Experiment: Nginx Ingress on Kind

In this experiment, we will deploy Nginx and access on Kind.

The manifests contains kind specific patches to forward the hostPorts to the ingress controller, set taint tolerations and schedule it to the custom labelled node.

Create a cluster, for the experiment

```
cat <<EOF | kind create cluster  --image kindest/node:v1.21.12  --config=-
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
EOF
```

We create a namespace to isolate Nginx from nginx-01-namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/instance: ingress-nginx
---
```

Create the service account from nginx-02-controller-serviceaccount.yaml
Create a configmap from nginx-03-controller-configmap.yaml
Create the ClusterRole from nginx-04-clusterrole-status.yaml
Bind the role nginx-05-clusterrole-status-binding.yaml
Create the configmap role nginx-06-role-configmaps.yaml
Bind the configmap nginx-07-role-configmaps-binding.yaml
Create the Ingress Controller webhook nginx-08-controller-server-webhook.yaml
Create the service nginx-09-controller-service.yaml
Deploy the Ingress Controller deployment from nginx-10-controller-deployment.yaml

Deploy the Admission webhook from nginx-11-controller-admission-webhook.yaml
Create the service account from nginx-12-controller-admission-serviceaccount.yaml
Create the Admission ClusterRole from nginx-13-controller-admission-clusterrole.yaml
Bind the role for Admission nginx-14-controller-admission-clusterrole-binding.yaml
Create the admission role nginx-15-controller-admission-role.yaml
Bind the admission role nginx-16-controller-admission-role-binding.yaml
Create the secret implementation from nginx-17-controller-secret.yaml
Patch including the certification generator from nginx-18-controller-patch.yaml

Note: In reviewing you'll see that this implementation does best practice label application to ensure that the components can be identified across the cluster for observability

We could individually apply these files but for simplicity we'll apply all the nginx files in one sweep.

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml

```
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-
admission created
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
 job.batch/ingress-nginx-admission-patch created
```

kubernetes@DESKTOP-1M2VN7E MINGW64 /c/projects/KubernetesNetworking
$ kubectl wait --namespace ingress-nginx --for=condition=ready pod --selector=app.kubernetes.io/component=controller --timeout=90s

pod/ingress-nginx-controller-78f889f8b9-dlltl condition met

Now the Ingress should be all setup. Wait until the Ingress Controller is ready to process requests running.

Using Ingress

The following example creates simple http-echo services and an Ingress object to route to these services.

```
kind: Pod
apiVersion: v1
```

```
metadata:
  name: foo-app
  labels:
    app: foo
spec:
  containers:
  - name: foo-app
    image: hashicorp/http-echo:0.2.3
    args:
    - "-text=foo"
---
kind: Service
apiVersion: v1
metadata:
  name: foo-service
spec:
  selector:
    app: foo
  ports:
  # Default port used by the image
  - port: 5678
---
kind: Pod
apiVersion: v1
metadata:
  name: bar-app
  labels:
    app: bar
spec:
  containers:
  - name: bar-app
    image: hashicorp/http-echo:0.2.3
    args:
    - "-text=bar"
---
kind: Service
apiVersion: v1
metadata:
  name: bar-service
spec:
  selector:
    app: bar
  ports:
  # Default port used by the image
  - port: 5678
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - http:
      paths:
      - pathType: Prefix
        path: "/foo"
        backend:
          service:
            name: foo-service
            port:
              number: 5678
      - pathType: Prefix
        path: "/bar"
        backend:
          service:
            name: bar-service
            port:
              number: 5678
---
```

Apply the Pod/Container Deployment

$ kubectl apply -f https://kind.sigs.k8s.io/examples/ingress/usage.yaml

$ kubectl get pods

```
NAME      READY   STATUS    RESTARTS   AGE
bar-app   1/1     Running   0          37s
foo-app   1/1     Running   0          37s
```

```
kubernetes@DESKTOP-1M2VN7E MINGW64
/c/projects/KubernetesNetworking
```

$ kubectl get pods --namespace ingress-nginx

```
NAME                                        READY   STATUS
RESTARTS   AGE
ingress-nginx-admission-create-8vfwf        0/1     Completed   0
13m
ingress-nginx-admission-patch-4nmc9         0/1     Completed   3
13m
ingress-nginx-controller-78f889f8b9-dlltl   1/1     Running     0
13m
```

Test our services with the Nginx Ingress Controller

Note: For MacOS you may need to substitute 127.0.0.1 for localhost

```
kubernetes@DESKTOP-1M2VN7E MINGW64
/c/projects/KubernetesNetworking
```

$ curl -s localhost/foo

```
foo
```

```
kubernetes@DESKTOP-1M2VN7E MINGW64
/c/projects/KubernetesNetworking
```

$ curl -s localhost/bar

```
bar
```

Experiment Cleanup

Remove the Zookeeper and Kafka containers with Docker Compose

$ kubectl delete -f https://kind.sigs.k8s.io/examples/ingress/usage.yaml

```
pod "foo-app" deleted
service "foo-service" deleted
pod "bar-app" deleted
service "bar-service" deleted
```

$ kubectl delete -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml

```
namespace "ingress-nginx" deleted
serviceaccount "ingress-nginx" deleted
configmap "ingress-nginx-controller" deleted
clusterrole.rbac.authorization.k8s.io "ingress-nginx" deleted
```

```
clusterrolebinding.rbac.authorization.k8s.io "ingress-nginx" deleted
role.rbac.authorization.k8s.io "ingress-nginx" deleted
rolebinding.rbac.authorization.k8s.io "ingress-nginx" deleted
service "ingress-nginx-controller-admission" deleted
service "ingress-nginx-controller" deleted
deployment.apps "ingress-nginx-controller" deleted
validatingwebhookconfiguration.admissionregistration.k8s.io "ingress-nginx-
admission" deleted
serviceaccount "ingress-nginx-admission" deleted
clusterrole.rbac.authorization.k8s.io "ingress-nginx-admission" deleted
clusterrolebinding.rbac.authorization.k8s.io "ingress-nginx-admission" deleted
role.rbac.authorization.k8s.io "ingress-nginx-admission" deleted
rolebinding.rbac.authorization.k8s.io "ingress-nginx-admission" deleted
job.batch "ingress-nginx-admission-create" deleted
job.batch "ingress-nginx-admission-patch" deleted
```

$ kind get clusters

```
kind
```

$ kind delete cluster

```
Deleting cluster "kind" ...
```