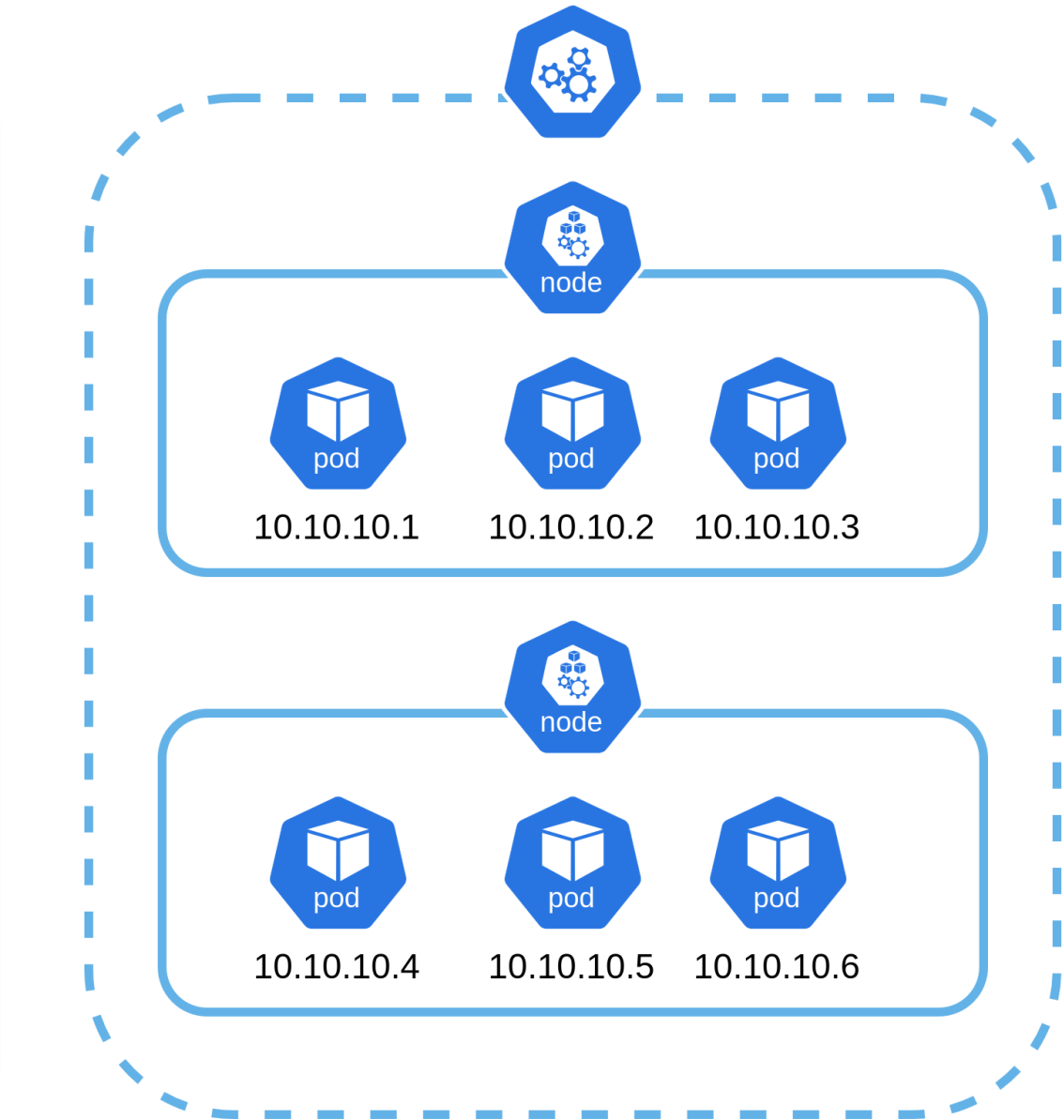


Kubernetes Network

Dalam mengakses pod atau kontainer dalam kubernetes, kita dapat menggunakan beberapa cara atau metode.

1. ClusterIP

ClusterIP digunakan ketika kita ingin menghubungkan antar pods dalam satu cluster kubernetes. Ketika kita ingin menggunakan metode ini, kita bisa langsung menggunakan ketika kita pertama kali membuat sebuah deployment atau pods. Yang mana akses terhadap kontainer bisa dibatasi sesuai dengan kebutuhan dari cluster kubernetes.



IP yang digunakan juga merupakan alokasi secara otomatis dari ClusterIP milik kubernetes. Jadi kita bisa mengakses kontainer tersebut dengan menggunakan kontainer lain.

Apabila kita ingin me-redirect port tertentu, kita bisa menggunakan service untuk men-define secara langsung port mana yang dipakai.

Berikut merupakan contoh implementasi dari ClusterIP secara langsung

ubuntu-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu
  labels:
    app: ubuntu
spec:
  containers:
    - name: ubuntu
      image: ubuntu:latest
      command: ["/bin/sleep", "3650d"]
      imagePullPolicy: IfNotPresent
      restartPolicy: Always
```

Pada file konfigurasi diatas, kita bisa melihat bahwa kita akan membuat sebuah container dengan menggunakan image dari ubuntu dengan tag latest. Kontainer tersebut diberikan label

```
app: ubuntu
```

dengan tujuan nantinya akan didefine juga ke service yang melakukan redirect port ke kontainer tersebut.

Untuk menggunakan file tersebut, kita bisa menggunakan perintah

```
kubectl apply -f ubuntu-pod.yaml
```

Ketika file tersebut digunakan, maka outputnya akan menjadi seperti ini.

```
ex-dec@ex-pc:~/project/kubernetes
ex-dec@ex-pc:~/project/kubernetes> kubectl apply -f ubuntu-pod.yaml
pod/ubuntu created
ex-dec@ex-pc:~/project/kubernetes> kubectl describe pods ubuntu
Name:          ubuntu
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Sat, 27 Jan 2024 11:36:33 +0700
Labels:        app=ubuntu
Annotations:    <none>
Status:        Running
IP:            10.244.0.83
IPs:
  IP: 10.244.0.83
Containers:
  ubuntu:
    Container ID:  docker://3e543a7bd1e411217c57dda2e4bfc8e97f7b4d42db1e895d99580198e4d84572
    Image:         ubuntu:latest
    Image ID:      docker-pullable://ubuntu@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd060a74
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sleep
      3650d
    State:         Running
      Started:     Sat, 27 Jan 2024 11:36:35 +0700
    Ready:         True
    Restart Count: 0
```

Pada detail dari pods tersebut, kita bisa melihat bahwa kontainer tersebut memiliki IP

```
10.244.0.83
```

Hal itu karena secara default dari kubernetes akan melakukan assign IP secara random ke kontainer tersebut.

Untuk selanjutnya kita akan menggunakan service untuk melakukan redirect terhadap port yang sudah kita tentukan. Buat file berikut

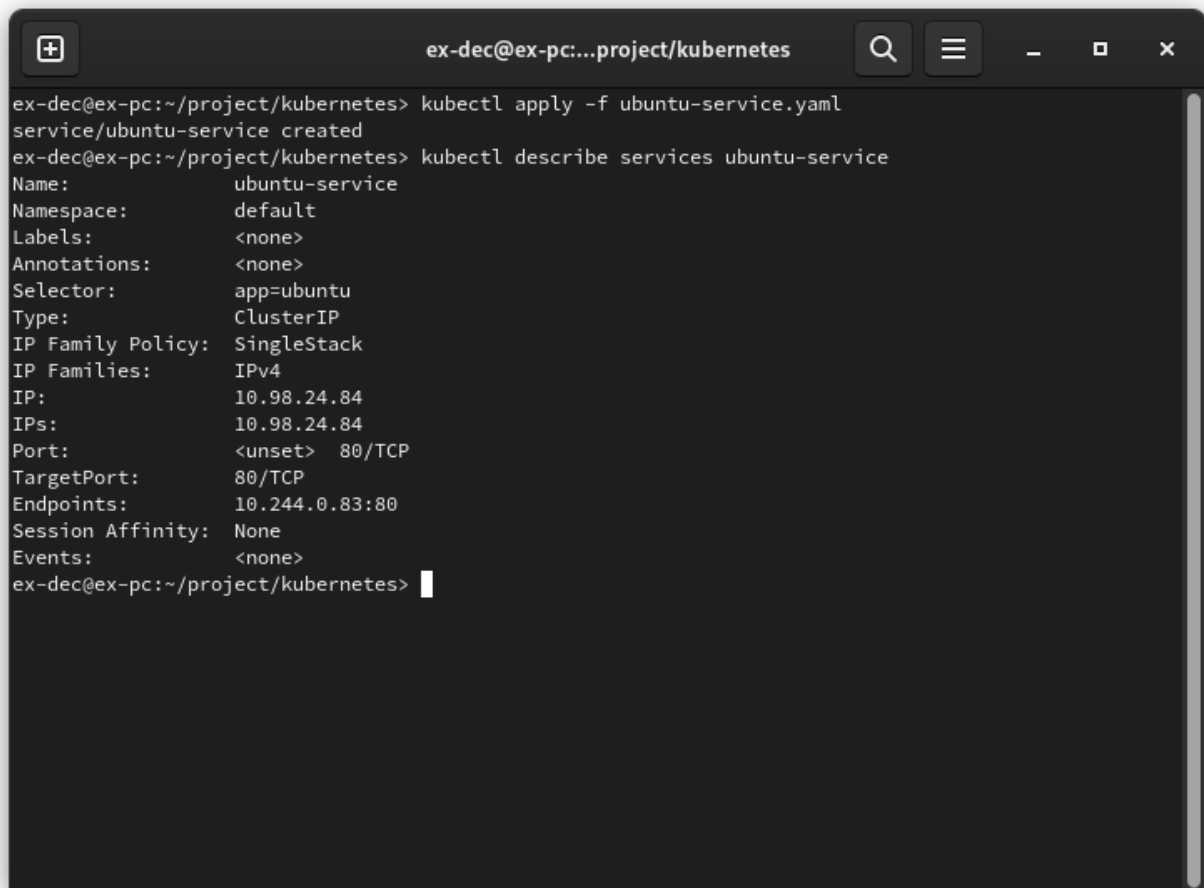
ubuntu-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: ubuntu-service
spec:
  selector:
    app: ubuntu
  ports:
    - protocol: TCP
      port: 80
```

Terapkan file tersebut pada kubernetes

```
kubectl apply -f ubuntu-service.yaml
```

Hasil nya akan seperti ini

A terminal window titled 'ex-dec@ex-pc:...project/kubernetes' showing the execution of kubectl commands. The first command is 'kubectl apply -f ubuntu-service.yaml', which results in 'service/ubuntu-service created'. The second command is 'kubectl describe services ubuntu-service', which displays detailed information about the service. The output shows the service is named 'ubuntu-service' in the 'default' namespace, with a selector 'app=ubuntu' and type 'ClusterIP'. It has a single IP address '10.98.24.84' on port '80/TCP', with a target port of '80/TCP' and endpoints at '10.244.0.83:80'.

```
ex-dec@ex-pc:~/project/kubernetes> kubectl apply -f ubuntu-service.yaml
service/ubuntu-service created
ex-dec@ex-pc:~/project/kubernetes> kubectl describe services ubuntu-service
Name:         ubuntu-service
Namespace:    default
Labels:       <none>
Annotations:  <none>
Selector:     app=ubuntu
Type:         ClusterIP
IP Family Policy: SingleStack
IP Families:  IPv4
IP:           10.98.24.84
IPs:          10.98.24.84
Port:         <unset> 80/TCP
TargetPort:   80/TCP
Endpoints:    10.244.0.83:80
Session Affinity: None
Events:       <none>
ex-dec@ex-pc:~/project/kubernetes>
```

Pada detail tersebut, dapat kita lihat bahwa semua traffic pada port 80 dengan IP Address

```
10.99.24.84
```

akan dialihkan ke IP yang dimiliki oleh pod ubuntu diatas. Ketika kita melakukan akses ke ip tersebut juga akan memiliki output yang sama.

Testing dari pod lain pada satu cluster yang sama

```
ex-dec@ex-pc:...project/kubernetes
ex-dec@ex-pc:~/project/kubernetes> kubectl exec ubuntu2 -- curl 10.244.0.83
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100  265  100  265    0    0  492k      0 --:--:-- --:--:-- --:--:-- 258k
<!DOCTYPE html>
<html>
<head>
<title>Ini web milik pod ubuntu!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
</body>
</html>
ex-dec@ex-pc:~/project/kubernetes>
```

```
ex-dec@ex-pc:...project/kubernetes
ex-dec@ex-pc:~/project/kubernetes> kubectl exec ubuntu2 -- curl 10.98.24.84
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100  265  100  265    0    0  464k      0 --:--:-- --:--:-- --:--:-- 258k
<!DOCTYPE html>
<html>
<head>
<title>Ini web milik pod ubuntu!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
</body>
</html>
ex-dec@ex-pc:~/project/kubernetes>
```

Dari hasil testing keduanya memiliki output yang sama, yang berarti keduanya mengakses pod yang sama pada port 80.

2. NodePort Sharing

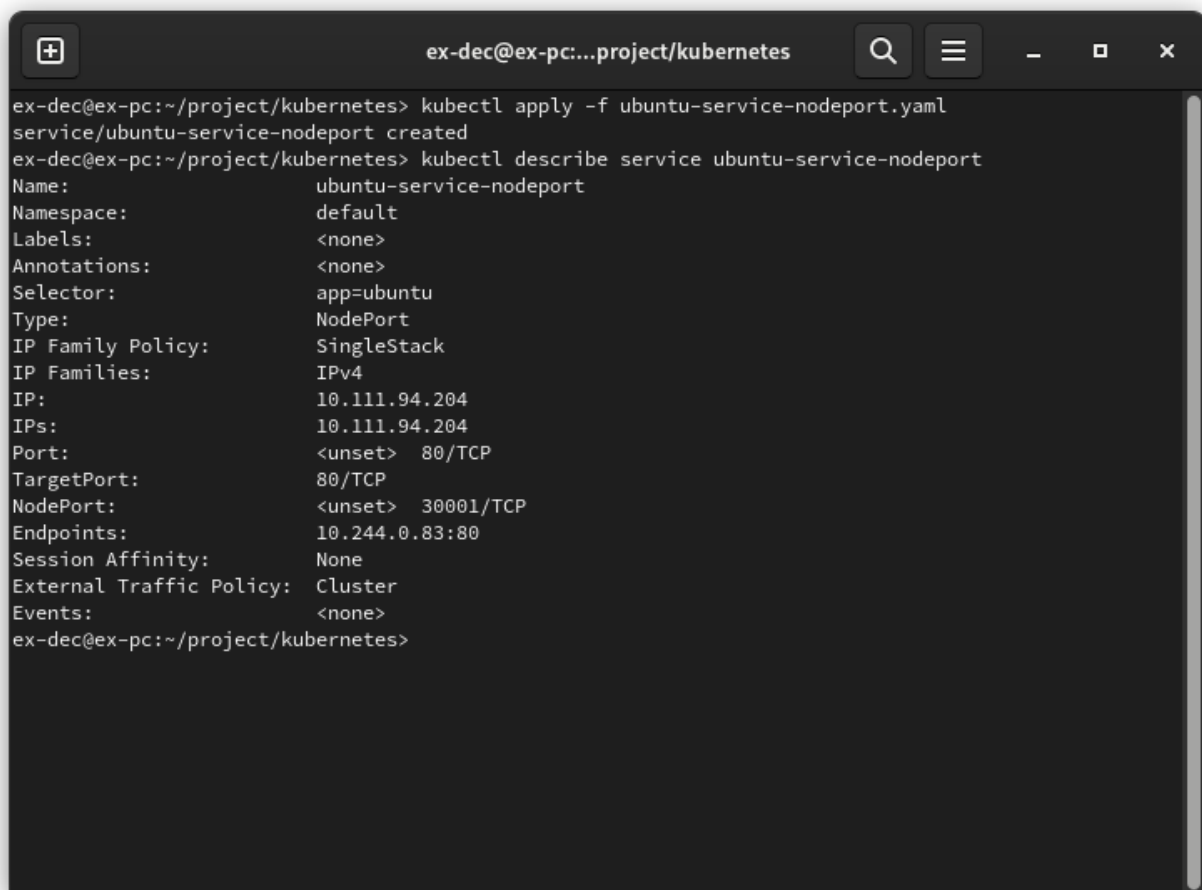
Selanjutnya adalah NodePort Sharing. NodePort sharing memiliki konsep yang sedikit berbeda dengan ClusterIP sebelumnya. Perbedaannya hanya ada pada service yang dibentuk. kontainer yang kita miliki bisa diakses melalui jaringan luar yang langsung terhubung dengan Node. Sistemnya seperti kontainer pada docker yang melakukan publish port, sehingga bisa diakses dari jaringan diluar cluster.

contoh konfigurasi service yang menerapkan metode ini seperti berikut

ubuntu-service-nodeport.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: ubuntu-service-nodeport
spec:
  type: NodePort
  selector:
    app: ubuntu
  ports:
    - protocol: TCP
      port: 80
      nodePort: 30001
```

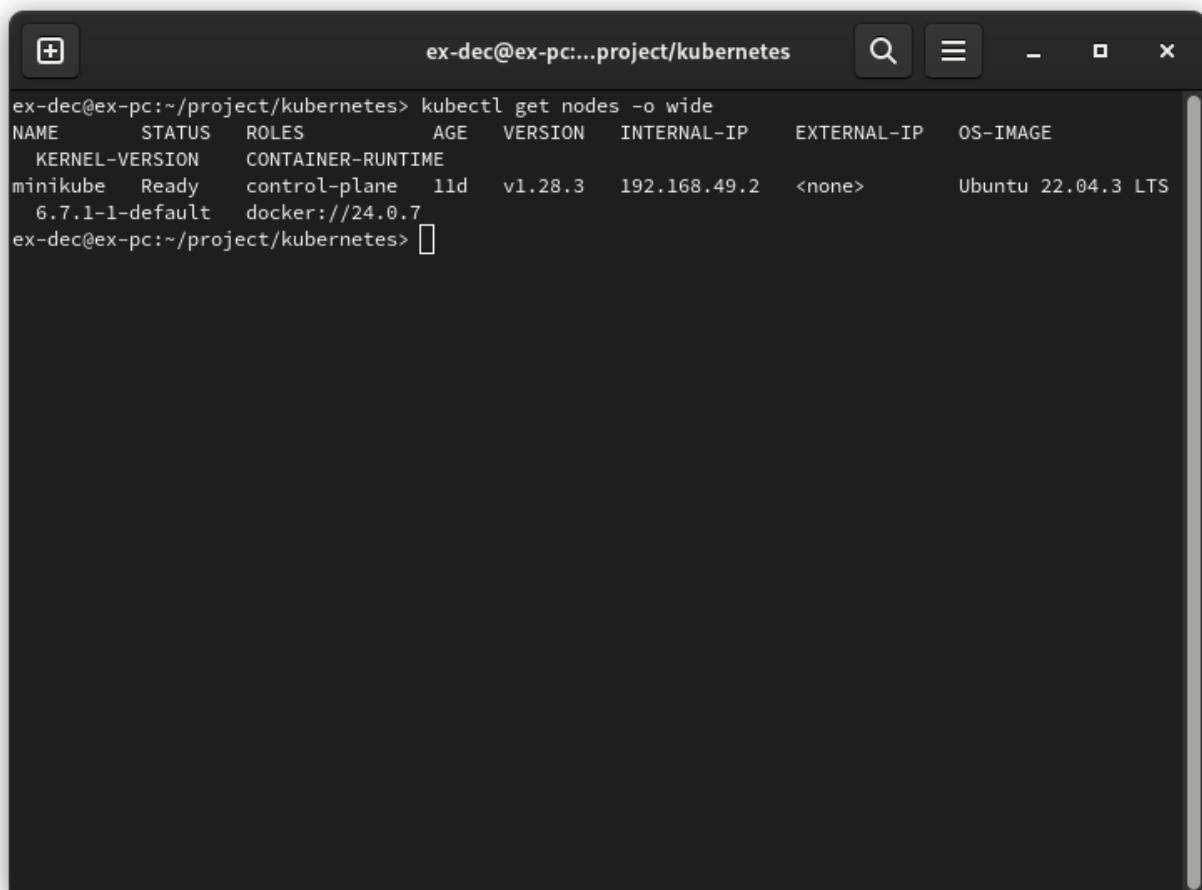
Setelah konfigurasi tersebut diterapkan, maka hasilnya akan menjadi seperti berikut

A terminal window titled 'ex-dec@ex-pc:...project/kubernetes' with search, menu, and window control icons. It shows the execution of 'kubectl apply -f ubuntu-service-nodeport.yaml' and 'kubectl describe service ubuntu-service-nodeport'. The output of the describe command lists service details: Name (ubuntu-service-nodeport), Namespace (default), Labels (none), Annotations (none), Selector (app=ubuntu), Type (NodePort), IP Family Policy (SingleStack), IP Families (IPv4), IP (10.111.94.204), IPs (10.111.94.204), Port (unset 80/TCP), TargetPort (80/TCP), NodePort (unset 30001/TCP), Endpoints (10.244.0.83:80), Session Affinity (None), External Traffic Policy (Cluster), and Events (none).

```
ex-dec@ex-pc:~/project/kubernetes> kubectl apply -f ubuntu-service-nodeport.yaml
service/ubuntu-service-nodeport created
ex-dec@ex-pc:~/project/kubernetes> kubectl describe service ubuntu-service-nodeport
Name:                ubuntu-service-nodeport
Namespace:            default
Labels:               <none>
Annotations:          <none>
Selector:             app=ubuntu
Type:                 NodePort
IP Family Policy:     SingleStack
IP Families:          IPv4
IP:                   10.111.94.204
IPs:                  10.111.94.204
Port:                 <unset> 80/TCP
TargetPort:           80/TCP
NodePort:             <unset> 30001/TCP
Endpoints:            10.244.0.83:80
Session Affinity:     None
External Traffic Policy: Cluster
Events:               <none>
ex-dec@ex-pc:~/project/kubernetes>
```

Untuk membuktikan hasil konfigurasi tersebut, kita bisa mengakses web dari laptop kita sebagai host secara langsung. Sebelum nya kita cek terlebih dahulu ip yang dimiliki oleh node kita.

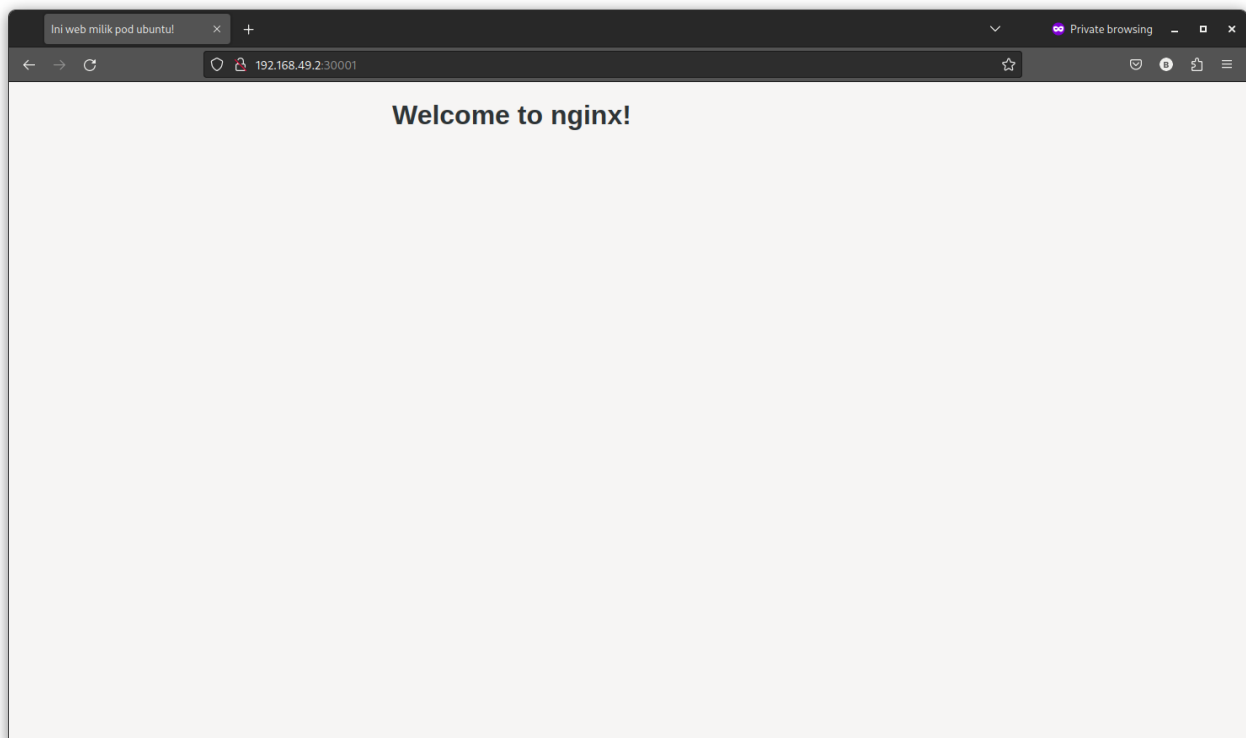
```
kubectl get nodes -o wide
```



```
ex-dec@ex-pc:~/project/kubernetes> kubectl get nodes -o wide
NAME          STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
minikube      Ready     control-plane 11d   v1.28.3   192.168.49.2   <none>        Ubuntu 22.04.3 LTS
6.7.1-1-default  docker://24.0.7
```

The terminal window shows the command `kubectl get nodes -o wide` being executed. The output is a table with columns: NAME, STATUS, ROLES, AGE, VERSION, INTERNAL-IP, EXTERNAL-IP, and OS-IMAGE. The first row shows a node named 'minikube' with status 'Ready', role 'control-plane', age '11d', version 'v1.28.3', internal IP '192.168.49.2', and OS 'Ubuntu 22.04.3 LTS'. The second row shows the container runtime '6.7.1-1-default' with 'docker://24.0.7'.

Selanjutnya kita bisa langsung lakukan testing pada browser kita

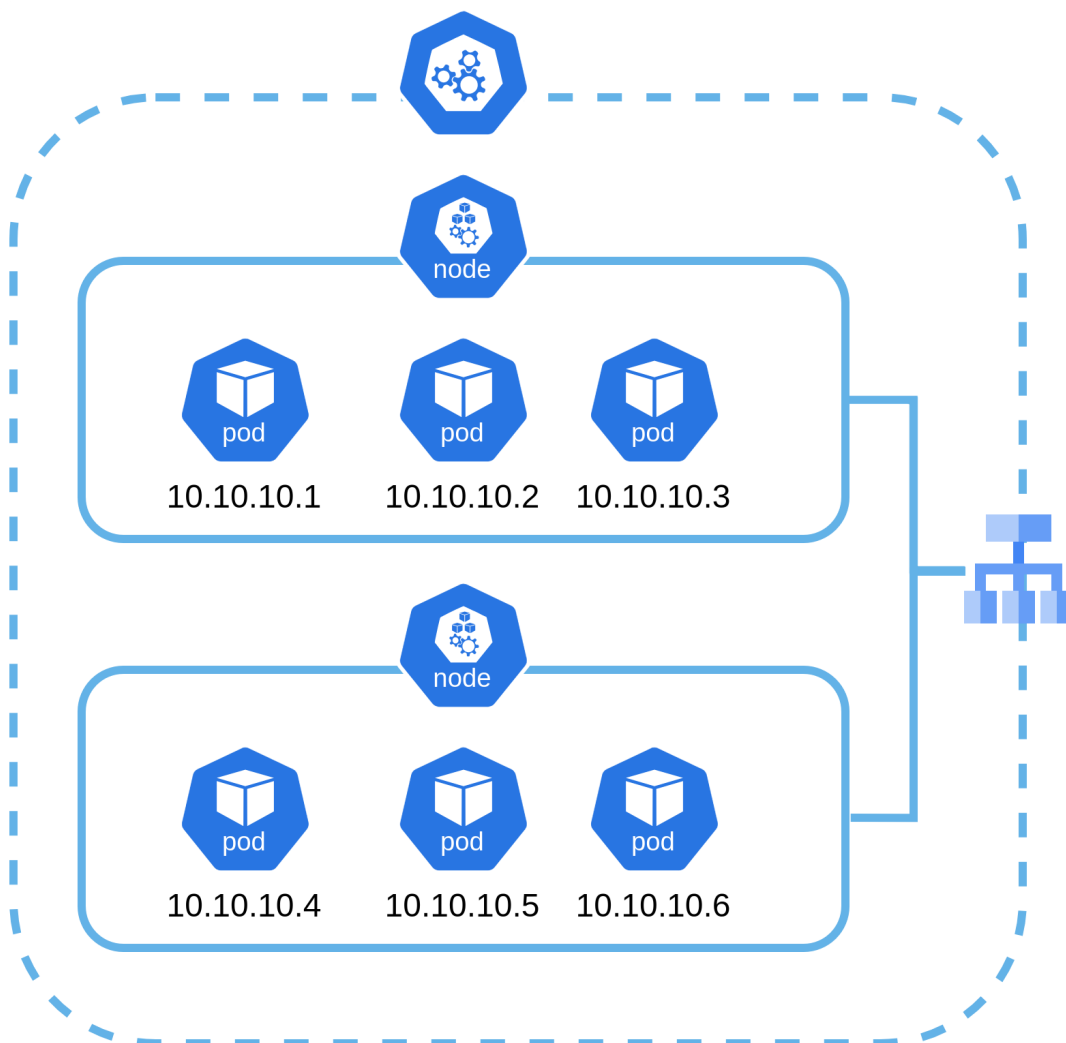


Metode ini memiliki beberapa kelebihan dan kekurangan. Kelebihannya adalah kita bisa melakukan direct access ke sebuah resource yang ada di cluster kita, namun di sisi lain, karena memang aksesnya langsung ke IP yang dimiliki oleh node kita, maka kita tidak bisa melakukan expose secara

masif menggunakan nodeport. Satu lagi kekurangan dari metode ini adalah redirect port yang dilakukan hanya bisa menggunakan range 30000-32767. Jadi metode ini lebih cocok digunakan untuk testing environment yang sudah dibuat agar tidak terlalu banyak yang dikonfigurasi terlebih dahulu.

3. Load Balancer

Metode selanjutnya adalah menggunakan load balancer. Seperti load balancer pada umumnya, jadi kita bisa membuat sebuah load balancer yang nantinya berperan sebagai gateway dari jaringan luar untuk mengakses layanan dalam cluster kita. Untuk implementasi sekarang, kita hanya membuat sebuah service yang digunakan untuk handle load balancer tersebut. Ada bahasan lain yang akan mendukung load balancer ini yaitu endpoint, akan kita bahas pada materi selanjutnya.



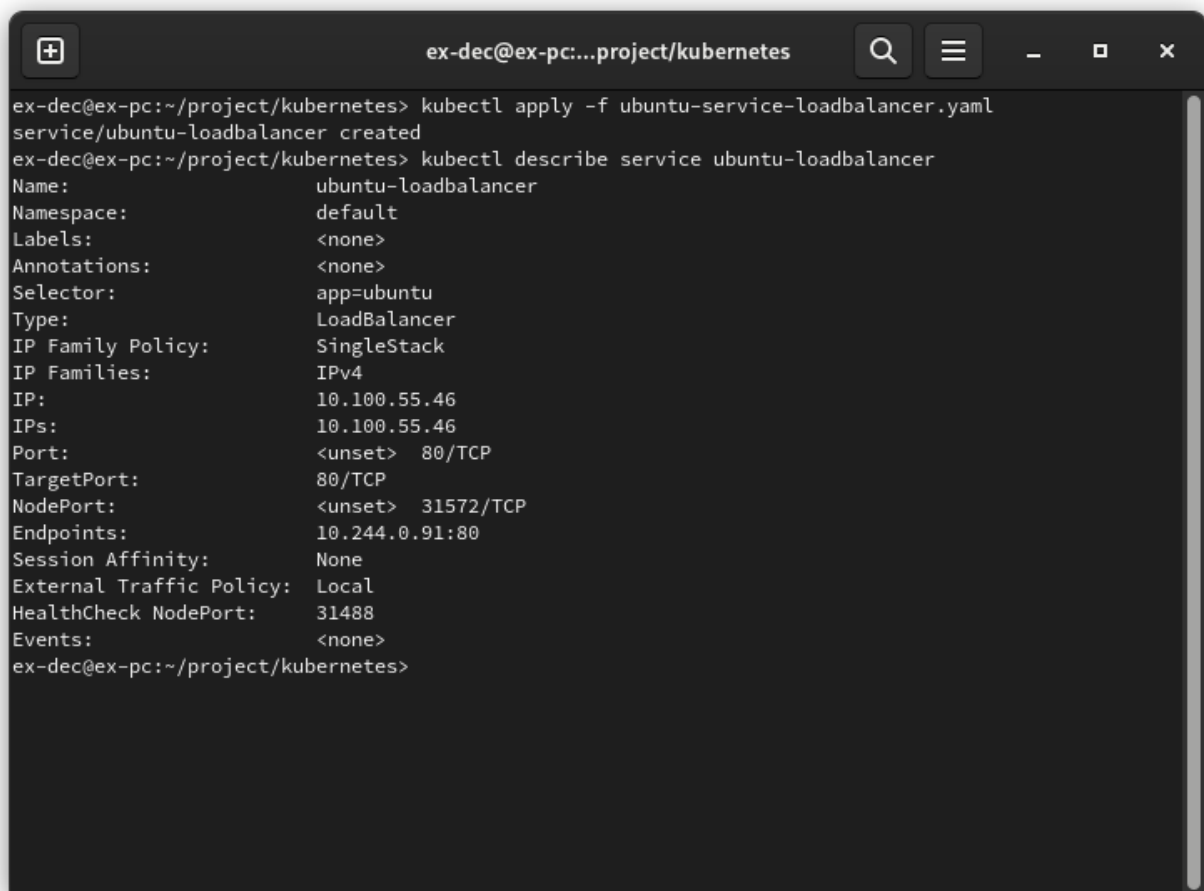
Berikut adalah contoh konfigurasi untuk digunakan sebagai load balancer.

ubuntu-service-loadbalancer.yaml

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: ubuntu-loadbalancer
spec:
  type: LoadBalancer
  selector:
    app: ubuntu
  ports:
    - port: 80
  externalTrafficPolicy: Local
```

Jangan lupa apply konfigurasi tersebut dan hasilnya akan menjadi seperti ini.

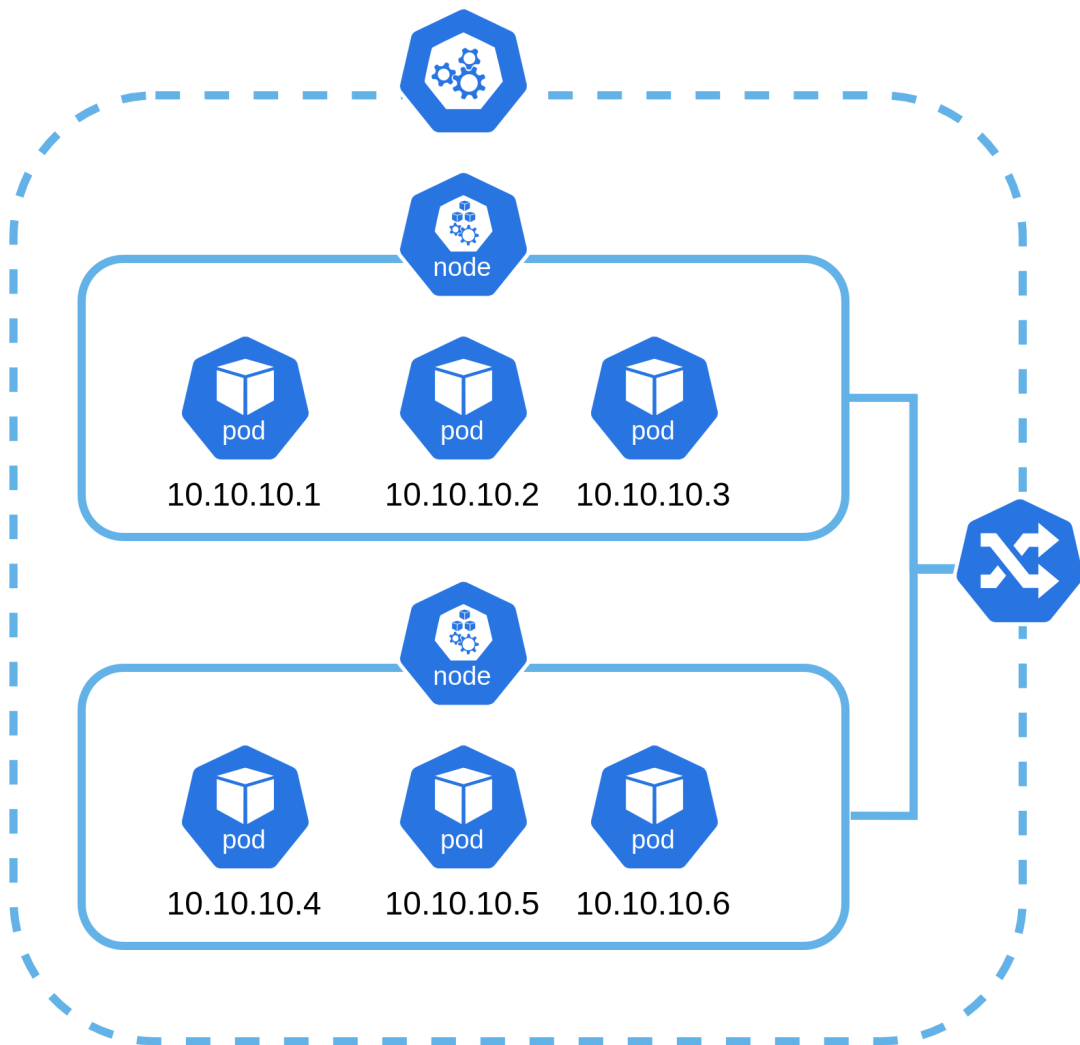
A terminal window titled 'ex-dec@ex-pc:...project/kubernetes' shows the following commands and output:

```
ex-dec@ex-pc:~/project/kubernetes> kubectl apply -f ubuntu-service-loadbalancer.yaml
service/ubuntu-loadbalancer created
ex-dec@ex-pc:~/project/kubernetes> kubectl describe service ubuntu-loadbalancer
Name:                ubuntu-loadbalancer
Namespace:            default
Labels:               <none>
Annotations:          <none>
Selector:             app=ubuntu
Type:                 LoadBalancer
IP Family Policy:     SingleStack
IP Families:          IPv4
IP:                   10.100.55.46
IPs:                  10.100.55.46
Port:                 <unset> 80/TCP
TargetPort:           80/TCP
NodePort:             <unset> 31572/TCP
Endpoints:            10.244.0.91:80
Session Affinity:     None
External Traffic Policy: Local
HealthCheck NodePort: 31488
Events:               <none>
ex-dec@ex-pc:~/project/kubernetes>
```

Konfigurasi dari load balancer ini tidak lengkap apabila tidak ditambahkan endpoint tersendiri ke load balancer tersebut. Namun kita tidak akan membahas hal tersebut kali ini.

4. Ingress

Selanjutnya adalah menggunakan ingress. Ingress adalah salah satu metode yang bisa kita gunakan untuk menghubungkan cluster kita dengan jaringan luar. Konsepnya hampir sama dengan load balancer, namun ingress mengambil konsep dari gateway API yang melakukan mapping directory. Jadi masing-masing endpoint yang sudah dibuat di dalam cluster dapat diakses melalui pengolahan domain di kubernetes sendiri.



Untuk implementasinya, kita harus mengaktifkan ingress terlebih dahulu dengan menggunakan minikube.

```
minikube addons enable ingress
```

```
ex-dec@ex-pc:~/project/private-notes> minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	3rd party (Ambassador)
auto-pause	minikube	disabled	minikube
cloud-spanner	minikube	disabled	Google
csi-hostpath-driver	minikube	disabled	Kubernetes
dashboard	minikube	enabled ✓	Kubernetes
default-storageclass	minikube	enabled ✓	Kubernetes
efk	minikube	disabled	3rd party (Elastic)
freshpod	minikube	disabled	Google
gcp-auth	minikube	disabled	Google
gvisor	minikube	disabled	minikube
headlamp	minikube	disabled	3rd party (kinvolk.io)
helm-tiller	minikube	disabled	3rd party (Helm)
inacel	minikube	disabled	3rd party (InAccel [info@inacel.com])
ingress	minikube	disabled	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 (MetaLB)
metrics-server	minikube	enabled ✓	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)
olm	minikube	disabled	3rd party (Operator Framework)
pod-security-policy	minikube	disabled	3rd party (unknown)
portainer	minikube	disabled	3rd party (Portainer.io)
registry	minikube	disabled	minikube
registry-aliases	minikube	disabled	3rd party (unknown)
registry-creds	minikube	disabled	3rd party (UPMC Enterprises)
storage-provisioner	minikube	enabled ✓	minikube
storage-provisioner-gluster	minikube	disabled	3rd party (Gluster)
storage-provisioner-rancher	minikube	disabled	3rd party (Rancher)
volumesnapshots	minikube	disabled	Kubernetes

```
ex-dec@ex-pc:~/project/private-notes> 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>

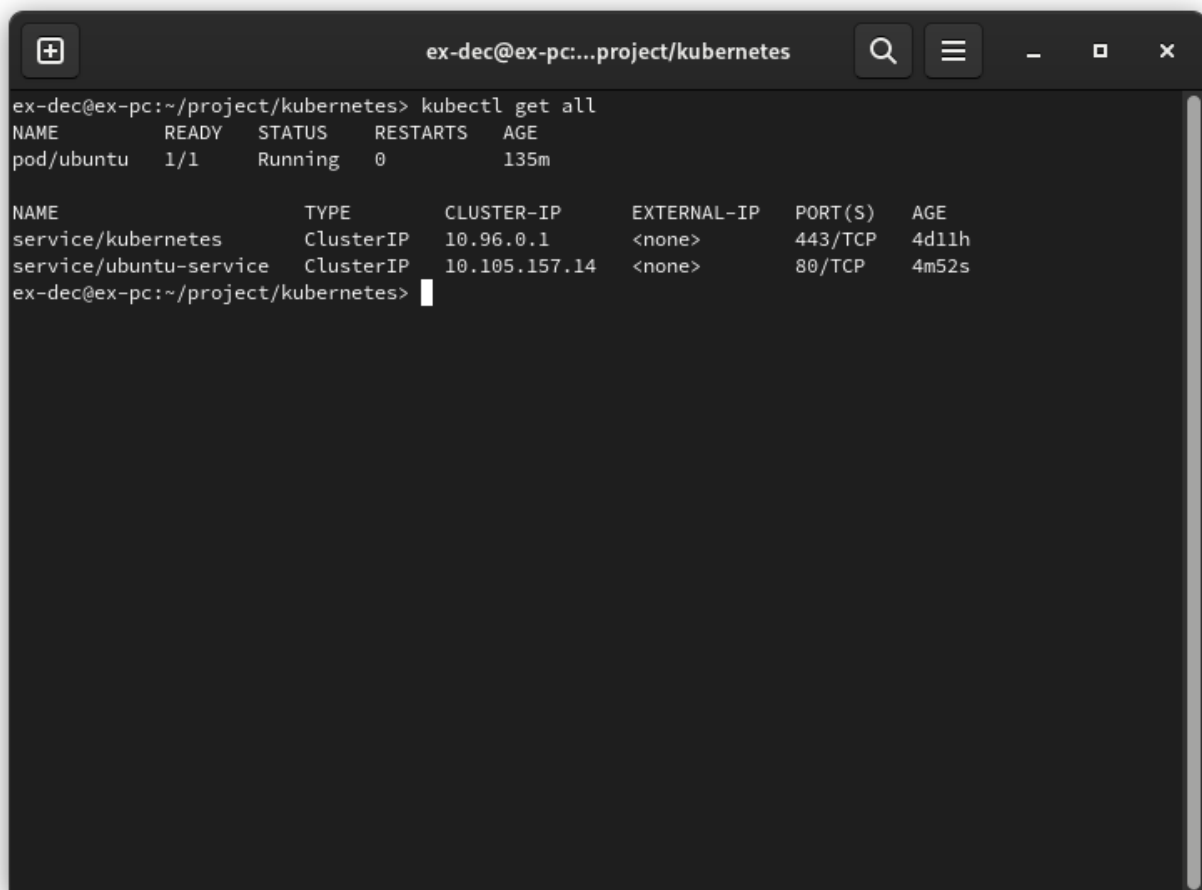
- Using image registry.k8s.io/ingress-nginx/controller:v1.9.4
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
- Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0

🔍 Verifying ingress addon...

🌟 The 'ingress' addon is enabled

```
ex-dec@ex-pc:~/project/private-notes> 
```

Setelah ingress aktif, siapkan pod yang akan dibuka akses ke luar. Pastikan pod sudah berjalan dan sudah dijalankan dengan servicenya. Kita cukup menggunakan service dengan ClusterIP untuk melakukan expose port dari pod nya.

A terminal window titled 'ex-dec@ex-pc:...project/kubernetes' with search, menu, and window control icons. The command 'kubectl get all' has been executed, displaying two tables. The first table shows a single pod 'pod/ubuntu' in a 'Running' state. The second table shows two services: 'service/kubernetes' and 'service/ubuntu-service'.

```
ex-dec@ex-pc:~/project/kubernetes> kubectl get all
NAME                READY   STATUS    RESTARTS   AGE
pod/ubuntu           1/1     Running   0           135m

NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes  ClusterIP   10.96.0.1    <none>        443/TCP   4d11h
service/ubuntu-service ClusterIP   10.105.157.14 <none>        80/TCP    4m52s
ex-dec@ex-pc:~/project/kubernetes>
```

Setelah kedua hal tersebut sudah berjalan, sekarang kita siapkan ingress nya.

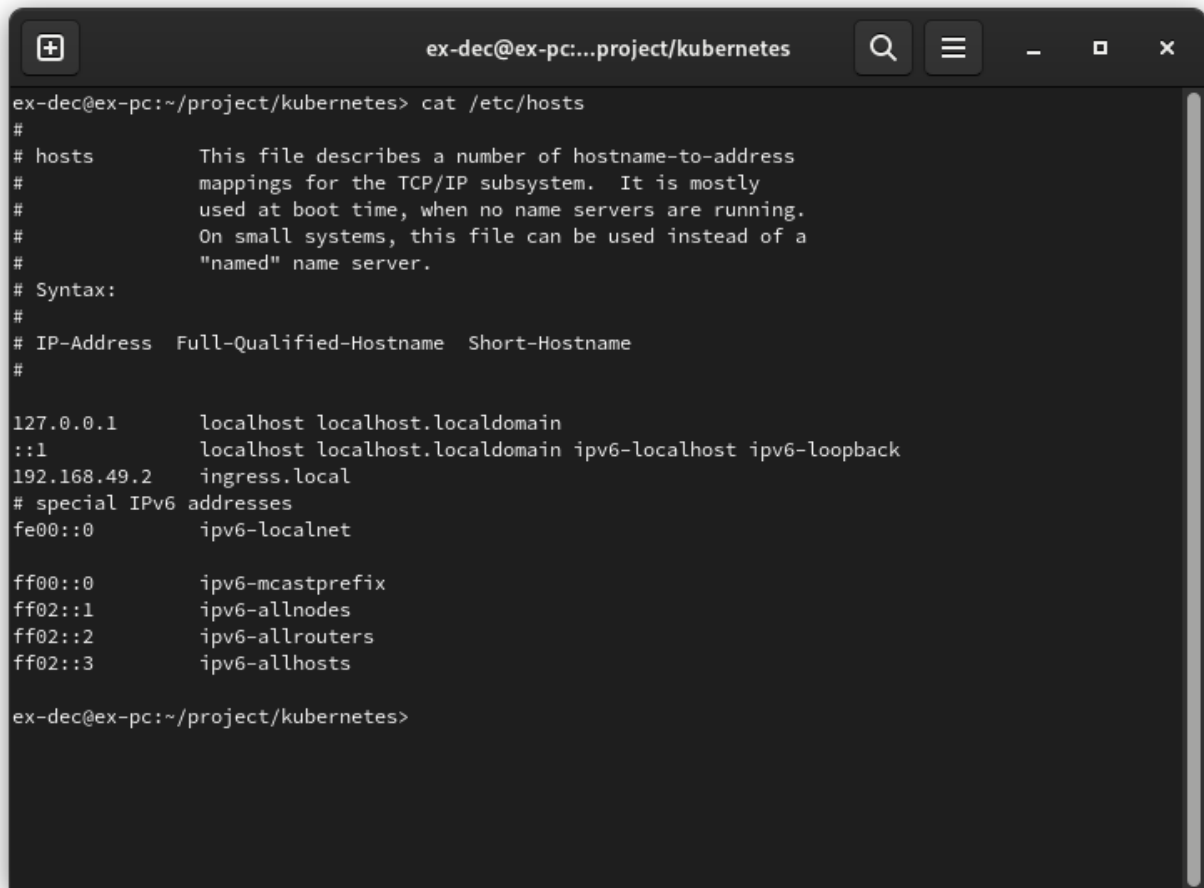
ubuntu-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ubuntu-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: ingress.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: ubuntu-service
            port:
              number: 80
```

Terapkan konfigurasi tersebut dan konfigurasi hostname agar bisa terbaca di host kita. Untuk host saya sendiri perlu mengkonfigurasi pada file `/etc/hosts`.

Disini saya akan menggunakan host 'ingress.local' sesuai dengan konfigurasi diatas.

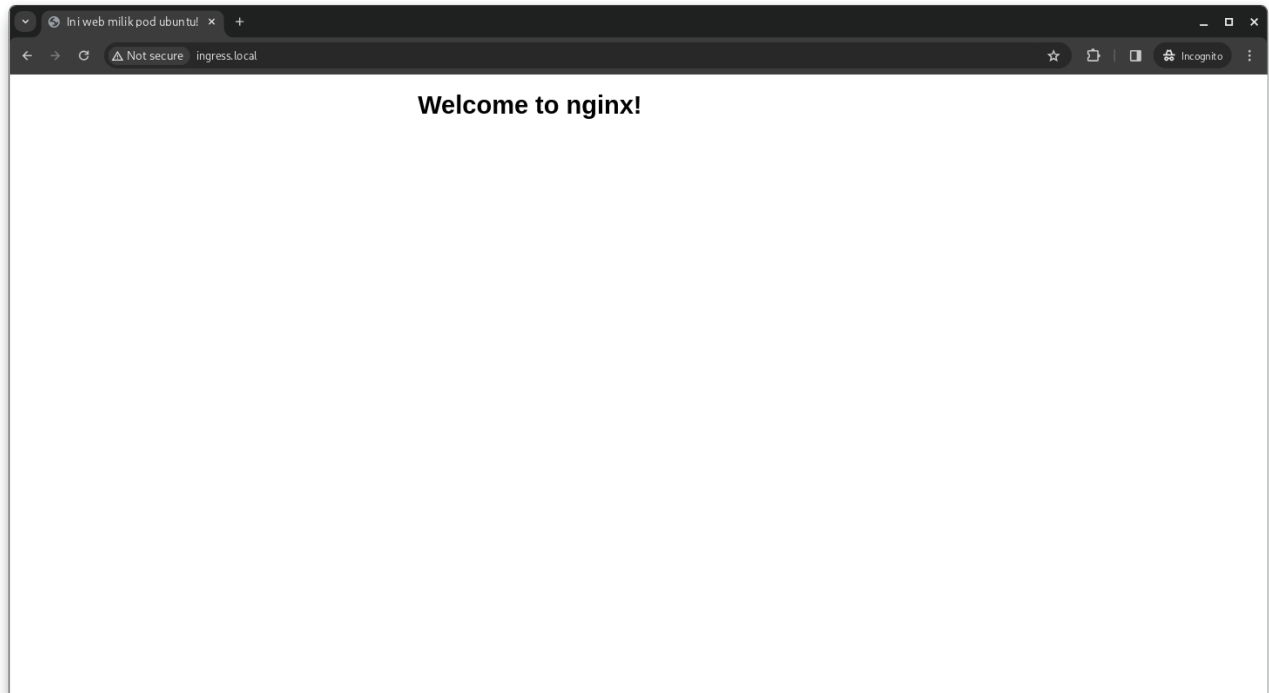
`/etc/hosts`

A terminal window titled 'ex-dec@ex-pc:...project/kubernetes' with search, menu, and window control icons. The terminal shows the command 'cat /etc/hosts' and its output. The output lists standard system host entries including localhost, IPv6 loopback, and special IPv6 addresses, and adds a new entry for 'ingress.local' at IP 192.168.49.2.

```
ex-dec@ex-pc:~/project/kubernetes> cat /etc/hosts
#
# hosts                This file describes a number of hostname-to-address
#                        mappings for the TCP/IP subsystem.  It is mostly
#                        used at boot time, when no name servers are running.
#                        On small systems, this file can be used instead of a
#                        "named" name server.
#
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#
127.0.0.1     localhost localhost.localdomain
::1          localhost localhost.localdomain ipv6-localhost ipv6-loopback
192.168.49.2  ingress.local
# special IPv6 addresses
fe00::0      ipv6-localnet
ff00::0      ipv6-mcastprefix
ff02::1      ipv6-allnodes
ff02::2      ipv6-allrouters
ff02::3      ipv6-allhosts

ex-dec@ex-pc:~/project/kubernetes>
```

Setelah itu kita coba akses melalui browser.



Ingress lebih cocok digunakan untuk layanan yang berbasis HTTP dikarenakan ingress hanya mendukung protokol HTTP/HTTPS. Untuk implementasi protokol yang lain, kubernetes telah dikembangkan menjadi gateway API yang bisa handle request yang lebih kompleks. Topik tersebut akan kita bahas pada waktu yang lain.