INGRESS

About Ingress:-

Ingress is top of services, so in realtime ingress will be used to expose outside.
In service nodeport option, we can expose application for client access with nodeport but there is difficult to remember by client if multiple applications hosted, so to avoid this another option called service load balancer will work 80 port and problem with load balancer is cost is involved because for each server load balancer will create.so here ingress is an option which will can be worked with path-based routing or name based (virtual hosting), so production env ingress will be used.
multiple domains to be pointed with help of ingress. nginx controller method is used ingress as a default in GCP, incase third-party Traefik can be used as an Ingress controller for a Kubernetes cluster and there are many like HA proxy controller method which will work as ingress controller.
https://docs.traefik.io/user-guide/kubernetes/
https://traefik.io/

Actual Readme:-

What is Ingress?
Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

```
 internet
     |
[ Ingress ]
--|-----|--
[ Services ]
```

An Ingress can be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name based virtual hosting. An Ingress controller is responsible for fulfilling the Ingress, usually with a load balancer, though it may also configure your edge router or additional frontends to help handle the traffic.

Types of Ingress?
1, Single Service Ingress :- simple service to point single domain, where it can be work as load balancer.
2, Name based Ingress :- with single public ip, multiple web services hosted.
3, Path based Ingress :- with same path multiple domain exposed.

##Single Service Ingress (There are existing Kubernetes concepts that allow you to expose a single Service)##

:- Create a pod and node port service by using manifest yml file.

#cat ingress-deploy1.yml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
  - protocol: TCP
    port: 5000
    targetPort: 80
    nodePort: 32001
```

kubectl apply -f ingress-deploy1.yml

kubectl get deployments -o wide

kubectl get pods -o wide

kubectl get svc -o wide

```
[root@ansikube manifest]# kubectl get svc -o wide
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE   SELECTOR
kubernetes      ClusterIP   10.36.0.1     <none>        443/TCP          43m   <none>
nginx-service   NodePort    10.36.7.139   <none>        5000:32001/TCP   40s   app=nginx
[root@ansikube manifest]#
```

#kubectl get nodes -o wide

```
[root@ansikube manifest]# kubectl get nodes -o wide
NAME                                      STATUS   ROLES    AGE   VERSION          INTERNAL-IP   EXTERNAL-IP      OS-IMAGE                            KE
RNEL-VERSION   CONTAINER-RUNTIME
gke-robo-default-pool-e5c2cd89-2mkz   Ready    <none>   45m   v1.13.11-gke.9   10.128.0.41   35.238.230.197   Container-Optimized OS from Google   4.
14.145+        docker://18.9.7
gke-robo-default-pool-e5c2cd89-bmnk   Ready    <none>   45m   v1.13.11-gke.9   10.128.0.44   34.67.183.232    Container-Optimized OS from Google   4.
14.145+        docker://18.9.7
gke-robo-default-pool-e5c2cd89-mmvz   Ready    <none>   45m   v1.13.11-gke.9   10.128.0.43   34.69.177.49     Container-Optimized OS from Google   4.
14.145+        docker://18.9.7
[root@ansikube manifest]#
```

:- Test whether nodeport service is working or not http://35.238.230.197:32001/

@@Create a single service ingress@@

Create a single service ingress manifest yml.

cat single-service.yml

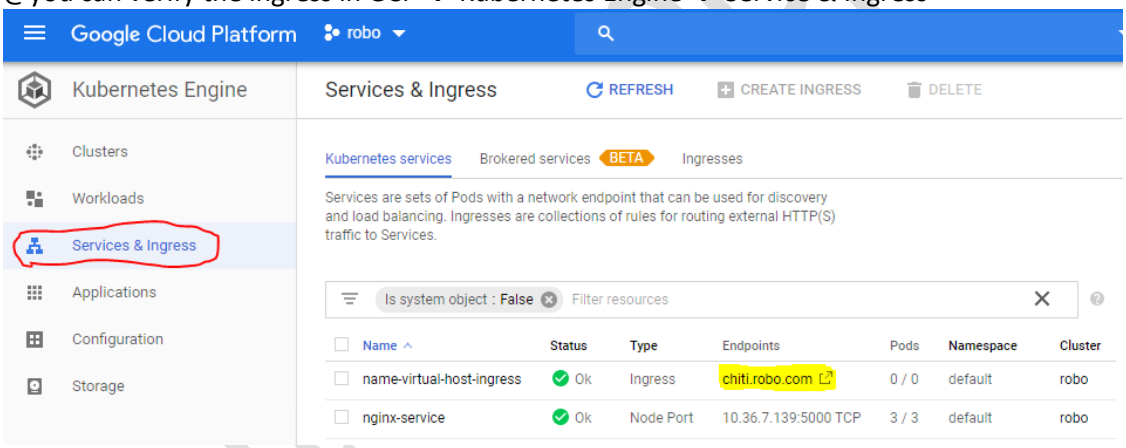
```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: chiti.robo.com
    http:
      paths:
      - backend:
          serviceName: nginx-service
          servicePort: 5000
```

kubectl apply -f single-service.yml

kubectl get ingress

```
[root@ansikube manifest]# kubectl get ingress
NAME                        HOSTS            ADDRESS          PORTS   AGE
name-virtual-host-ingress   chiti.robo.com   34.102.213.223   80      2m47s
[root@ansikube manifest]#
```

@you can verify the ingress in GCP → Kubernetes Engine → Service & Ingress



:-Test the ingress http://chiti.robo.com/

Note:- Before testing, ensure to have DNS entries on dns server or entries on local hosts file of your system.

kubectl delete -f single-service.yml

```
[root@ansikube manifest]# kubectl delete -f single-service.yml
ingress.extensions "name-virtual-host-ingress" deleted
[root@ansikube manifest]#
```

Note:- Make sure to delete single service ingress.

##Name based Ingress :- with single public ip, multiple web services hosted.##

:- Create a pod and node port service.

cat ingress-deploy2.yml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: apache
spec:
  replicas: 3
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: httpd
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  selector:
    app: apache
  type: NodePort
  ports:
  - protocol: TCP
    port: 5001
    targetPort: 80
```

kubectl apply -f ingress-deploy2.yml

kubectl get deploy -o wide

kubectl get svc -o wide

```
[root@ansikube manifest]# kubectl get svc -o wide
NAME            TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE   SELECTOR
apache-service  NodePort    10.36.7.242   <none>        5001:31550/TCP   94s   app=apache
kubernetes      ClusterIP   10.36.0.1     <none>        443/TCP          77m   <none>
nginx-service   NodePort    10.36.7.139   <none>        5000:32001/TCP   35m   app=nginx
[root@ansikube manifest]#
```

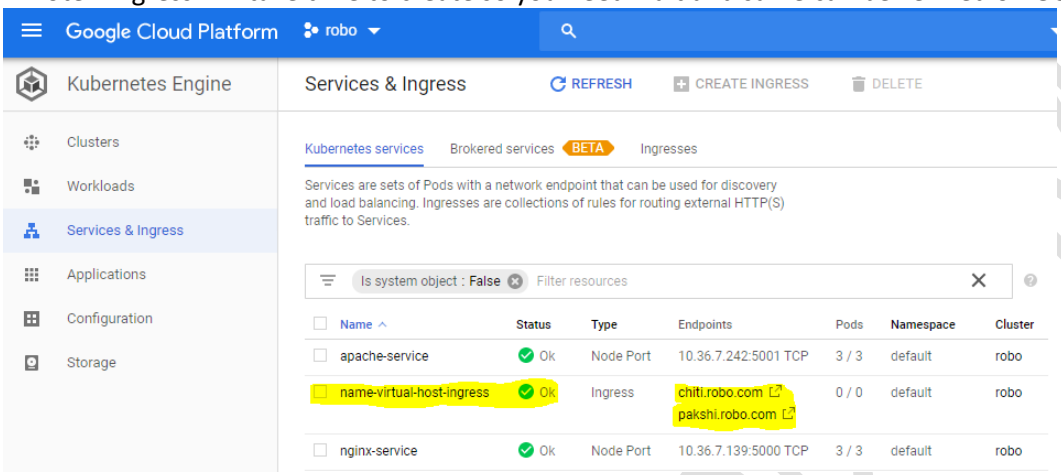
@@Create a two-service ingress@@

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: chiti.robo.com
    http:
      paths:
      - backend:
          serviceName: nginx-service
          servicePort: 5000
  - host: pakshi.robo.com
    http:
      paths:
      - backend:
          serviceName: apache-service
          servicePort: 5001
```

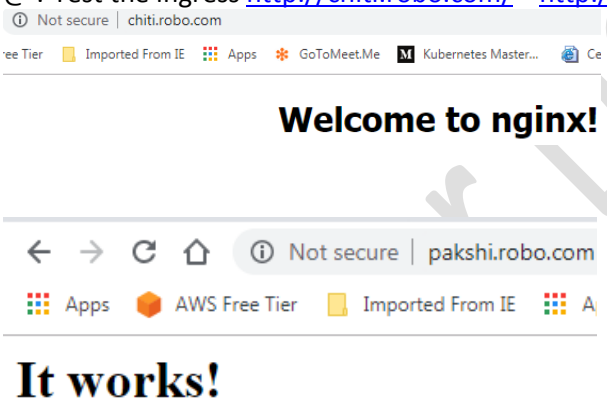
kubectl apply -f two-service.yml
kubectl get ingress

```
[root@ansikube manifest]# kubectl get ingress
NAME                      HOSTS                          ADDRESS          PORTS   AGE
name-virtual-host-ingress chiti.robo.com,pakshi.robo.com 34.102.213.223   80      79s
[root@ansikube manifest]#
```

:- Note:- Ingress will take time to create so you need wait and same can be verified on GCP as well.



@ :-Test the ingress http://chiti.robo.com/ http://pakshi.robo.com/



Note:- Before testing, ensure to have DNS entries on dns server or entries on local hosts file of your system.

<u>@Delete the ingress.</u>
kubectl delete -f two-service.yml
kubectl get ingress

<u>##Path based Ingress :- with same path multiple domain exposed.##</u>

<u>@Edit replicaset from 3 to 1</u>
kubectl get deploy

```
[root@ansikube manifest]# kubectl get deploy
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
apache-deployment    3/3     3            3           62m
nginx-deployment     3/3     3            3           96m
```

kubectl edit deploy apache-deployment
kubectl edit deploy nginx-deployment

kubectl get deploy

```
[root@ansikube manifest]# kubectl get deploy
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
apache-deployment    1/1     1            1           132m
nginx-deployment     1/1     1            1           166m
[root@ansikube manifest]#
```

kubectl get pods

```
[root@ansikube manifest]# kubectl get pods
NAME                                  READY   STATUS        RESTARTS   AGE
apache-deployment-548989b57d-c2hvt    0/1     Terminating   0          68m
apache-deployment-548989b57d-wgz99    1/1     Running       0          68m
apache-deployment-548989b57d-wrjv5    0/1     Terminating   0          68m
```

cat path-based-ingress.yml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: chiti.robo.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: nginx-service
          servicePort: 5000
      - path: /bar
        backend:
          serviceName: apache-service
          servicePort: 5001
```

kubectl apply -f path-based-ingress.yml
kubectl get ingress

<u>@Login to pod, run below commands</u>

#kubectl exec -it apache-deployment-548989b57d-wgz99 /bin/bash
mkdir /usr/local/apache2/htdocs /bar && echo -n "vasi...baaaaaaaaaaaaaaa" >/usr/local/apache2/htdocs/bar/index.html && ls -lrt |grep bar

kubectl exec -it nginx-deployment-69944b6c5b-9xv2h /bin/bash
mkdir /usr/share/nginx/html/foo && echo -n "pakshi.. uuffh uuffh uuffh" >/usr/share/nginx/html foo/index.html && ls -lrt |grep foo

<u>@Test path-based ingress@</u>
http://chiti.robo.com/bar/
http://chiti.robo.com/foo/

Note: - you may get this output “default backend – 404” because path needs to be configured on apps configuration file.