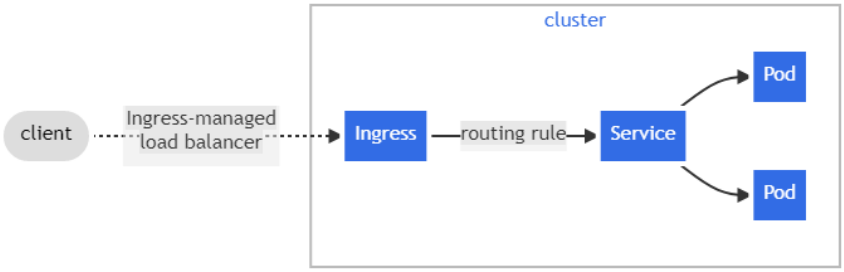
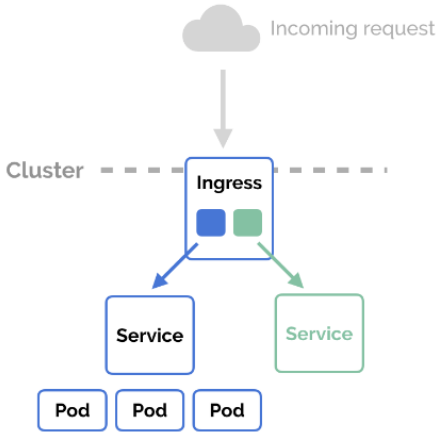
02Nov2022
Day**19**
Kubernetes Ingress
Monitoring Tools Prometheus, Grafana and Alert-manager Installation
---

Ingress
https://kubernetes.io/docs/concepts/services-networking/ingress/
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.



https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-ingress-guide-nginx-example.html
-In Kubernetes, an Ingress is an object that allows access to your Kubernetes services from outside the Kubernetes cluster.
-You configure access by creating a collection of rules that define which inbound connections reach which services.

Ingress



Prerequisites
You must have an Ingress controller to satisfy an Ingress. Only creating an Ingress resource has no effect.
Ingress Controllers
https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/
Kubernetes as a project supports and maintains AWS, GCE, and **nginx** ingress controllers
https://github.com/kubernetes/ingress-nginx/blob/main/README.md#readme
https://kubernetes.github.io/ingress-nginx/deploy/

prepare Minikube to enable Ingress Controller
-start minikube machine
-login to minikube host
minikube start --vm-driver=none --docker-env NO_PROXY=$NO_PROXY
minikube status
minikube addons list
| ingress | minikube | disabled | Kubernetes |
minikube addons enable ingress
minikube addons list
| ingress | minikube | enabled ✅ | Kubernetes |
verify
kubectl get ns
ingress-nginx Active 117s
kubectl get pods -A
kubectl get deployments.apps -A
kubectl get svc -A

Implement Ingress

kubectl create deployment nginx1 --image nginx
kubectl create deployment nginx2 --image nginx
watch kubectl get pod -o wide
nginx1-85b76cbcbb-l4sjs 1/1 Running 0 54s 172.17.0.4 minikube.example.com
nginx2-b648d744f-5jm7t 1/1 Running 0 51s 172.17.0.5 minikube.example.com
Now, need to change deployment's content
kubectl exec -it nginx1-85b76cbcbb-l4sjs -- /bin/bash
root@nginx1-85b76cbcbb-l4sjs:/# cat /usr/share/nginx/html/index.html
root@nginx1-85b76cbcbb-l4sjs:/# echo "nginx1" >/usr/share/nginx/html/index.html
root@nginx1-85b76cbcbb-l4sjs:/# curl localhost
nginx1
root@nginx1-85b76cbcbb-l4sjs:/# exit

root@minikube:~# kubectl exec -it nginx2-b648d744f-5jm7t -- /bin/bash
root@nginx2-b648d744f-5jm7t:/# cat /usr/share/nginx/html/index.html
root@nginx2-b648d744f-5jm7t:/# echo "nginx2" > /usr/share/nginx/html/index.html
root@nginx2-b648d744f-5jm7t:/# cat /usr/share/nginx/html/index.html
nginx2
root@nginx2-b648d744f-5jm7t:/# exit

kubectl expose deployment nginx1 --name nginx1-svc-ext --port 80 --protocol TCP --type NodePort
kubectl expose deployment nginx2 --name nginx2-svc-ext --port 80 --protocol TCP --type NodePort
verify
kubectl get svc
nginx1-svc-ext NodePort 10.97.9.35 <none> 80:31308/TCP 31s
nginx2-svc-ext NodePort 10.100.119.189 <none> 80:31701/TCP 18s
http://192.168.29.117:31308/
nginx1
http://192.168.29.117:31701/
nginx2

Ingress through IP

kubectl api-resources | grep -i "ingress"
ingresses ing networking.k8s.io/v1 true Ingress
vim ingress1.yaml
```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress1
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - http:
      paths:
      - path: /v1
        pathType: Prefix
        backend:
          service:
            name: nginx1-svc-ext
            port:
              number: 80
      - path: /v2
        pathType: Prefix
        backend:
          service:
            name: nginx2-svc-ext
            port:
              number: 80
```
:wq!
kubectl create -f ingress1.yaml --dry-run=client
kubectl create -f ingress1.yaml
kubectl get ingress
kubectl describe ingress ingress1
verify
http://192.168.29.117/v1
nginx1
http://192.168.29.117/v2
nginx2

Ingress through Name

hostname
minikube.example.com

define new name in to Linux localDNS file

vim /etc/hosts
192.168.29.117 **foo.bar.com** ->any name
:wq!
ping foo.bar.com
64 bytes from control-plane.minikube.internal (192.168.29.117): icmp_seq=1 ttl=64 time=0.033 ms
cp ingress1.yaml ingress2.yaml
vim ingress2.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress2
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - host: "foo.bar.com"                    ->appended line
    http:
      paths:
      - path: /v1
        pathType: Prefix
        backend:
          service:
            name: nginx1-svc-ext
            port:
              number: 80
      - path: /v2
        pathType: Prefix
        backend:
          service:
            name: nginx2-svc-ext
            port:
              number: 80
```

:wq!
kubectl apply -f ingress2.yaml --dry-run=client
kubectl apply -f ingress2.yaml
kubectl get ingress
kubectl describe ingress ingress2

verify through Linux

curl foo.bar.com/v1
nginx1
curl foo.bar.com/v2
nginx1

verify through Win

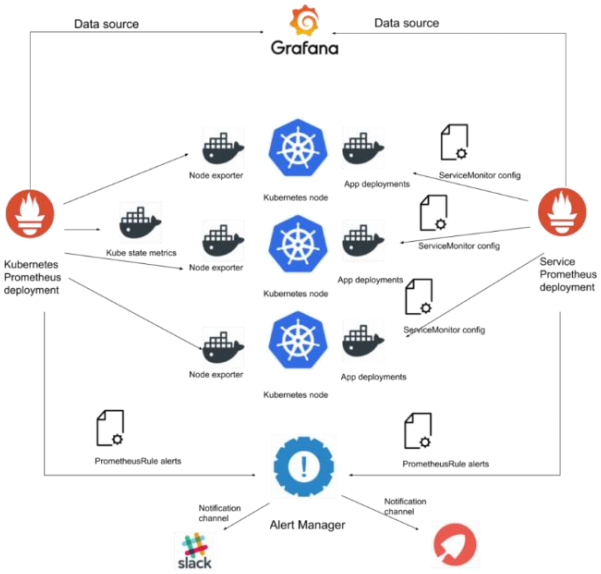
append localDNS record to 'c:\Windows\System32\Drivers\etc\hosts' file
press the Windows key.
Type Notepad in the search field.
In the search results, right-click Notepad and select Run as administrator
From Notepad, open the following file: c:\Windows\System32\Drivers\etc\hosts
192.168.29.117 foo.bar.com
save/quit
open browser
http://foo.bar.com/v1
nginx1
http://foo.bar.com/v2
nginx2

reference

https://docs.rackspace.com/support/how-to/modify-your-hosts-file/

Monitoring Tools Prometheus, Grafana and Alert-manager Installation

Prometheus is a monitoring solution for storing time series data like metrics. Grafana allows to visualize the data stored in Prometheus.
https://prometheus.io/
https://prometheus.io/docs/visualization/grafana/



Implement Prometheus, Grafana through Prometheus-Operator

https://computingforgeeks.com/setup-prometheus-and-grafana-on-kubernetes/

Step1.
yum install git -y
git clone https://github.com/prometheus-operator/kube-prometheus.git
cd kube-prometheus
ls

Step2. Create monitoring namespace, CustomResourceDefinitions & operator pod
kubectl create -f manifests/setup

Step3. Deploy Prometheus Monitoring Stack on Kubernetes
kubectl create -f manifests/
kubectl get pods -A
kubectl get svc -n monitoring

Step4. Access Prometheus, Grafana, and Alertmanager dashboards
hostname -i
192.168.0.191
kubectl --namespace monitoring port-forward svc/grafana --address 192.168.0.191 3000 &
kubectl --namespace monitoring port-forward svc/prometheus-k8s --address 192.168.0.191 9090 &
kubectl --namespace monitoring port-forward svc/alertmanager-main --address 192.168.0.191 9093 &
jobs
verify
open web browser
http://192.168.0.191:3000/
http://192.168.0.191:9090
http://192.168.0.191:9093

NOTE: Grafana default login credentials
admin
admin

reference
https://computingforgeeks.com/setup-prometheus-and-grafana-on-kubernetes/