Version - **2024.01**

Dernière mise-à-jour : 2020/05/27 13:51

DOF305 - Gestion du Réseau, des Services et d'une Architecture de Microservices

Contenu du Module

Ressources

LAB #1

- https://www.dropbox.com/scl/fi/rqxq22c6fxgr2zivf02au/clusterip-example.yaml?rlkey=q79w12mcfj5jaj48j7kl4gv3p&dl=0
- https://www.dropbox.com/scl/fi/d105k0mjg4guwn1gg2sr6/clusterip-service.yaml?rlkey=bvvdhihx85p5n6bct0cdy5jlg&dl=0
- https://www.dropbox.com/scl/fi/9zgm5sgh8l8f5jhde9e3z/clusterippod.yaml?rlkey=crh5jpt5om0ekcibjrd8ec64y&dl=0
- https://www.dropbox.com/scl/fi/pt62nfxzi9tsa0fj8hfza/dnstest.yaml?rlkey=qob9301qplpplt6v2tm3x1l7j&dl=0
- https://www.dropbox.com/scl/fi/wtx04mm4um741dlj9wmj7/myingress.yaml?rlkey=mqeggcq8ccms9nv1zunk43kuh&dl=0
- https://www.dropbox.com/scl/fi/nvud4cx3jcya5e5ji188u/mynetworkpolicy.yaml?rlkey=osrpfjrxietbrrcoalflhmbl5&dl=0
- https://www.dropbox.com/scl/fi/k84yyq96t7hnigo8q66qs/npbusybox.yaml?rlkey=ehq8qowb04r26s6jfk3qkl4cz&dl=0
- https://www.dropbox.com/scl/fi/ivq5emll325nwj9yzjuam/npnginx.yaml?rlkey=sk1rso41e3wrou5y4iy024xdq&dl=0
- https://www.dropbox.com/scl/fi/3cp23paw353zpllsily8/service-definition.yaml?rlkey=oe5sfo9soa6q25a8mjqw7ax59&dl=0

LAB #2

- https://www.dropbox.com/scl/fi/c87nyp8f2o9vh64pifcmy/postgres-deployment.yaml?rlkey=bu3n6i0372131q9qzonry6kal&dl=0
- https://www.dropbox.com/scl/fi/qionkk9d5lj5cqbkqpg9x/postgres-service.yaml?rlkey=h4smnpd1afkyscx8eg9sanh7h&dl=0
- https://www.dropbox.com/scl/fi/o00mmelwwhx0ytkjq7kvl/redis-deployment.yaml?rlkey=2ne90svzrmzne619mtxswwi3e&dl=0
- https://www.dropbox.com/scl/fi/l0j16x1ais5686u8qaesf/redis-service.yaml?rlkey=t3sezo8is3pu34vmjoq1zw4ug&dl=0
- https://www.dropbox.com/scl/fi/ap63boqbt0mot16sx3fva/result-app-deployment.yaml?rlkey=5epq45fioqdkecueo5fcwn2h8&dl=0
- https://www.dropbox.com/scl/fi/qxo4g3bim0bc1v537tnse/result-app-service.yaml?rlkey=u7ryslr2lf25m9ibl4t7yujux&dl=0

- https://www.dropbox.com/scl/fi/uinl9q5h1uqkkva9txad3/voting-app-deployment.yaml?rlkey=9os74agx9tljxcg44hwas917f&dl=0
- https://www.dropbox.com/scl/fi/yo29xrt2h4414tl0z9pk9/voting-app-service.yaml?rlkey=h36b4xocyhjvkjosntmpu3bha&dl=0
- https://www.dropbox.com/scl/fi/3cwnbhext63brqqit7pzx/worker-deployment.yaml?rlkey=6u8elahie7ah3hqgj2cksnx75&dl=0

LAB #1 - Gestion du Réseau et des Services

1.1 - Présentation des Extensions Réseau

Kubernetes impose des conditions pour l'implémentation d'un réseau :

- Les PODs sur un nœud peuvent communiquer avec tous les PODs sur tous le nœuds sans utiliser NAT,
- Les agents sur un nœud (par exemple kubelet) peuvent communiquer avec tous les PODs sur le nœud.

Important : La description technique et détaillée de l'approche réseau de Kubernetes peut être consultée à l'adresse : **https://kubernetes.io/docs/concepts/cluster-administration/networking/**.

Lors de l'installation du cluster nous avons spécifié l'utilisation d'une extension réseau appelée **Calico**, issue de la liste suivante :

- **Calico**,
- **Cilium**,
- **Flannel**,
- **Kube-router**,
- **Romana**,
- **WeaveNet**,
- **Antrea**,
- **kube-ovn**,
- Canal (utilise Flannel pour le réseau et Calico pour le pare-feu).

Important : Une étude comparative des extensions réseau pour Kubernetes peut être trouvée à la page : **https://itnext.io/benchmark-results-of-kubernetes-network-plugins-cni-over-10gbit-s-network-updated-august-2020-6e1b757b9e49**.

1.2 - DNS K8s

Présentation

Les services DNS du cluster utilisant le plugin **Calico** sont fournis par **CoreDNS** :

```
root@kubemaster:~# kubectl get deployments -n kube-system
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
calico-kube-controllers   1/1     1            1           12d
coredns                   2/2     2            2           12d
metrics-server            1/1     1            1           11d

root@kubemaster:~# kubectl get service -n kube-system
NAME             TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)                  AGE
kube-dns         ClusterIP   10.96.0.10    <none>        53/UDP,53/TCP,9153/TCP   12d
metrics-server   ClusterIP   10.98.89.81   <none>        443/TCP                  11d
```

Tous les pods sont attribués un nom d'hôte au format suivant :

```
adresse_ip_du_pod_sous_le_format_xxx-xxx-xxx-xxx.nom_namespace.pod.cluster.local
```

Mise en Application

Pour tester le DNS, créez le fichier **dnstest.yaml** :

```
root@kubemaster:~# vi dnstest.yaml
root@kubemaster:~# cat dnstest.yaml
apiVersion: v1
kind: Pod
metadata:
  name: busybox-dnstest
spec:
  containers:
    - name: busybox
      image: radial/busyboxplus:curl
      command: ['sh', '-c', 'while true; do sleep 3600; done']
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-dnstest
spec:
  containers:
  - name: nginx
    image: nginx:1.19.2
    ports:
    - containerPort: 80
```

Important : Notez que ce fichier va créer deux pods - **busybox-dnstest** et **nginx-dnstest**.

Créez les deux pods à l'aide de ce fichier :

```
root@kubemaster:~# kubectl create -f dnstest.yaml
pod/busybox-dnstest created
pod/nginx-dnstest created
```

Copiez l'adresse IP du pod **nginx-test** :

```
root@kubemaster:~# kubectl get pods nginx-dnstest -o wide
NAME            READY   STATUS    RESTARTS   AGE   IP               NODE                         NOMINATED NODE
READINESS GATES
nginx-dnstest   1/1     Running   0          48s   192.168.150.33   kubenode2.ittraining.loc   <none>
<none>
```

Exécutez la commande **curl <adresse IP copiée>** dans le conteneur du pod **busybox-dnstest** :

```
root@kubemaster:~# kubectl exec busybox-dnstest -- curl 192.168.150.33
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
```

```
</body>
</html>
100   612  100   612    0     0   533k      0 --:--:-- --:--:-- --:--:--  597k
```

Important : Notez que **busybox-dnstest** a pu contacter **nginx-dnstest** en utilisant son adresse IP.

Utilisez maintenant le DNS K8s pour résoudre le nom d'hôte du pod **nginx-dnstest** :

```
root@kubemaster:~# kubectl exec busybox-dnstest -- nslookup 192-168-150-33.default.pod.cluster.local
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      192-168-150-33.default.pod.cluster.local
Address 1: 192.168.150.33
```

Important : Notez que le nom d'hôte a été résolu grâce au DNS K8s.

Exécutez maintenant la commande **curl <nom_d_hote_du_pod_nginx_dnstest>** dans le conteneur du pod **busybox-dnstest** :

```
root@kubemaster:~# kubectl exec busybox-dnstest -- curl 192-168-150-33.default.pod.cluster.local
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
...
<title>Welcome to nginx!</title>
...
100   612  100   612    0     0   355k      0 --:--:-- --:--:-- --:--:--  597k
```

Important : Notez que **busybox-dnstest** a pu contacter **nginx-dnstest** en utilisant son nom d'hôte.

1.3 - Network Policies

Présentation

Un **NetworkPolicy** est un objet K8s qui permet de contrôler la communication vers et à partir des pods.

Les composants d'un NetworkPolicy sont :

- **from et to Selectors**,
  - le **from selector** s'opère sur le trafic **Ingress**,
    - le mot Ingress indique du trafic réseau vers un pod,
  - le **to selector** s'opère sur le trafic **Egress**,
    - le mot Egress indique du trafic reçu d'un pod.

Les from et to Selectors utilisent des **Types** :

- **podSelector**,
  - un podSelector peut sélectionner des pods en utilisant des Labels (*étiquettes en français*),
  - par défaut, un pod n'est pas isolé dans le cluster. Par contre dès qu'un podSelector sélectionne un pod, celui-ci est considéré comme isolé et ne peut que communiquer en suivant les **NetworkPolicies**,
- **namespaceSelector**,
  - un namespaceSelector peut sélectionner des nameSpaces en utilisant des Labels (*étiquettes en français*),
- **ipBlock**,
  - un IPBlock peut sélectionner des pods en utilisant une plage d’adresses IP au format CIDR.

En complément des Types ci-dessus, il est aussi possible de spécifier :

- **Ports**,
  - les ports spécifient le numéro de port ainsi que le protocole,
  - le trafic réseau n'est accepté que dans le cas où les règles spécifiées par le Type **et** le port/protocole sont satisfaits.

Mise en Application

Pour mieux comprendre, créez un NameSpace dénommé **nptest** :

```
root@kubemaster:~# kubectl create namespace nptest
namespace/nptest created
```

Etiquettez ce NameSpace :

```
root@kubemaster:~# kubectl label namespace nptest lab=nptest
namespace/nptest labeled
```

Important : Notez l'étiquette **lab=nptest**.

Créez maintenant le fichier **npnginx.yaml** :

```
root@kubemaster:~# vi npnginx.yaml
root@kubemaster:~# cat npnginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: npnginx
  namespace: nptest
  labels:
    app: nginx
spec:
```

```
  containers:
  - name: nginx
    image: nginx
```

Important : Notez l'étiquette **app: nginx**.

Creéz le pod npnginx :

```
root@kubemaster:~# kubectl create -f npnginx.yaml
pod/npnginx created
```

Créez maintenant le fichier **npbusybox.yaml** :

```
root@kubemaster:~# vi npbusybox.yaml
root@kubemaster:~# cat npbusybox.yaml
apiVersion: v1
kind: Pod
metadata:
  name: npbusybox
  namespace: nptest
  labels:
    app: client
spec:
  containers:
    - name: busybox
      image: radial/busyboxplus:curl
      command: ['sh', '-c', 'while true; do sleep 5; done']
```

Important : Notez l'étiquette **app: client**.

Creéz le pod **npbusybox** :

```
root@kubemaster:~# kubectl create -f npbusybox.yaml
pod/npbusybox created
```

Visualisez les informations des deux pods créés :

```
root@kubemaster:~# kubectl get pods -n nptest -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP               NODE                         NOMINATED NODE
READINESS GATES
npbusybox   1/1     Running   0          48s     192.168.150.35   kubenode2.ittraining.loc   <none>
<none>
npnginx     1/1     Running   0          4m13s   192.168.239.33   kubenode1.ittraining.loc   <none>
<none>
```

Copiez l'adresse IP du pode **npnginx** et créez une variable appelée **NGINX_IP** :

```
root@kubemaster:~# NGINX_IP=192.168.239.33

root@kubemaster:~# echo $NGINX_IP
192.168.239.33
```

Testez la communication entre **npbusybox** et **npnginx** :

```
root@kubemaster:~# kubectl exec -n nptest npbusybox -- curl $NGINX_IP
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   615  100   615    0     0  78977      0 --:--:-- --:--:-- --:--:-- 87857
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
</head>
```

```
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Important : Rappelez-vous : par défaut, un pod n'est pas isolé dans le cluster. La communication a donc réussi.

Créez maintenant le fichier **mynetworkpolicy.yaml** :

```
root@kubemaster:~# vi mynetworkpolicy.yaml
root@kubemaster:~# cat mynetworkpolicy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: mynetworkpolicy
  namespace: nptest
spec:
  podSelector:
    matchLabels:
      app: nginx
  policyTypes:
```

```
    - Ingress
    - Egress
```

Important : Notez l'étiquette **app: nginx**. La policy s'applique donc au pod **npnginx**.

Créez maintenant la NetworkPolicy :

```
root@kubemaster:~# kubectl create -f mynetworkpolicy.yaml
networkpolicy.networking.k8s.io/mynetworkpolicy created
```

Testez de nouveau la communication entre **npbusybox** et **npnginx** :

```
root@kubemaster:~# kubectl exec -n nptest npbusybox -- curl $NGINX_IP
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:24 --:--:--     0^C
```

Important : Notez que la NetworkPolicy bloque la communication. Notez aussi l'utilisation de **^C** pour terminer le processus.

Editez maintenant la NetworkPolicy :

```
root@kubemaster:~# kubectl edit networkpolicy -n nptest mynetworkpolicy

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  creationTimestamp: "2022-09-16T13:24:29Z"
  generation: 1
  name: mynetworkpolicy
  namespace: nptest
  resourceVersion: "1490105"
  uid: b130f09f-2ab1-4dc6-9059-95f900234be3
spec:
  podSelector:
    matchLabels:
      app: nginx
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          lab: nptest
    ports:
    - protocol: TCP
      port: 80
status: {}
:wq

root@kubemaster:~# kubectl edit networkpolicy -n nptest mynetworkpolicy
networkpolicy.networking.k8s.io/mynetworkpolicy edited
```

Important : Notez la création de la règle **ingress**. Cette règle utilise un namespaceSelector pour permettre du trafic à partir de pods dans un NameSpace ayant une étiquette **lab: nptest**. La règle ports permet le trafic sur le port 80/tcp.

Testez de nouveau la communication entre **npbusybox** et **npnginx** :

```
root@kubemaster:~# kubectl exec -n nptest npbusybox -- curl $NGINX_IP
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   615  100   615    0     0   531k      0 --:--:-- --:--:-- --:--:--  600k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Important : Notez que la communication a réussi.

1.4 - Services

Présentation

Les services de K8s sont :

- NodePort,
  - Ce Service rend un POD accessible sur un port du nœud le contenant,
- ClusterIP
  - Ce Service crée une adresse IP virtuelle afin de permettre la communication entre de services différents dans le cluster, par exemple des serveurs front-end avec des serveurs back-end,
- LoadBalancer
  - Ce service provisionne un équilibrage de charge pour une application dans certains fournisseurs de Cloud publique tels **A**mazon **W**eb **S**ervices et **G**oogle **C**loud **P**latform.
- ExternalName
  - Ne fait pas parti de la certification CKA.

Mise en Application

Commencez par créer le deployment **myapp-deployment** :

```
root@kubemaster:~# kubectl create -f deployment-definition.yaml
deployment.apps/myapp-deployment created
```

Constatez l'état des pods :

```
root@kubemaster:~# kubectl get pods -o wide
NAME                              READY   STATUS    RESTARTS        AGE    IP               NODE
NOMINATED NODE   READINESS GATES
busybox-dnstest                   1/1     Running   0               4h9m   192.168.150.34
kubenode2.ittraining.loc   <none>           <none>
```

```
myapp-deployment-7c4d4f7fc6-2km9n      1/1     Running   0                83s    192.168.239.34
kubenode1.ittraining.loc   <none>           <none>
myapp-deployment-7c4d4f7fc6-7pts7      1/1     Running   0                83s    192.168.239.35
kubenode1.ittraining.loc   <none>           <none>
myapp-deployment-7c4d4f7fc6-9pw5x      1/1     Running   0                83s    192.168.150.36
kubenode2.ittraining.loc   <none>           <none>
mydaemonset-hmdhp                      1/1     Running   1 (7h29m ago)    23h    192.168.239.32
kubenode1.ittraining.loc   <none>           <none>
mydaemonset-kmf4z                      1/1     Running   1                23h    192.168.150.31
kubenode2.ittraining.loc   <none>           <none>
nginx-dnstest                          1/1     Running   0                4h9m   192.168.150.33
kubenode2.ittraining.loc   <none>           <none>
```

Important : Notez que les adresses **192.168.239.x** sont associées aux PODs sur kubenode1 tandis que les adresses **192.168.150.x** sont associées aux PODs sur kubenode2. Ces adresses sont issues du réseau **192.168.0.0/16** stipulé par l'option **–pod-network-cidr** lors de l'initialisation du maître du cluster.

En sachant que dans chaque POD existe un conteneur Nginx, testez si vous pouvez afficher la page d'accueil de Nginx en vous connectant à kubenode1 et kubenode2 à partir de votre Gateway :

```
trainee@kubemaster:~$ exit
déconnexion
Connection to 10.0.2.65 closed.
trainee@gateway:~$ curl 192.168.56.3
curl: (7) Failed to connect to 192.168.56.3 port 80: Connection refused
trainee@gateway:~$ curl 192.168.56.4
curl: (7) Failed to connect to 192.168.56.4 port 80: Connection refused
```

Important : Notez l'échec de la connexion.

Testez maintenant si vous pouvez afficher la page d'accueil de Nginx en vous connectant à un des PODs **à partir de votre Gateway** :

```
trainee@gateway:~$ curl 192.168.239.34
^C
```

Connectez-vous à **kubemaster** :

```
trainee@gateway:~$ ssh -l trainee 192.168.56.2
trainee@192.168.56.2's password: trainee
Linux kubemaster.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jul 13 15:45:46 2022 from 10.0.2.40
trainee@kubemaster:~$ su -
Mot de passe : fenestros
root@kubemaster:~#
```

Bien évidement, il est possible d'afficher la page en vous connectant à un des PODs de **l'intérieur** du cluster :

```
root@kubemaster:~# curl 192.168.239.34
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
```

```
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

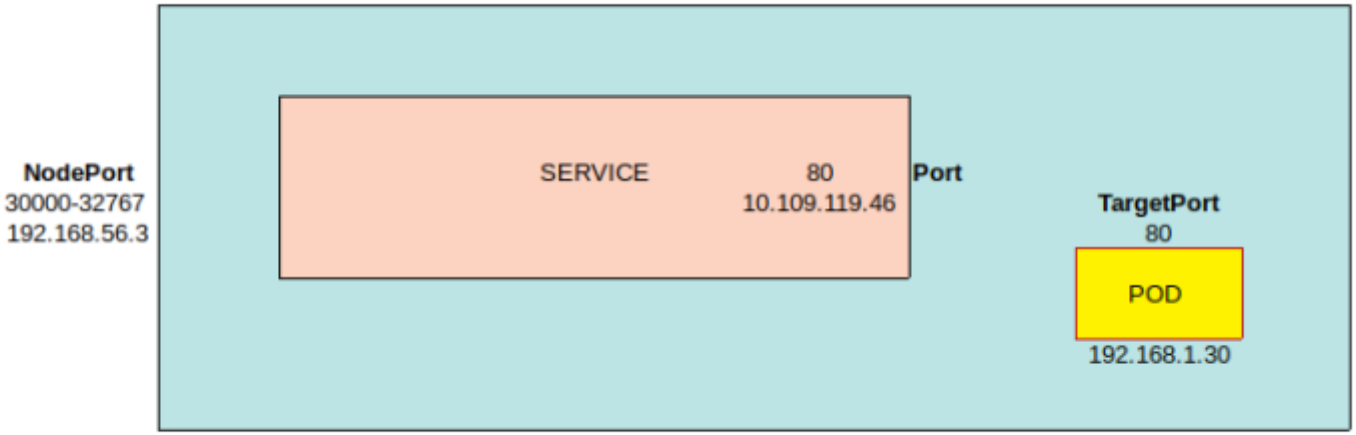
Important : Retenez donc qu'à ce stade il n'est pas possible d'afficher la page d'accueil de Nginx en vous connectant de l'extérieur du cluster.

Le Service NodePort

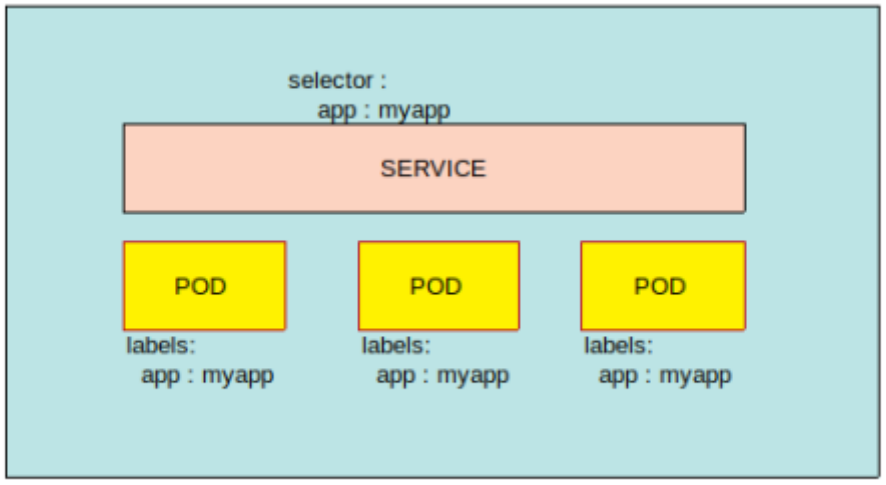
Présentation

Le Service NodePort définit trois ports :

- **TargetPort** : le port sur le POD,
- **Port** : le port sur le Service lié à un IP du Cluster,
- **NodePort** : le port sur le Nœud issu de la plage 30000-32767.

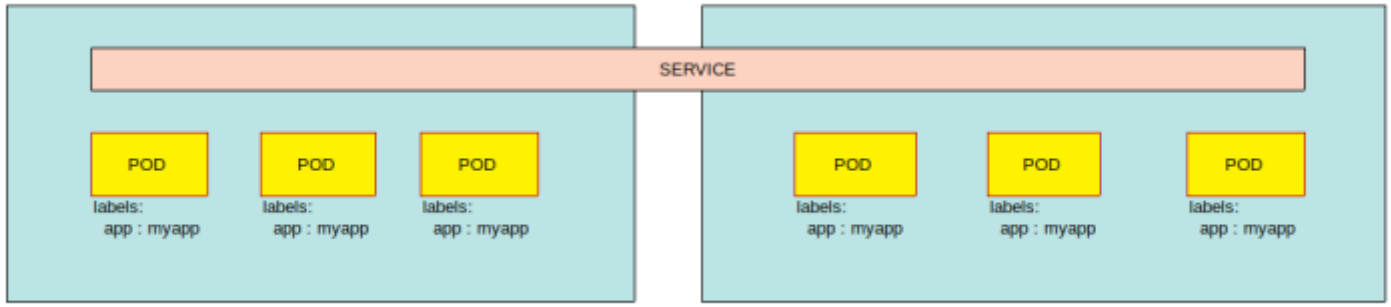


Si dans le même nœud, plusieurs PODs ont les étiquettes qui correspondent au **selector** du Service, le Service identifie les PODs et s'étend automatiquement pour englober tous les PODs. Les PODs sont appelés des **End-Points** :



Important : Notez que dans ce cas l'équilibrage de charge est automatique est utilise l'algorithme **Random** avec une affinité de session..

De même, quand les PODs sont distribués sur plusieurs nœuds, le Service s'étend pour tout englober :



Mise en Application

Créez donc le fichier YAML **service-definition.yaml** :

```
root@kubemaster:~# vi service-definition.yaml
root@kubemaster:~# cat service-definition.yaml
apiVersion: v1
kind: Service
metadata:
  name: myapp-service

spec:
  type: NodePort
  ports:
    - targetPort: 80
```

```
      port: 80
      nodePort: 30008
  selector:
    app: myapp
    type: front-end
```

Important : Notez que si le champ **type:** est manquant, sa valeur par défaut est **ClusterIP**. Notez aussi que dans **ports**, seul le champ **port** est obligatoire. Si le champ **targetPort** est manquant, sa valeur par défaut est celle du champ **port**. Si le champ **nodePort** est manquant, sa valeur par défaut est le premier port disponible dans la plage entre **30 000** et **32 767**. Dernièrement, il est possible de spécifier de multiples définitions de ports dans le service.

Le champs **selector** contient les étiquettes des PODs concernés par la mise en place du Service :

```
root@kubemaster:~# cat pod-definition.yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Créez le Service en utilisant le fichier **service-definition.yaml** :

```
root@kubemaster:~# kubectl create -f service-definition.yaml
```

```
service/myapp-service created
```

Constatez la création du Service :

```
root@kubemaster:~# kubectl get services
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1      <none>        443/TCP        26h
myapp-service   NodePort    10.97.228.14   <none>        80:30008/TCP   13s
```

Important : Notez que le Service a une adresse IP du cluster et qu'il a exposé le port **30008**.

Testez maintenant si vous pouvez afficher la page d'accueil de Nginx en vous connectant à un des PODs à partir de votre Gateway en utilisant le port exposé :

```
root@kubemaster:~# exit
déconnexion

trainee@kubemaster:~$ exit
déconnexion
Connection to 192.168.56.2 closed.

trainee@gateway:~$ curl 192.168.56.3:30008
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
```

```
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>

trainee@gateway:~$ curl 192.168.56.4:30008
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
```

```
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Le Service ClusterIP

Présentation

Le Service **ClusterIP** permet de regrouper les PODs offrant le même service afin de faciliter la communication entre pods à l'intérieur du cluster.

Mise en Application

Pour créer un Service ClusterIP, créez le fichier **clusterip-example.yaml** :

```
root@kubemaster:~# vi clusterip-example.yaml
root@kubemaster:~# cat clusterip-example.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploymentclusterip
spec:
  replicas: 3
  selector:
    matchLabels:
```

```
      app: clusteripexample
  template:
    metadata:
      labels:
        app: clusteripexample
    spec:
      containers:
      - name: nginx
        image: nginx:1.19.1
        ports:
        - containerPort: 80
```

Créez un deployment en utilisant le fichier **clusterip-example.yaml** :

```
root@kubemaster:~# kubectl create -f clusterip-example.yaml
deployment.apps/deploymentclusterip created
```

Creéz maintenant un service de type ClusterIP pour exposer les pods du deplyment **deploymentclusterip** :

```
root@kubemaster:~# vi clusterip-service.yaml
root@kubemaster:~# cat clusterip-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: clusteripservice
spec:
  type: ClusterIP
  selector:
    app: clusteripexample
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Créez un service en utilisant le fichier **clusterip-service.yaml** :

```
root@kubemaster:~# kubectl create -f clusterip-service.yaml
service/clusteripservice created

root@kubemaster:~# kubectl get services
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
clusteripservice   ClusterIP   10.109.80.217   <none>        80/TCP    5s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP   12d
```

Consultez les EndPoints du service en utilisant la commande suivante :

```
root@kubemaster:~# kubectl get endpoints clusteripservice
NAME               ENDPOINTS                                                AGE
clusteripservice   192.168.150.39:80,192.168.150.40:80,192.168.239.38:80   114s
```

Créez maintenant un pod qui utilisera le service **clusteripservice** :

```
root@kubemaster:~# vi clusterippod.yaml
root@kubemaster:~# cat clusterippod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: clusterippod
spec:
  containers:
  - name: busybox
    image: radial/busyboxplus:curl
    command: ['sh', '-c', 'while true; do sleep 10; done']
```

Créez le pod en utilisant le fichier **clusterippod.yaml** :

```
root@kubemaster:~# kubectl create -f clusterippod.yaml
```

```
pod/clusterippod created
```

Vérifiez que le pod **clusterippod** est en cours de fonctionnement :

```
root@kubemaster:~# kubectl get pod clusterippod
NAME           READY   STATUS    RESTARTS   AGE
clusterippod   1/1     Running   0          2m28s
```

Consultez le service **clusteripservice** de l''intérieur du pod **clusterippod** :

```
root@kubemaster:~# kubectl exec clusterippod -- curl clusteripservice
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
100   612  100   612    0     0   6224      0 --:--:-- --:--:-- --:--:--  6652
```

1.5 - Services et le DNS k8s

Avant de poursuivre, nettoyez le cluster :

```
root@kubemaster:~# kubectl delete service myapp-service
service "myapp-service" deleted

root@kubemaster:~# kubectl delete deployment myapp-deployment
deployment.extensions "myapp-deployment" deleted

root@kubemaster:~# kubectl delete daemonset mydaemonset
daemonset.apps "mydaemonset" deleted

root@kubemaster:~# kubectl delete pods busybox-dnstest nginx-dnstest
pod "busybox-dnstest" deleted
pod "nginx-dnstest" deleted
```

Présentation

Chaque service K8s est attribué un FQDN sous la forme :

```
nom-service.nom-namespace.svc.nom-cluster-domain.example
```

Notez que :

- Le **nom-cluster-domain.example** par défaut est **cluster.local**.
- Le FQDN peut être utilisé pour atteindre un service à partir de n'importe quel NameSpace.
- Les pods du même NameSpace que le service peuvent l'atteindre en utilisant son nom court, à savoir, **nom-service**.

Mise en Application

Visualisez le service **clusteripservice** créé précédement :

```
root@kubemaster:~# kubectl get service clusteripservice
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
clusteripservice   ClusterIP   10.109.80.217   <none>        80/TCP    12m
```

ainsi que les pods présents dans le cluster :

```
root@kubemaster:~# kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
clusterippod                          1/1     Running   0          11m
deploymentclusterip-7776dc8d55-bmfjl  1/1     Running   0          15m
deploymentclusterip-7776dc8d55-pgmcg  1/1     Running   0          15m
deploymentclusterip-7776dc8d55-qvphh  1/1     Running   0          15m
```

Visualisez le FQDN du service **clusteripservice** en utilisant le pod **clusterippod** :

```
root@kubemaster:~# kubectl exec clusterippod -- nslookup 10.109.80.217
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.109.80.217
Address 1: 10.109.80.217 clusteripservice.default.svc.cluster.local
```

Important : Notez que le FQDN du service est **clusteripservice.default.svc.cluster.local**.

Vérifiez la communication avec le service en utilisant son adresse IP :

```
root@kubemaster:~# kubectl exec clusterippod -- curl 10.109.80.217
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

```
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0  35322      0 --:--:--<!DOCTYPE html>:--     0
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
 --:--:-- --:--:-- 36000
```

Vérifiez la communication avec le service en utilisant son nom court :

```
root@kubemaster:~# kubectl exec clusterippod -- curl clusteripservice
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0  81404      0 --:--:-- --:--:-- --:--:--  597k
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Important : Notez que la communication a réussi parce que le pod **clusterippod** et le service **clusteripservice** sont dans le même namespace.

Vérifiez la communication avec le service en utilisant son FQDN :

```
root@kubemaster:~# kubectl exec clusterippod -- curl clusteripservice.default.svc.cluster.local
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0   269k      0 --:--:-- --:--:-- --:--:--  597k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Vérifiez maintenant la communication avec le service en utilisant son nom court à partir du pod **npbusybox** dans le namespace **nptest** :

```
root@kubemaster:~# kubectl exec -n nptest npbusybox -- curl clusteripservice
curl: (6) Couldn't resolve host 'clusteripservice'
```

```
command terminated with exit code 6
```

Important : Notez que la communication n'a pas réussi parce que le pod **npbusybox** et le service **clusteripservice** ne sont pas dans le même namespace.

Vérifiez maintenant la communication avec le service en utilisant son FQDN à partir du pod **npbusybox** dans le namespace **nptest** :

```
root@kubemaster:~# kubectl exec -n nptest npbusybox -- curl clusteripservice.default.svc.cluster.local
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0   291k      0 --:--:-- --:--:-- --:--:--  597k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
```

```
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Important : Notez que la communication a réussi grâce à l'utilisation du FQDN du service.

1.6 - Gestion de K8s Ingress

Présentation

Un Ingress est un objet k8s qui gère l'accès aux services de l'extérieur du cluster. Un Ingress est capable d'avantage de fonctionnalités qu'un simple service NodePort, par exemple :

- SSL,
- équilibrage de charge,
- hôtes virtuels par nom.

L'Ingress ne fait rien tout seul. Il a besoin d'un **Contrôleur Ingress** pour fonctionner. La mise en place et la configuration d'un Contrôleur Ingress ne fait pas parti de la certification CKA.

Mise en Application

Commencez par créer le fichier **myingress.yaml** :

```
root@kubemaster:~# vi myingress.yaml
root@kubemaster:~# cat myingress.yaml
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - http:
      paths:
      - path: /somepath
        pathType: Prefix
        backend:
          service:
            name: clusteripservice
            port:
              number: 80
```

Important : Notez que dans ce fichier Ingress nous avons une règle qui définie un **path**. Des requêtes qui référence le path, par exemple http://<endpoint>/somepath, seront routées vers le **backend**. Dans cet exemple, le backend est un service, **clusteripservice** qui écoute sur le port **80**.

Créez maintenant l'Ingress :

```
root@kubemaster:~# kubectl create -f myingress.yaml
ingress.networking.k8s.io/my-ingress created
```

Consultez maintenant l'Ingress :

```
root@kubemaster:~# kubectl describe ingress my-ingress
Name:             my-ingress
Labels:           <none>
```

```
Namespace:        default
Address:
Ingress Class:    <none>
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /somepath   clusteripservice:80 (192.168.150.39:80,192.168.150.40:80,192.168.239.38:80)
Annotations:  <none>
Events:       <none>
```

Important : Notez que les endpoints du service **clusteripservice** sont affichés dans la sortie de la commande.

Editez maintenant le fichier **clusterip-service.yaml** et ajoutez une ligne **name** dans la section **ports** :

```
root@kubemaster:~# vi clusterip-service.yaml
root@kubemaster:~# cat clusterip-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: clusteripservice
spec:
  type: ClusterIP
  selector:
    app: clusteripexample
  ports:
    - name: myingress
      protocol: TCP
      port: 80
```

```
      targetPort: 80
```

Important : Notez que le nom peut être n'importe quelle chaîne de caractères.

Appliquez la modification du clusteripservice :

```
root@kubemaster:~# kubectl apply -f clusterip-service.yaml
Warning: resource services/clusteripservice is missing the kubectl.kubernetes.io/last-applied-configuration
annotation which is required by kubectl apply. kubectl apply should only be used on resources created
declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched
automatically.
service/clusteripservice configured
```

Important : Notez que l'erreur est sans importance.

Editez maintenant le fichier **myingress.yaml** et ajoutez une ligne **name** dans la section **ports** et en supprimant la ligne **number: 80** :

```
root@kubemaster:~# cat myingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - http:
      paths:
      - path: /somepath
        pathType: Prefix
```

```
        backend:
          service:
            name: clusteripservice
            port:
              name: myingress
```

Appliquez la modification de l'Ingress :

```
root@kubemaster:~# kubectl apply -f myingress.yaml
Warning: resource ingresses/my-ingress is missing the kubectl.kubernetes.io/last-applied-configuration annotation
which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by
either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
ingress.networking.k8s.io/my-ingress configured
```

Important : Notez que l'erreur est sans importance.

Consultez maintenant l'Ingress :

```
root@kubemaster:~# kubectl describe ingress my-ingress
Name:             my-ingress
Labels:           <none>
Namespace:        default
Address:
Ingress Class:    <none>
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /somepath   clusteripservice:myingress (192.168.150.39:80,192.168.150.40:80,192.168.239.38:80)
Annotations:  <none>
```

```
Events:         <none>
```

Important : Notez que l'Ingress peut toujours trouver le backend grâce à l'utilisation du nom **myingress**.

LAB #2 - Gestion d'une Architecture de Microservices

Avant de continer, nettoyez le cluster :

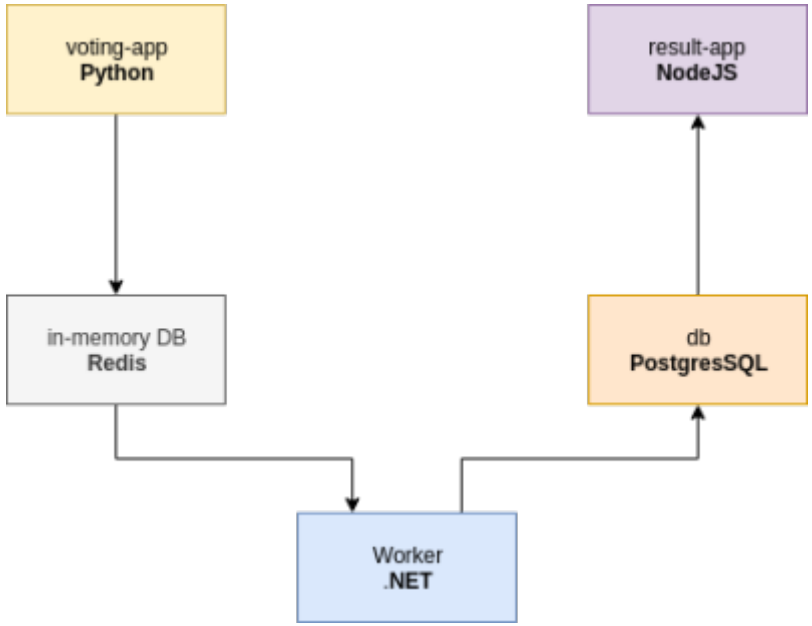
```
root@kubemaster:~# kubectl delete service clusteripservice
service "clusteripservice" deleted

root@kubemaster:~# kubectl delete deployment deploymentclusterip
deployment.apps "deploymentclusterip" deleted

root@kubemaster:~# kubectl delete ingress my-ingress
ingress.networking.k8s.io "my-ingress" deleted

root@kubemaster:~# kubectl delete pod clusterippod
pod "clusterippod" deleted
```

Vérifiez qu'il ne reste que le service par défaut **kubernetes** :

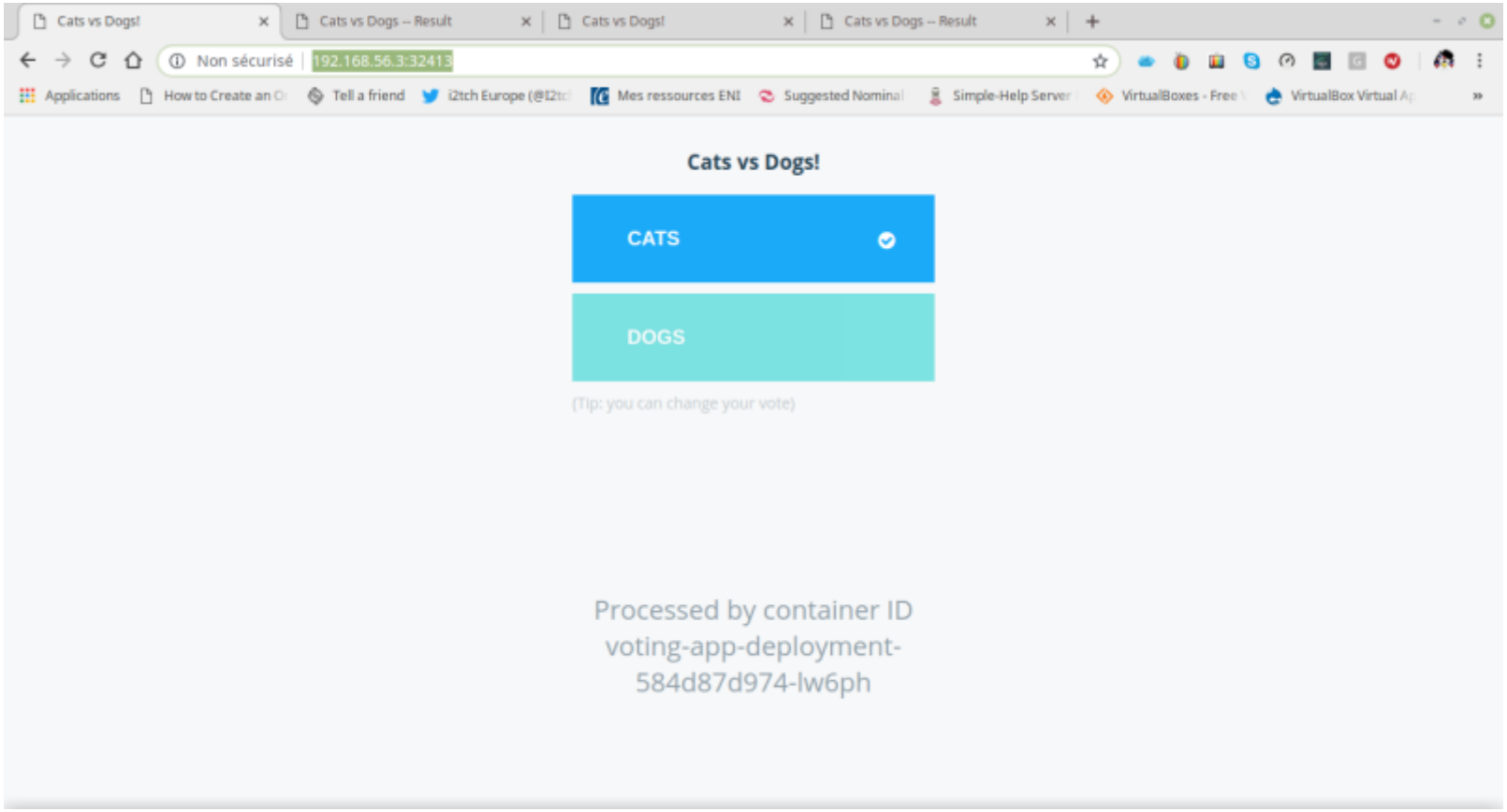
```
root@kubemaster:~# kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   13d
```

2.1 - Présentation

Vous allez mettre en place une application simple, appelé **demo-voting-app** et développé par Docker, sous forme de microservices :

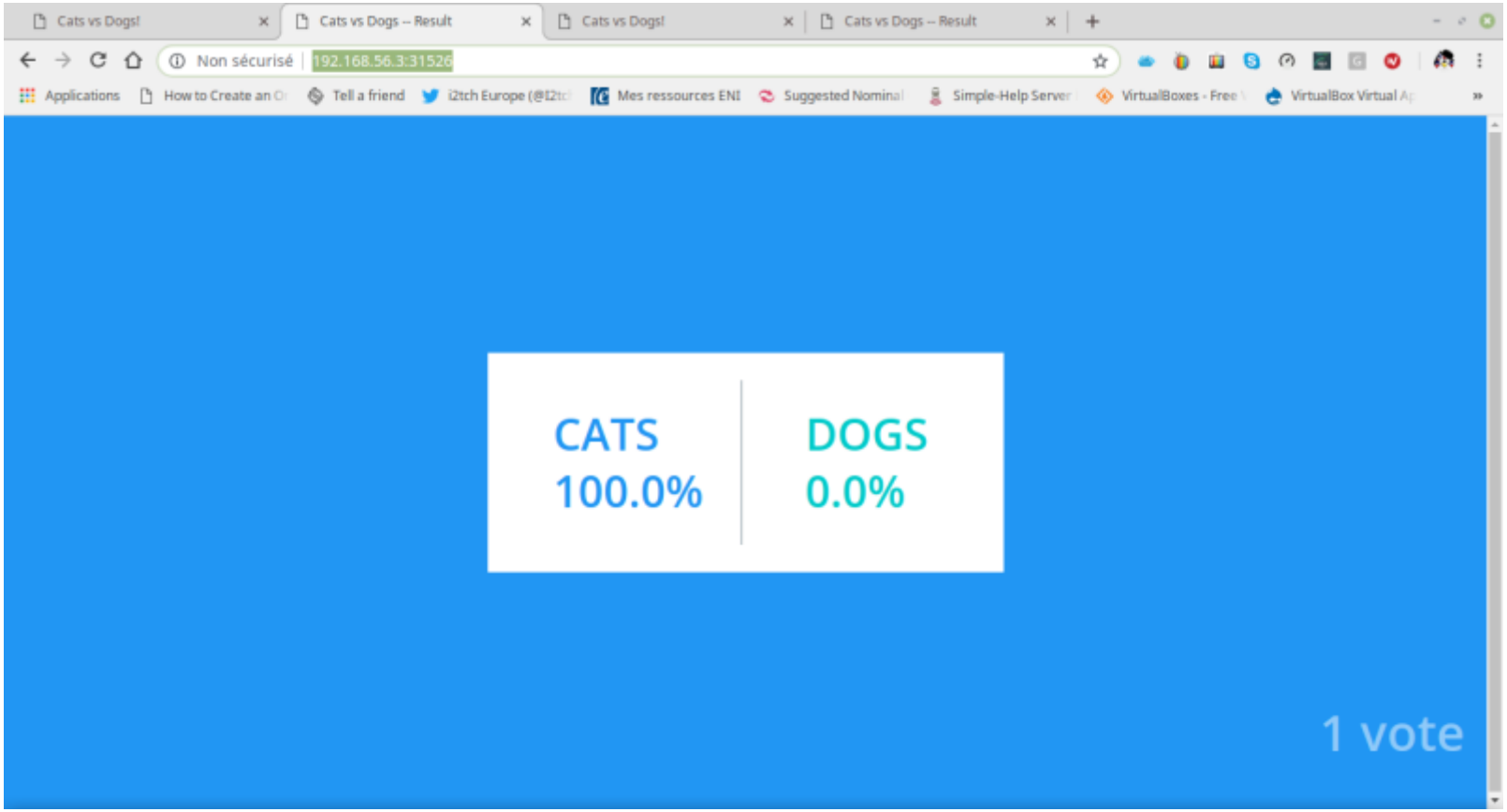


Dans cette application le conteneur **voting-app** permet de voter pour des **chats** ou des **chiens**. Cette application tourne sous Python et fournit une interace HTML :



Lors de la vote, le résultat de celle-ci est stocké dans **Redis** dans une base de données en mémoire. Le résultat est ensuite passé au conteneur **Worker** qui tourne sous .NET et qui met à jour la base de données persistante dans le conteneur **db** qui tourne sous PostgreSQL.

L'application **result-app** qui tourne sous NodeJS lit ensuite la table dans la base de données PostgreSQL et affiche le résultat sous forme HTML :



2.2 - Création des Deployments

Créez le répertoire **myapp**. Placez-vous dans ce répertoire et créez le fichier **voting-app-deployment.yaml** :

```
root@kubemaster:~# mkdir myapp
root@kubemaster:~# cd myapp
root@kubemaster:~/app# vi voting-app-deployment.yaml
root@kubemaster:~/app# cat voting-app-deployment.yaml
---
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: voting-app-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 1
  selector:
    matchLabels:
      name: voting-app-pod
      app: demo-voting-app
  template:
    metadata:
      name: voting-app-pod
      labels:
        name: voting-app-pod
        app: demo-voting-app

    spec:
      containers:
      - name: voting-app
        image: dockersamples/examplevotingapp_vote
        ports:
        - containerPort: 80
```

Important : Ce fichier décrit un Deployment. Notez que le Deployment crée **un** replica du POD spécifié par **template** contenant un conteneur dénommé **voting-app** qui utilise le port 80 et qui est créé à partir de l'image **dockersamples/examplevotingapp_vote**.

Créez maintenant le fichier **redis-deployment.yaml** :

```
root@kubemaster:~/app# vi redis-deployment.yaml
root@kubemaster:~/app# cat redis-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 1
  selector:
    matchLabels:
      name: redis-pod
      app: demo-voting-app
  template:
    metadata:
      name: redis pod
      labels:
        name: redis-pod
        app: demo-voting-app

    spec:
      containers:
      - name: redis
        image: redis
        ports:
        - containerPort: 6379
```

Important : Ce fichier décrit un Deployment. Notez que le Deployment crée **un** replica du POD spécifié par **template** contenant un conteneur dénommé **redis** qui utilise le port 6379 et qui est créé à partir de l'image **redis**.

Créez le fichier **worker-deployment.yaml** :

```
root@kubemaster:~/app# vi worker-deployment.yaml
root@kubemaster:~/app# cat worker-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: worker-app-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 1
  selector:
    matchLabels:
      name: worker-app-pod
      app: demo-voting-app
  template:
    metadata:
      name: worker-app-pod
      labels:
        name: worker-app-pod
        app: demo-voting-app

    spec:
      containers:
      - name: worker-app
        image: dockersamples/examplevotingapp_worker
```

Important : Ce fichier décrit un Deployment. Notez que le Deployment crée **un** replica du POD spécifié par **template** contenant un conteneur dénommé **worker-app** qui est créé à partir de l'image **dockersamples/examplevotingapp_worker**.

Créez ensuite le fichier **postgres-deployment.yaml** :

```
root@kubemaster:~/app# vi postgres-deployment.yaml
root@kubemaster:~/app# cat postgres-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 1
  selector:
    matchLabels:
      name: postgres-pod
      app: demo-voting-app
  template:
    metadata:
      name: postgres pod
      labels:
        name: postgres-pod
        app: demo-voting-app

    spec:
      containers:
      - name: postgres
        image: postgres:9.4
        env:
        - name: POSTGRES_USER
          value: postgres
        - name: POSTGRES_PASSWORD
          value: postgres
        ports:
```

```
        - containerPort: 5432
```

Important : Ce fichier décrit un Deployment. Notez que le Deployment crée **un** replica du POD spécifié par **template** contenant un conteneur dénommé **postgres** qui utilise le port 5432 et qui est créé à partir de l'image **postgres:9.4**.

Dernièrement, créez le fichier **result-app-deployment.yaml** :

```
root@kubemaster:~/app# vi result-app-deployment.yaml
root@kubemaster:~/app# cat result-app-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: result-app-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 1
  selector:
    matchLabels:
      name: result-app-pod
      app: demo-voting-app
  template:
    metadata:
      name: result-app-pod
      labels:
        name: result-app-pod
        app: demo-voting-app

    spec:
```

```
      containers:
      - name: result-app
        image: dockersamples/examplevotingapp_result
        ports:
        - containerPort: 80
```

Important : Ce fichier décrit un Deployment. Notez que le Deployment crée **un** replica du POD spécifié par **template** contenant un conteneur dénommé **result-app** qui utilise le port 80 et qui est créé à partir de l'image **dockersamples/examplevotingapp_result**.

2.3 - Création des Services

Créez maintenant le fichier **redis-service.yaml** :

```
root@kubemaster:~/app# vi redis-service.yaml
root@kubemaster:~/app# cat redis-service.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: redis
  labels:
    name: redis-service
    app: demo-voting-app

spec:
  ports:
  - port: 6379
    targetPort: 6379
  selector:
```

```
    name: redis-pod
    app: demo-voting-app
```

Important : Ce fichier décrit un Service **ClusterIP**. Notez que le Service expose le port **6379** sur tout POD ayant le nom **redis-pod**.

Créez ensuite le fichier **postgres-service.yaml** :

```
root@kubemaster:~/app# vi postgres-service.yaml
root@kubemaster:~/app# cat postgres-service.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: db
  labels:
    name: db-service
    app: demo-voting-app

spec:
  ports:
  - port: 5432
    targetPort: 5432
  selector:
    name: postgres-pod
    app: demo-voting-app
```

Important : Ce fichier décrit un Service **ClusterIP**. Notez que le Service expose le port **5432** sur tout POD ayant le nom **postgres-pod**.

Créez le fichier **voting-app-service.yaml** :

```
root@kubemaster:~/app# vi voting-app-service.yaml
root@kubemaster:~/app# cat voting-app-service.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: voting-service
  labels:
    name: voting-service
    app: demo-voting-app

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
  selector:
    name: voting-app-pod
    app: demo-voting-app
```

Important : Ce fichier décrit un Service **NodePort**. Notez que le Service expose le port **80** sur tout POD ayant le nom **voting-app-pod**.

Dernièrement, créez le fichier **result-app-service.yaml** :

```
root@kubemaster:~/app# vi result-app-service.yaml
root@kubemaster:~/app# cat result-app-service.yaml
---
apiVersion: v1
```

```
kind: Service
metadata:
  name: result-service
  labels:
    name: result-service
    app: demo-voting-app

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
  selector:
    name: result-app-pod
    app: demo-voting-app
```

Important : Ce fichier décrit un Service **NodePort**. Notez que le Service expose le port **80** sur tout POD ayant le nom **result-app-pod**.

2.4 - Déployer l'Application

Vérifiez que vous avez créé tous les fichiers YAML necéssaires :

```
root@kubemaster:~/myapp# ls
postgres-deployment.yaml  redis-deployment.yaml  result-app-deployment.yaml  voting-app-deployment.yaml  worker-
deployment.yaml
postgres-service.yaml     redis-service.yaml     result-app-service.yaml     voting-app-service.yaml
```

Utilisez ensuite la commande **kubectl create** :

```
root@kubemaster:~/myapp# kubectl create -f .
deployment.apps/postgres-deployment created
service/db created
deployment.apps/redis-deployment created
service/redis created
deployment.apps/result-app-deployment created
service/result-service created
deployment.apps/voting-app-deployment created
service/voting-service created
deployment.apps/worker-app-deployment created
```

Important : Notez l'utilisation du caractère **.** qui indique tout fichier dans le répertoire courant.

Attendez que tous les Deployments soient **READY** (7 à 10 minutes) :

```
root@kubemaster:~/myapp# kubectl get deployments
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
postgres-deployment    1/1     1            1           51m
redis-deployment       1/1     1            1           51m
result-app-deployment  1/1     1            1           51m
voting-app-deployment  1/1     1            1           51m
worker-app-deployment  1/1     1            1           51m
```

Contrôlez ensuite l'état des PODs :

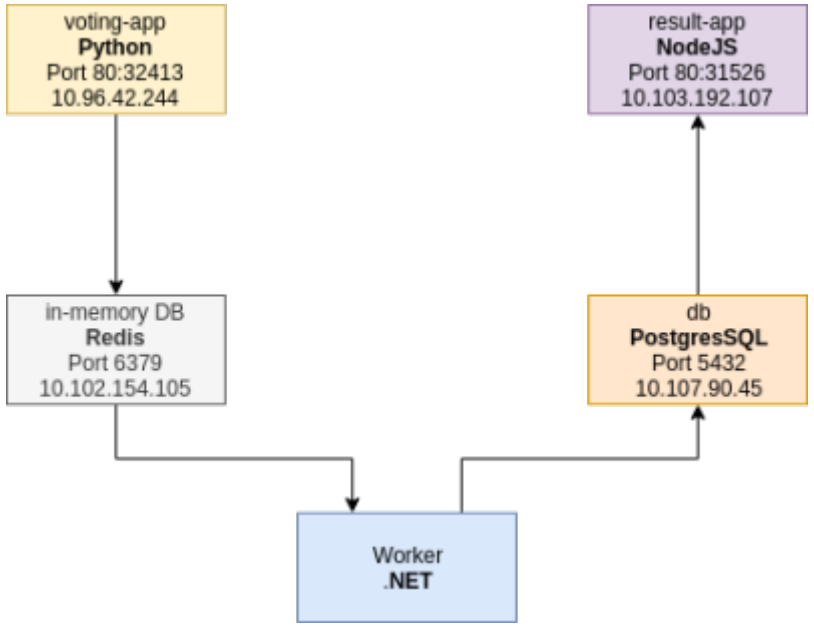
```
root@kubemaster:~/myapp# kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
postgres-deployment-5b8bd66778-j99zz  1/1     Running   0          51m
redis-deployment-67d4c466c4-9wzfn     1/1     Running   0          51m
result-app-deployment-b8f9dc967-nzbgd 1/1     Running   0          51m
```

```
voting-app-deployment-669dccccfb-jpn6h   1/1     Running   0          51m
worker-app-deployment-559f7749b6-jh86r   1/1     Running   0          51m
```

ainsi que la liste des Services :

```
root@kubemaster:~/myapp# kubectl get services
NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
db               ClusterIP   10.107.90.45     <none>        5432/TCP       24h
kubernetes       ClusterIP   10.96.0.1        <none>        443/TCP        4d9h
redis            ClusterIP   10.102.154.105   <none>        6379/TCP       24h
result-service   NodePort    10.103.192.107   <none>        80:31526/TCP   24h
voting-service   NodePort    10.96.42.244     <none>        80:32413/TCP   24h
```

Dans le cas donc de l'exemple dans ce cours, l'application ressemble maintenant au diagramme suivant :



2.5 - Scaling Up

Éditez le fichier **voting-app-deployment.yaml** et modifiez la valeur du champ **replicas** de 1 à 3 :

```
root@kubemaster:~/app# vi voting-app-deployment.yaml
root@kubemaster:~/app# cat voting-app-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: voting-app-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 3
  selector:
    matchLabels:
      name: voting-app-pod
      app: demo-voting-app
  template:
    metadata:
      name: voting-app-pod
      labels:
        name: voting-app-pod
        app: demo-voting-app

    spec:
      containers:
      - name: voting-app
        image: dockersamples/examplevotingapp_vote
        ports:
        - containerPort: 80
```

Éditez le fichier **result-app-deployment.yaml** et modifiez la valeur du champ **replicas** de 1 à 3 :

```
root@kubemaster:~/app# vi result-app-deployment.yaml
root@kubemaster:~/app# cat result-app-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: result-app-deployment
  labels:
    app: demo-voting-app
spec:
  replicas: 3
  selector:
    matchLabels:
      name: result-app-pod
      app: demo-voting-app
  template:
    metadata:
      name: result-app-pod
      labels:
        name: result-app-pod
        app: demo-voting-app

    spec:
      containers:
      - name: result-app
        image: dockersamples/examplevotingapp_result
        ports:
        - containerPort: 80
```

Appliquez les modifications à l'aide de la commande **kubectl apply** :

```
root@kubemaster:~/myapp# kubectl apply -f voting-app-deployment.yaml
```

```
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
deployment.apps/voting-app-deployment configured

root@kubemaster:~/myapp# kubectl apply -f result-app-deployment.yaml
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
deployment.apps/result-app-deployment configured
```

Contrôlez ensuite les Deployments :

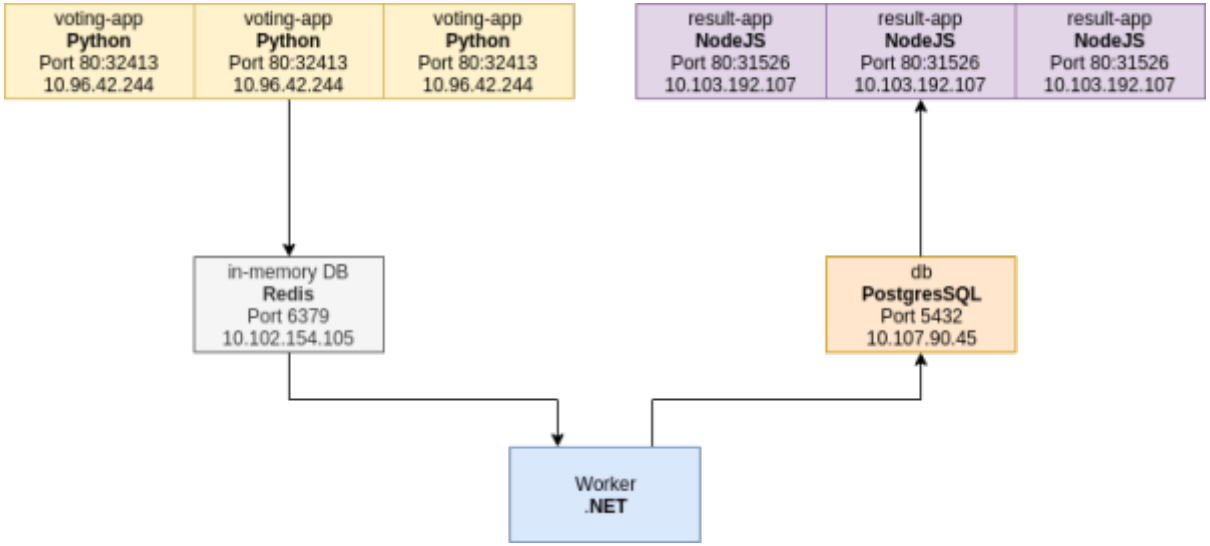
```
root@kubemaster:~/myapp# kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
postgres-deployment     1/1     1            1           23h
redis-deployment        1/1     1            1           23h
result-app-deployment   3/3     3            3           23h
voting-app-deployment   3/3     3            3           23h
worker-app-deployment   1/1     1            1           23h
```

ainsi que les PODs :

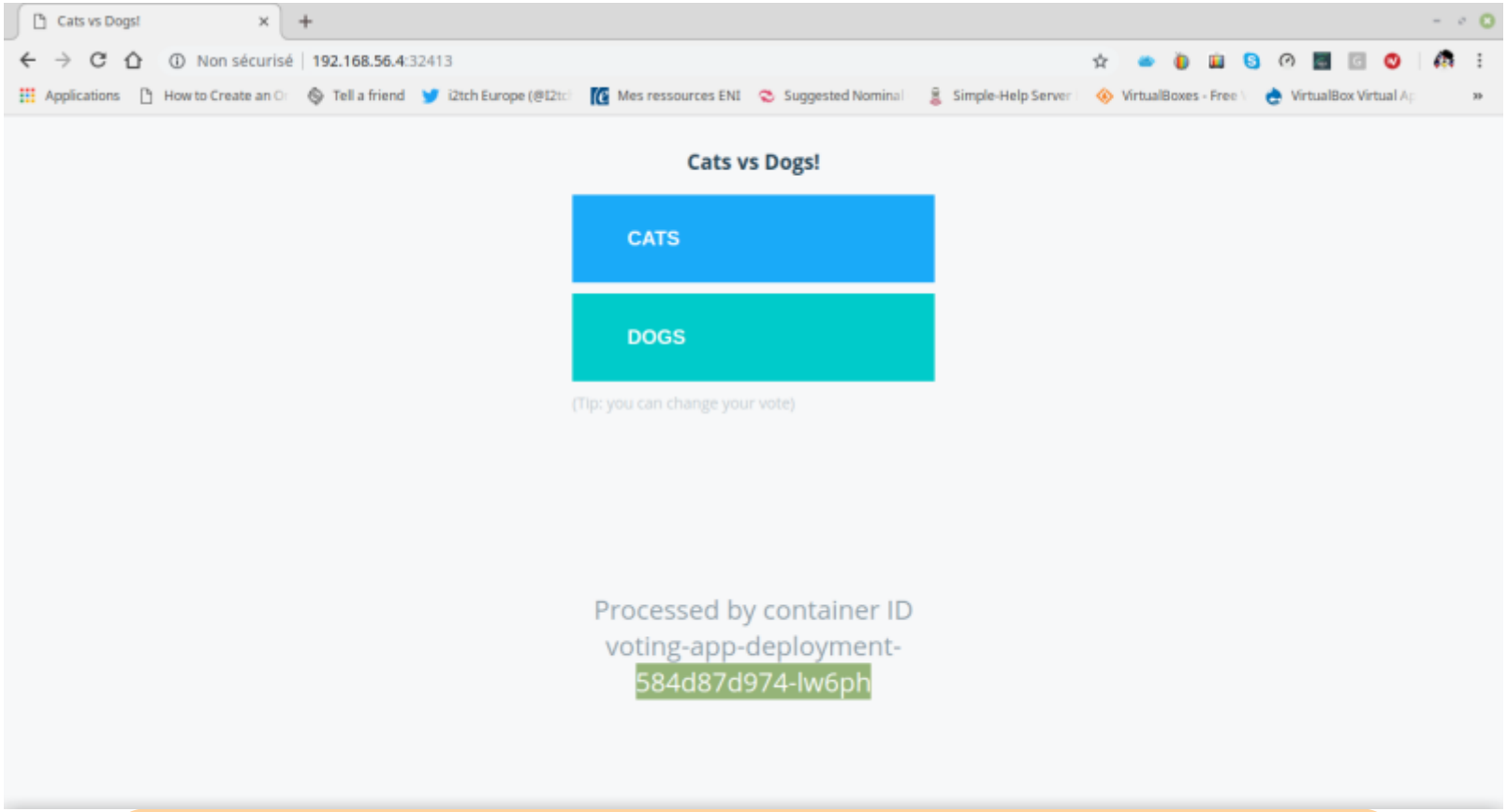
```
root@kubemaster:~/myapp# kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE     IP                NODE
NOMINATED NODE   READINESS GATES
postgres-deployment-5b8bd66778-j99zz     1/1     Running   1          169m    192.168.35.83     kubenode2
<none>           <none>
redis-deployment-67d4c466c4-9wzfn        1/1     Running   1          169m    192.168.205.217   kubenode1
<none>           <none>
result-app-deployment-b8f9dc967-nzbgd    1/1     Running   1          169m    192.168.205.218   kubenode1
<none>           <none>
result-app-deployment-b8f9dc967-r84k6    1/1     Running   0          2m36s   192.168.35.86     kubenode2
<none>           <none>
result-app-deployment-b8f9dc967-zbsk2    1/1     Running   0          2m36s   192.168.35.85     kubenode2
<none>           <none>
voting-app-deployment-669dccccfb-jpn6h   1/1     Running   1          169m    192.168.35.82     kubenode2
<none>           <none>
```

```
voting-app-deployment-669dccccfb-ktd7d   1/1     Running   0          2m50s   192.168.35.84     kubenode2
<none>           <none>
voting-app-deployment-669dccccfb-x868p   1/1     Running   0          2m50s   192.168.205.219   kubenode1
<none>           <none>
worker-app-deployment-559f7749b6-jh86r   1/1     Running   2          169m    192.168.205.216   kubenode1
<none>           <none>
```

Dans le cas de l'exemple dans ce cours, l'application ressemble maintenant au diagramme suivant :

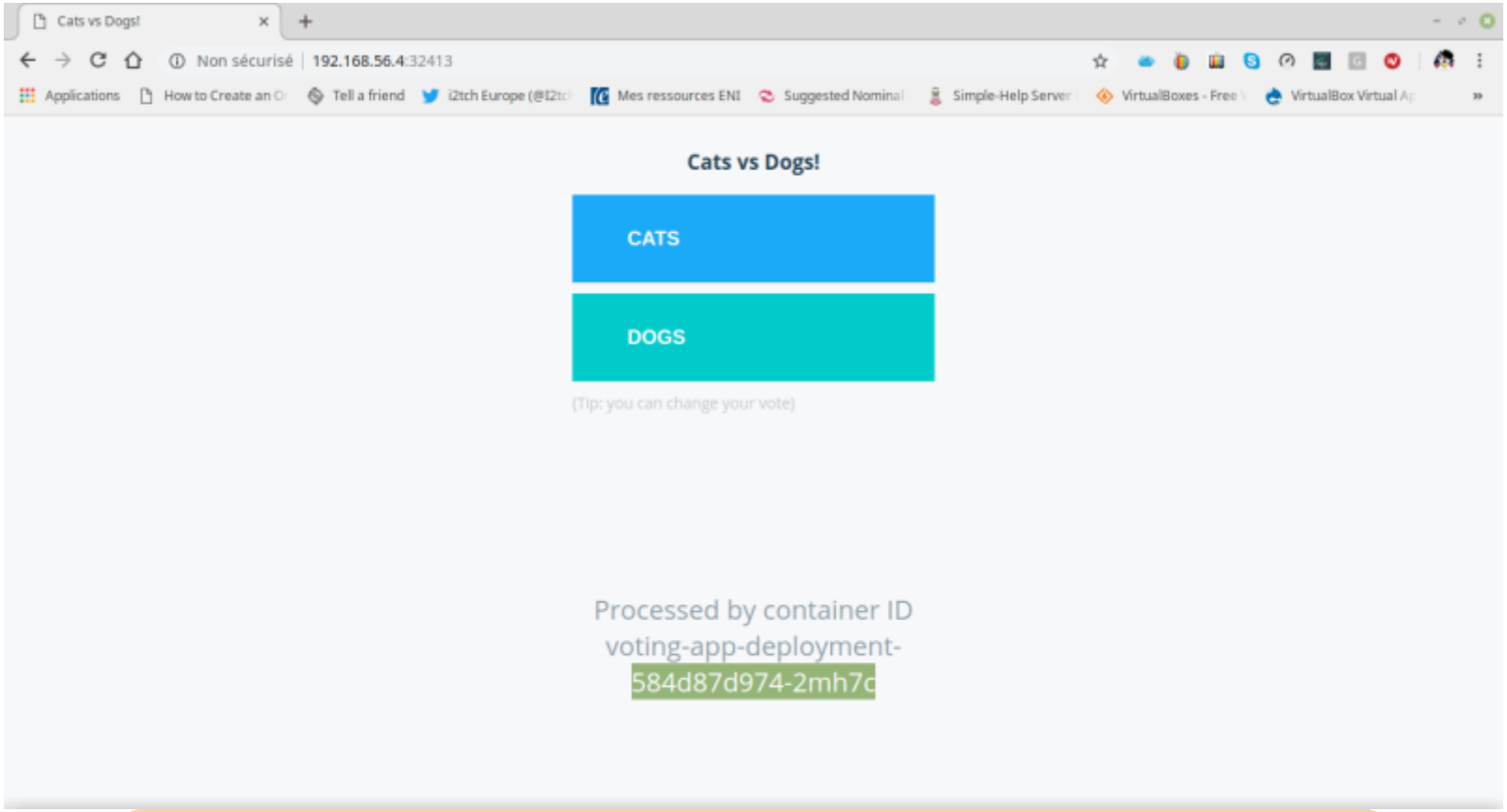


Retournez sur le navigateur de votre machine hôte et rafraichissez la page du voting-app :



Important : Notez le POD qui a servi la page.

Rafraîchissez la page de nouveau :



Important : Notez que le POD qui a servi la page a changé.

Notez que ce changement de POD n'indique pas un équilibrage de charge. En effet, il faudrait mettre en place une autre machine virtuelle sous, par exemple, HAProxy pour obtenir l'équilibrage.

Par contre, dans le cas d'une application sur GCP par exemple, il convient de modifier les deux fichiers suivants en changeant la valeur de champ **type** de NodePort à **LoadBalancer** puis de configurer une instance du Load Balancer natif de GCP :

```
root@kubemaster:~/app# vi voting-app-service.yaml
```

```
root@kubemaster:~/app# cat voting-app-service.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: voting-service
  labels:
    name: voting-service
    app: demo-voting-app

spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 80
  selector:
    name: voting-app-pod
    app: demo-voting-app
```

Important : Ce fichier décrit un Service **LoadBalancer**. Notez que le Service expose le port **80** sur tout POD ayant le nom **voting-app-pod**.

Dernièrement, créez le fichier **result-app-service.yaml** :

```
root@kubemaster:~/app# vi result-app-service.yaml
root@kubemaster:~/app# cat result-app-service.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: result-service
```

```
  labels:
    name: result-service
    app: demo-voting-app

spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 80
  selector:
    name: result-app-pod
    app: demo-voting-app
```
