K8S (9) Ingress

To access our deployed containerized application from the external world via *Services*, we use NodePort and LoadBalancer.
For LB, we may not want to expose every service.
For NodePort, we need too keep updating the proxy setting.
We can add a Ingress API before request reach Service API to manage access
"An Ingress is a collection of rules that allow inbound connections to reach the cluster Services".

Ingress

- Name-Based Virtual Hosting Ingress (Two different rules):
  - User requests to both **blue.example.com** and **green.example.com** would go to the same Ingress endpoint
  - They would be forwarded to **webserver-blue-svc**, and **webserver-green-svc** respectively.

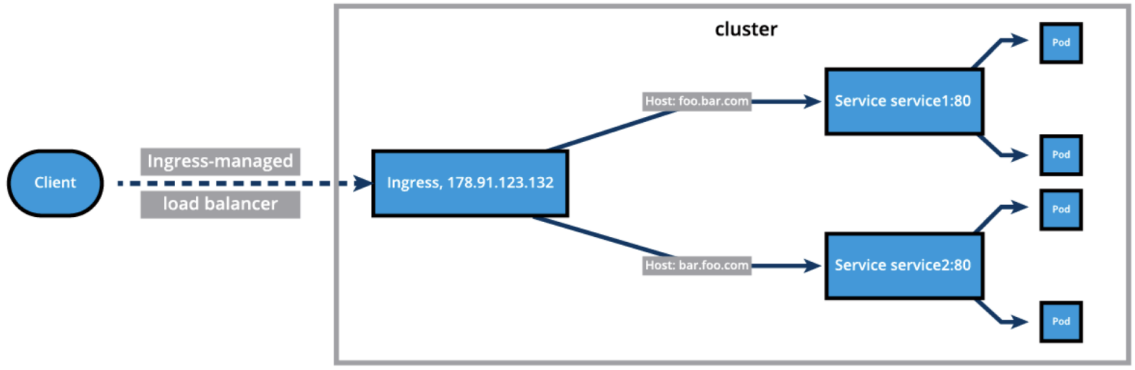
```
# name-based-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: "nginx"
  name: virtual-host-ingress
  namespace: default
spec:
  rules:
  - host: blue.example.com
    http:
      paths:
      - backend:
          service:
            name: webserver-blue-svc
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
  - host: green.example.com
    http:
      paths:
      - backend:
          service:
            name: webserver-green-svc
            port:
              number: 80
        path: /
        pathType: ImplementationSpecific
```

Fanout Ingress (One rule, multiple paths):

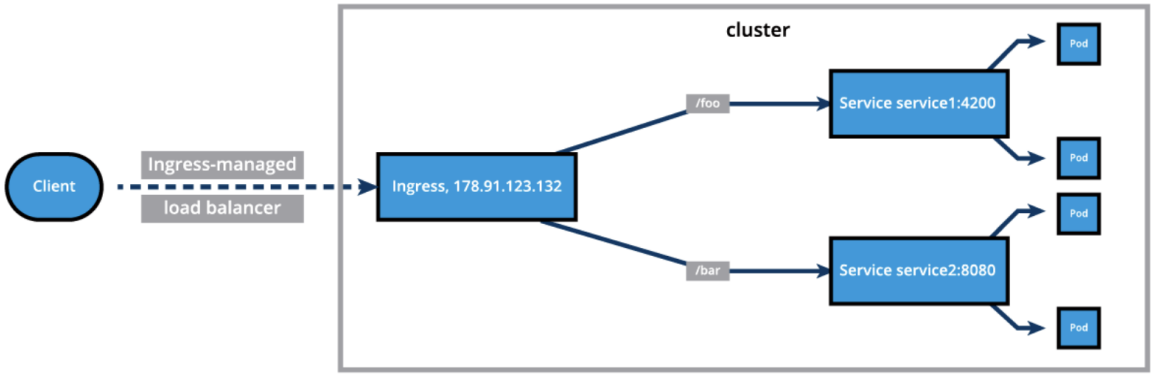
Requests to **example.com/blue** forwards to **webserver-blue-svc**.

Requests to **example.com/green** forwards to **webserver-green-svc**.

```
# fanout-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: "nginx"
  name: fan-out-ingress
  namespace: default
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /blue
        backend:
          service:
            name: webserver-blue-svc
            port:
              number: 80
        pathType: ImplementationSpecific
      - path: /green
        backend:
          service:
            name: webserver-green-svc
            port:
              number: 80
        pathType: ImplementationSpecific
```



Name-Based Virtual Hosting Ingress



Fanout Ingress

Ingress Controller

Ingress Controllers are also know as Controllers, Ingress Proxy, Service Proxy, Revers Proxy, etc...

An application watching the Control Plane Node's API server for changes in the Ingress resources and updates the Layer 7 Load Balancer accordingly.

minikube use [Nginx Ingress Controller] set up as an addon

```
$ minikube addons enable ingress

$ kubectl create -f virtual-host-ingress.yaml
```

```
$ sudo vim /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1 localhost
255.255.255.255 broadcasthost
::1 localhost

192.168.105.6 blue.io green.io # 192.168.105.6 is the minikube ip
```

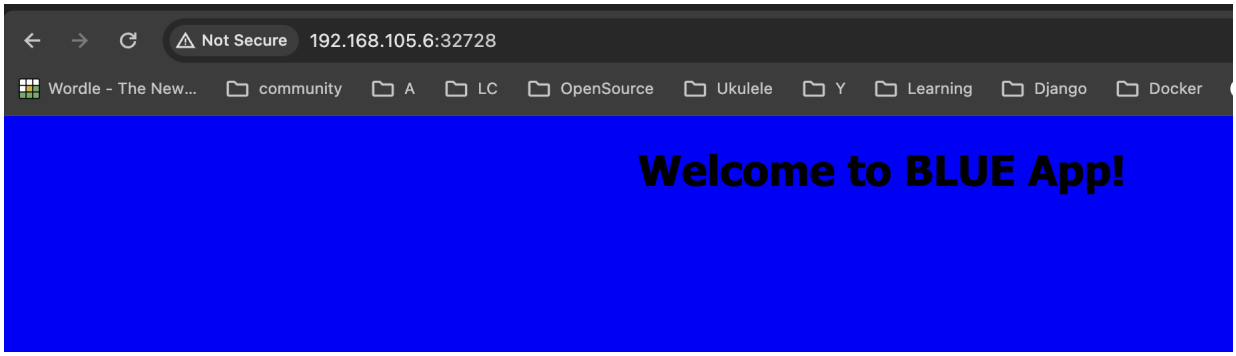
Demo of Ingress Rules:

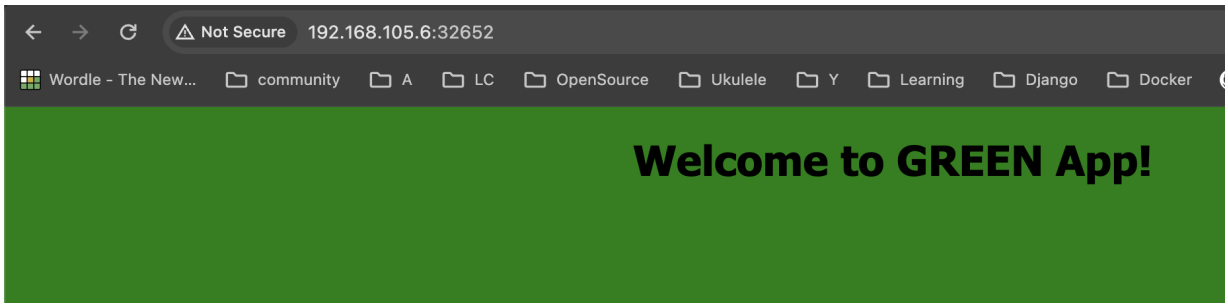
Both service blue-app and green-web are on the famous port 80.

We would need to use the Ingress in conjunction with an Ingress controller to achieve that.

Reuse the blue-web and green web deployment we had (web-blue-with-cm.yaml and web-green-with-cm.yaml)

```
$ kubectl get deploy,svc
NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/blue-web 1/1 1 1 47s
deployment.apps/green-web 1/1 1 1 3m57s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/blue-web NodePort 10.104.84.0 <none> 80:32728/TCP 39s
service/green-web NodePort 10.107.5.157 <none> 80:32652/TCP 3m52s
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 6m24s

$ minikube service list
|-------------|------------|--------------|---------------------------|
| NAMESPACE | NAME | TARGET PORT | URL |
|-------------|------------|--------------|---------------------------|
| default | blue-web | 80 | http://192.168.105.6:32728 |
| default | green-web | 80 | http://192.168.105.6:32652 |
|-------------|------------|--------------|---------------------------|
```





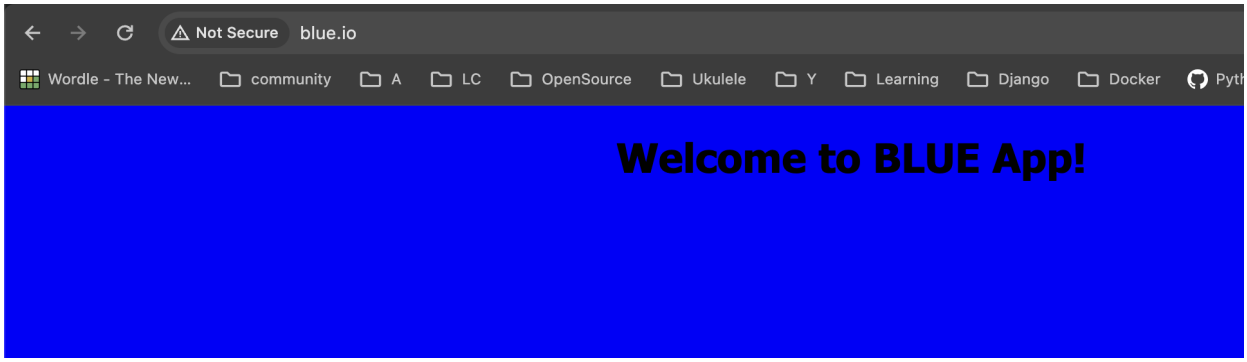

```
$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube
on GitHub.
You can view the list of minikube maintainers at:
https://github.com/kubernetes/minikube/blob/master/OWNERS
▪ Using image registry.k8s.io/ingress-nginx/controller:v1.9.4
▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled

% minikube addons list
...
| helm-tiller | minikube | disabled | 3rd party (Helm) |
| inaccel | minikube | disabled | 3rd party (InAccel |
| | | | [info@inaccel.com]) |
| ingress | minikube | enabled ✅ | Kubernetes |
| ingress-dns | minikube | disabled | minikube |
| inspektor-gadget | minikube | disabled | 3rd party |
| | | | (inspektor-gadget.io) |
...
```

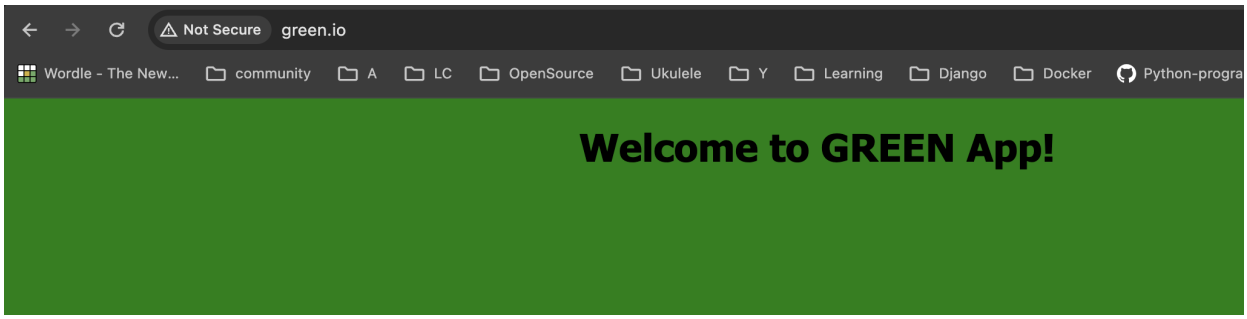
```
# ingress-demo.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-demo
  namespace: default
spec:
  rules:
  - host: blue.io
    http:
      paths:
      - backend:
          service:
            name: blue-web
            port:
              number: 80
        path: "/"
        pathType: ImplementationSpecific
  - host: green.io
    http:
      paths:
      - backend:
          service:
            name: green-web
            port:
              number: 80
        path: "/"
        pathType: ImplementationSpecific
```

```
$ kubectl apply -f ingress-demo.yaml
ingress.networking.k8s.io/ingress-demo created

$ kubectl get ingress
NAME CLASS HOSTS ADDRESS PORTS AGE
```

```
chenyang@ChenYangs-MBP myKubernetes % kubectl describe ingress ingress-demo
Name: ingress-demo
Labels: <none>
Namespace: default
Address:
Ingress Class: nginx
Default backend: <default>
Rules:
Host Path Backends
---- ---- --------
blue.io
/ blue-web:80 (10.244.0.4:80)
green.io
/ green-web:80 (10.244.0.3:80)
Annotations: <none>
Events:
Type Reason Age From Message
---- ------ ---- ---- -------
Normal Sync 23s nginx-ingress-controller Scheduled for sync

$ minikube ip
192.168.105.6

$ sudo vim /etc/hosts
// In order to take full advantage of the Ingress resource from a local browser, we
would need to make sure that our /etc/hosts file includes the IP address of the
Minikube control plane node

$ cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1 localhost
255.255.255.255 broadcasthost
::1 localhost

192.168.105.6 blue.io green.io
```

Bother blue.io and green.io are expose on port 80



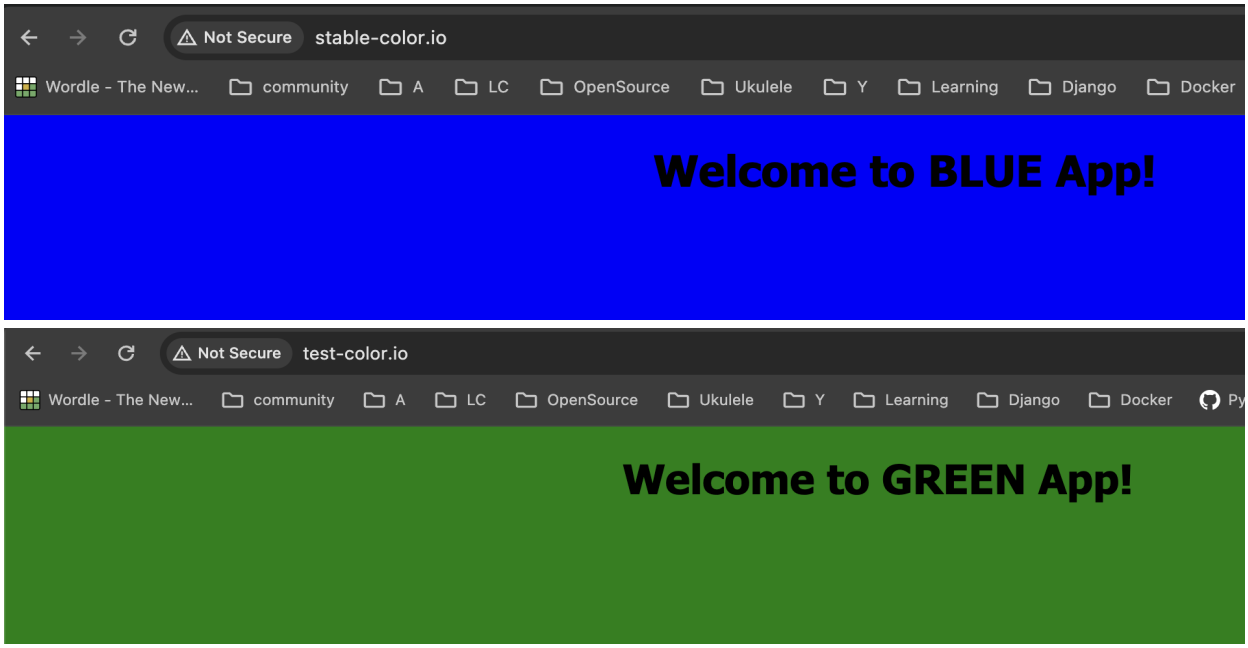


Continue with another scenario (common in enterprise setting)
Blue application is stable, Green application is not production-ready yet.
So we would like to setup a strategy: If the Green is production ready, send request to Green application. Otherwise send to Blue application.

```
# ingress-demo.yaml
# The same ingress, updating the host name, so we use replace ingress in the
following steps
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-demo
  namespace: default
spec:
  rules:
  - host: stable-color.io
    http:
      paths:
      - backend:
          service:
            name: blue-web
            port:
              number: 80
        path: "/"
        pathType: ImplementationSpecific
  - host: test-color.io
    http:
      paths:
      - backend:
          service:
            name: green-web
            port:
              number: 80
        path: "/"
        pathType: ImplementationSpecific
```

```
$ kubectl replace --force -f ingress-demo.yaml
ingress.networking.k8s.io "ingress-demo" deleted
ingress.networking.k8s.io/ingress-demo replaced
```

```
$ sudo vim /etc/hosts
Password:

$ cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1 localhost
255.255.255.255 broadcasthost
::1 localhost

192.168.105.6 blue.io green.io
192.168.105.6 stable-color.io test-color.io
```



Later on, when green-web is ready, update the stable-color.io host to green-web service, and
$ kubectl replace --force ingress-demo.yaml