated and will be replaced
with the output from kubectl version --short.  Use --output=yaml|json
to get the full version.
Client Version: version.Info{Major:"1", Minor:"24",
GitVersion:"v1.24.2",
GitCommit:"f66044f4361b9f1f96f0053dd46cb7dce5e990a8",
GitTreeState:"clean", BuildDate:"2022-06-15T14:14:10Z",
GoVersion:"go1.18.3", Compiler:"gc", Platform:"linux/amd64"}
Kustomize Version: v4.5.4
Server Version: version.Info{Major:"1", Minor:"25",
GitVersion:"v1.25.3",
GitCommit:"434bfd82814af038ad94d62ebe59b133fcb50506",
GitTreeState:"clean", BuildDate:"2022-10-25T19:35:11Z",
GoVersion:"go1.19.2", Compiler:"gc", Platform:"linux/amd64"}
sodo@sodo:~# kubectl get pods -A
NAMESPACE            NAME
READY   STATUS    RESTARTS   AGE
kube-system          coredns-565d847f94-cw24l                      1/1
Running   0          3m5s
kube-system          coredns-565d847f94-kdjtn                      1/1
```

```
Running   0            3m5s
kube-system          etcd-kind-control-plane                         1/1
Running   0            3m13s
kube-system          etcd-kind-control-plane2                        1/1
Running   0            2m38s
kube-system          etcd-kind-control-plane3                        1/1
Running   0            2m20s
kube-system          kindnet-6q79z                                   1/1
Running   0            3m5s
kube-system          kindnet-7cfdp                                   1/1
Running   0            2m7s
kube-system          kindnet-bzscx                                   1/1
Running   0            2m7s
kube-system          kindnet-dvzn5                                   1/1
Running   0            2m49s
kube-system          kindnet-hhgms                                   1/1
Running   0            2m7s
kube-system          kindnet-mwmgz                                   1/1
Running   0            2m24s
kube-system          kube-apiserver-kind-control-plane               1/1
Running   0            3m14s
kube-system          kube-apiserver-kind-control-plane2              1/1
Running   0            2m31s
kube-system          kube-apiserver-kind-control-plane3              1/1
Running   0            2m20s
kube-system          kube-controller-manager-kind-control-plane      1/1
Running   0            3m14s
kube-system          kube-controller-manager-kind-control-plane2     1/1
Running   0            2m44s
kube-system          kube-controller-manager-kind-control-plane3     1/1
Running   0            2m18s
kube-system          kube-proxy-6dcf6                                1/1
Running   0            3m5s
kube-system          kube-proxy-6swgl                                1/1
Running   0            2m7s
kube-system          kube-proxy-nz6jb                                1/1
Running   0            2m24s
kube-system          kube-proxy-psvjj                                1/1
Running   0            2m7s
kube-system          kube-proxy-rk952                                1/1
Running   0            2m7s
kube-system          kube-proxy-skbrp                                1/1
Running   0            2m49s
kube-system          kube-scheduler-kind-control-plane               1/1
Running   0            3m13s
```

```
kube-system          kube-scheduler-kind-control-plane2            1/1
Running   0          2m38s
kube-system          kube-scheduler-kind-control-plane3            1/1
Running   0          76s
local-path-storage   local-path-provisioner-684f458cdd-6zx4c       1/1
Running   0          3m5s
sodo@sodo:~# kubectl get nodes
NAME                  STATUS     ROLES           AGE     VERSION
kind-control-plane    NotReady   control-plane   2m13s   v1.25.3
kind-control-plane2   NotReady   control-plane   106s    v1.25.3
kind-control-plane3   NotReady   control-plane   81s     v1.25.3
kind-worker           NotReady   <none>          64s     v1.25.3
kind-worker2          NotReady   <none>          64s     v1.25.3
kind-worker3          NotReady   <none>          64s     v1.25.3
```

8. Instalujemy sterownik sieciowy Calico `kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.24.0/manifests/tigera-operator.yaml` `kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.24.0/manifests/custom-resources.yaml`, a na koniec weryfikujemy czy wszystkie komponenty zostały uruchomione - `watch kubectl get pods -n calico-system`

9. Ponownie uruchamiamy `kubectl get nodes`

Zadanie 5: Testy systemu Network Policy

Omówienie

Zadanie ma celu zaprezentowanie metod konfiguracji systemu Network Policy.

Wymagania początkowe

- Zrealizowanie zadanie nr 4

Ćwiczenie

1. Przechodzimy do katalogu **bezpieczenstwo-k8s/day2/network-policy**.
2. Wykonujemy komendę `kubectl apply -f pod.yaml`.
3. Wchodzi do poda `kubectl exec -it test-app` i sprawdzamy `ping onet.pl` oraz `ping wp.pl`.
4. Wykonujemy komendę `kubectl apply -f network-policy.yaml`.
5. Ponownie wykonujemy test z wew. poda (pkt 3).

Zadanie 6: Testy systemu POD Security Admissions

Omówienie

Zadanie ma celu zaprezentowanie metod konfiguracji systemu PSA.

Wymagania początkowe

- Zrealizowanie zadanie nr 4

Ćwiczenie

1. Przechodzimy do katalogu **bezpieczenstwo-k8s/day2/pod-security-admission**.
2. Wykonujemy komendę `kubectl apply -f ns.yaml`.
3. Wykonujemy komendę `kubectl apply -f pod.yaml`.
4. Wykonujemy komendę `kubectl delete -f pod.yaml` oraz `kubectl apply -f pod-v2.yaml`.