Ingress in Kubernets
Creatin ingress with nginx in clusterip mode

In Kubernetes, Ingress is an API object that manages external access to services in a cluster, typically HTTP/HTTPS. It provides features like load balancing, SSL termination, and name-based virtual hosting. In Minikube

Manages external access to services (HTTP/HTTPS).
Provides load balancing, SSL termination, and host-based routing.

Installing Ingress on minikube

To install minikube type this comand to install

minikube addons enable ingress

```
PS C:\Users\Niree> 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
💡  After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.11.3
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.4
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
```

Then need to check type this command

kubectl get pods -n kube-system

Minikube addons list

```
PS C:\Users\Niree> minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)         |
| amd-gpu-device-plugin       | minikube | disabled     | 3rd party (AMD)                |
| auto-pause                  | minikube | disabled     | minikube                       |
| cloud-spanner               | minikube | disabled     | Google                         |
| csi-hostpath-driver         | minikube | disabled     | Kubernetes                     |
```

```
|                             |          |              | [info@inaccel.com])            |
| ingress                     | minikube | enabled ✅   | Kubernetes                     |
| ingress-dns                 | minikube | disabled     | minikube                       |
| inspektor-gadget            | minikube | disabled     | 3rd party                      |
|                             |          |              | (inspektor-gadget.io)          |
| istio                       | minikube | disabled     | 3rd party (Istio)              |
| istio-provisioner           | minikube | disabled     | 3rd party (Istio)              |
| kong                        | minikube | disabled     | 3rd party (Kong HQ)            |
| kubeflow                    | minikube | disabled     | 3rd party                      |
| kubevirt                    | minikube | disabled     | 3rd party (KubeVirt)           |
| logviewer                   | minikube | disabled     | 3rd party (unknown)            |
| metallb                     | minikube | disabled     | 3rd party (MetalLB)            |
| metrics-server              | minikube | disabled     | Kubernetes                     |
| nvidia-device-plugin        | minikube | disabled     | 3rd party (NVIDIA)             |
| nvidia-driver-installer     | minikube | disabled     | 3rd party (NVIDIA)             |
| nvidia-gpu-device-plugin    | minikube | disabled     | 3rd party (NVIDIA)             |
```

It will shows ingress is enable

Then create app1.yml file and app2.yml file with nginx and in that yml file its self nedd to deployment file and servicefile

C: > Users > Niree > ! app1.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: app1
  template:
    metadata:
      labels:
        app: app1
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: app1-service
spec:
  selector:
    app: app1
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

app1.yml

C: > Users > Niree > ! app2.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: app2
  template:
    metadata:
      labels:
        app: app2
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: app2-service
spec:
  selector:
    app: app2
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

app2.yml

```
PS C:\Users\Niree> kubectl apply -f app1.yml
deployment.apps/app1 created
deployment.apps/app1 created
service/app1-service created
```

app1.yml file created

```
PS C:\Users\Niree> kubectl apply -f app2.yml
PS C:\Users\Niree> kubectl apply -f app2.yml
deployment.apps/app2 created
deployment.apps/app2 created
service/app2-service created
PS C:\Users\Niree>
```

app2.yml file created

Create Ingress.yml file in name of myapp.local

C: > Users > Niree > ! ingress.yml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
    - host: myapp.local
      http:
        paths:
          - path: /app1
            pathType: Prefix
            backend:
              service:
                name: app1-service
                port:
                  number: 80
          - path: /app2
            pathType: Prefix
            backend:
              service:
                name: app2-service
                port:
                  number: 80
```

powershell

```
PS C:\Users\Niree> kubectl apply -f ingress.yml
ingress.networking.k8s.io/my-ingress created
PS C:\Users\Niree> minikube ip
192.168.49.2
PS C:\Users\Niree>
```

C: > Users > Niree > OneDrive > Documents > hosts.txt

```
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost
# Added by Docker Desktop
10.0.12.74 host.docker.internal
10.0.12.74 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
192.168.49.2 myapp.local
# End of section
```

In windows Power shell go to this path and add this minikube ip and app name and save it .

Get-Content C:\Windows\System32\drivers\etc\hosts

If you Type This Command means you will get content of the host in terminal

```
PS C:\Users\Niree> minikube ip
192.168.49.2
PS C:\Users\Niree> Get-Content C:\Windows\System32\drivers\etc\hosts
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

#       127.0.0.1       localhost
#       ::1             localhost
# Added by Docker Desktop
10.0.12.74 host.docker.internal
10.0.12.74 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
```

```
PS C:\Users\Niree> kubectl run test-ingress-pod --image=busybox --restart=Never --rm -it -- /bin/sh
```

kubectl run test-ingress-pod --image=busybox --restart=Never --rm -it -- /bin/sh

Then Runtest ingress command

```
If you don't see a command prompt, try pressing enter.
/ # wget -O- http://app1-service
Connecting to app1-service (10.108.169.17:80)
```

in that type this **wget -O- <u>http://app1-service</u>** it will connect and welcome to nginx

app1 nginx content

```
PS C:\Users\Niree> kubectl run test-ingress-pod --image=busybox --restart=Never --rm -it -- /bin/sh
If you don't see a command prompt, try pressing enter.
/ # wget -O- http://app1-service
Connecting to app1-service (10.108.169.17:80)
writing to stdout
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
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>
-                    100% |*******************************************************************************|   615  0:00:00 ETA
written to stdout
/ #
```

in that type this **wget -O- http://app2-service** it will connect and welcome to nginx

```
/ # wget -O- http://app2-service
Connecting to app2-service (10.104.109.164:80)
writing to stdout
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
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>
-                    100% |*******************************************************************************|   615  0:00:00 ETA
written to stdout
/ #
```

app2 nginx content

Finally Ingress is working

Creatin ingress with nginx in clusterip mode

- First create two app.yml file for nginx application and inside its self create service.yml file also
- then create ingress.yml and inside that specify the port and name for app thoes things
- then in terminal create thies three files
- and get minikube ip
- then add that ip and app name in one path
- that path is **Get-Content C:\Windows\System32\drivers\etc\hosts**
- you can add in both wave
- one way go in windows setp by step and add
- or
- just type that command and add
- after that save it
- then type runtest pod command
- **kubectl run test-ingress-pod --image=busybox --restart=Never --rm -it -- /bin/sh**
- inside that type this command
- wget -O- http://app1-service
- Then it displlay the nginx data
- Finally its working