Deployment manages the replicaset - > replicaset manages the pod - > pod is an abstraction of Container

Yaml file configuration:

1. Metadata : name of the component
2. Specification : every kind of activities we want to apply for this component
3. Template(blueprint for pods): Template comes under Specification - > template also has its metadata and specification . so we can say template is configuration inside configuration
4. Status - > automatic by k8s by the help of etcd database which keeps the information for k8s component

Q. How deployment and pod will get connected ?

Ans: in the same deployment.yaml file - > we have metadata for deployment which gives name and labels ( labels is used to matched the deployment and pod in deployment.yaml file) - > like wise we have specification and under specification we have template which defines the pod and its specification - > now under template we have Matchables - > which match the deployment’s label’s key-value pair in specification - > at the same time we can find labels and key value pair under template .

= > Pods get the label through the template blueprint

= > This label is matched by selector

Note : Metadata part contains labels = > and specification part contains selectors . label is sticks to Kubernetes component , in this case template’s metadata for pod and upper part metadata for deployment . so label’s under both component is same key value pair . meaning this deployment is attached with this pod . how deployment knows that pod is attached ? Ans: we will provide matchable under selectors under specification of deployment which confirms or connect pod to deployment .

##################Deployment sample###############

|  |  |  |
| --- | --- | --- |
| Throws error if we did not match the matchable with template labels | Deployment labeles can be different but selectors is responsible to connect pod with deploy(better to put same label for all) | Service.yaml - > now this service will connect with deployment and its pods by its selectors |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: nginx-deployment  labels:  app: nginx  spec:  replicas: 3  selector:  matchLabels:  app: nginx1  template:  metadata:  labels:  app: nginx  spec:  containers:  - name: nginx  image: nginx:1.14.2  ports:  - containerPort: 8080 | apiVersion: apps/v1  kind: Deployment  metadata:  name: nginx-deployment  labels:  app: nginx (this can be any name)  spec:  replicas: 3  selector:  matchLabels:  app: nginx1 (matched to template labels)  template:  metadata:  labels:  app: nginx1  spec:  containers:  - name: nginx  image: nginx:1.14.2  ports:  - containerPort: 8080 | apiVersion: apps/v1  kind: Service  metadata:  name: nginx-service  spec:  selector:  app: nginx1  ports:  -protocol: TCP  Port: 80 (this is for service itself)  targetPort: 8080 (this is for pod)  note: selectors should be matched with selectors of deployment)  if deployment selectors and service selectors does not match then service can not map to deployment and pod . And can not able to forward the external servie to pod) |

= > Complete Demo project

mongoDb and mongoExpress

how it works:

1st create mongoDb pod - > create Internal service (no external request) only component in same cluster can talk not out sider - >

2nd - > create Mongo express pod - > we need to have mongoDb credentials - . it can be done by Env variable (deployment.yaml file)

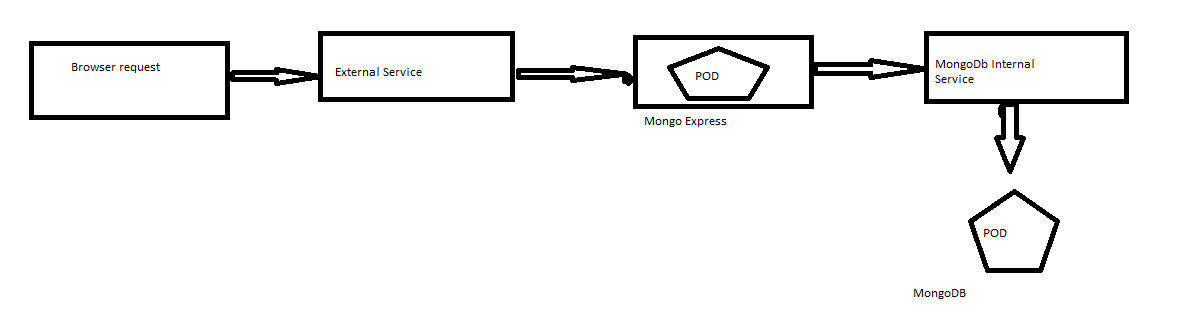
--- -- > we create ConfigMap - > it holds DB url

--- -🡪 We create Secret for credentials

----- > we will give reference of both ConfigMap and Secret to deployment.yaml file

Requirement: after doing this we need to have access mongodb express from web browser for this -- > we will create external Service which will allow web browser talk to mongo express - > url for this will be - > IP address of Node and Port of external Service

Flow :



Nodeport must be between : 30000 to 32767

|  |  |  |
| --- | --- | --- |
| configMap | Mongo-expression deployment | Mongo-expresson service ( connect to external) |
| apiVersion: v1  kind: ConfigMap  metadata:  name: mongodb-configmap  data:  database\_url: mongodb-service  NOTE: Database url is  Taken from mongodb  Service’s name | apiVersion: apps/v1  kind: Deployment  metadata:  name: mongo-express  labels:  app: mongo-express  spec:  replicas: 1  selector:  matchLabels:  app: mongo-express  template:  metadata:  labels:  app: mongo-express  spec:  containers:  - name: mongo-express  image: mongo-express  ports:  - containerPort: 8081  env:  - name: ME\_CONFIG\_MONGODB\_ADMINUSERNAME  valueFrom:  secretKeyRef:  name: mongodb-secret  key: mongo-root-username  - name: ME\_CONFIG\_MONGODB\_ADMINPASSWORD  valueFrom:  secretKeyRef:  name: mongodb-secret  key: mongo-root-password  - name: ME\_CONFIG\_MONGODB\_SERVER  valueFrom:  configMapKeyRef:  name: mongodb-configmap  key: database\_url  NOTE: Here mongo-expression will get connected to outsider by the help of it’s service at port 8082 with nodeport :3000. Then whatever request comes from outside to mongoexpress will be redirected to mongodb . Hence we have provided here password username and mongodb url . mongodb url is taken from configmap and username and password is taken from secret | apiVersion: v1  kind: Service  metadata:  name: mongo-express-service  spec:  selector:  app: mongo-express  type: LoadBalancer  ports:  - protocol: TCP  port: 8082  targetPort: 8081  nodePort: 30000 |

Flow : client - > mongo-expression-service - >mongo-expression-deployment- > (configmap,secret for url and credentials) - > mongodb-service - > mongodb-deployment

|  |  |  |
| --- | --- | --- |
| Mongodb-secret | Mongodb service | Mongodb-deployment |
| apiVersion: v1  kind: Secret  metadata:  name: mongodb-secret  type: opaque  data:  mongo-root-username: dXNlcm5hbWU=  mongo-root-password: cGFzc3dvcmQ= | apiVersion: v1  kind: Service  metadata:  name: mongodb-service  spec:  selector:  app: mongodb  ports:  - protocol: TCP  port: 27017  targetPort: 27017 | apiVersion: apps/v1  kind: Deployment  metadata:  name: mongodb-deployment  labels:  app: mongodb  spec:  replicas: 1  selector:  matchLabels:  app: mongodb  template:  metadata:  labels:  app: mongodb  spec:  containers:  - name: mongodb  image: mongo  ports:  - containerPort: 27017  env:  - name: MONGO\_INITDB\_ROOT\_USERNAME  valueFrom:  secretKeyRef:  name: mongodb-secret  key: mongo-root-username  - name: MONGO\_INITDB\_ROOT\_PASSWORD  valueFrom:  secretKeyRef:  name: mongodb-secret  key: mongo-root-password |

Note: apply all the file in kubernetes cluster - > and run command - > kubectl get service - > minikube service <Service with loadBalancer>

Q. What is namespace in k8s ?

Ans: namespace is like number of cluster(namespaces) in one cluster . Eg. like in one cluster we can have DEV, SIT,UAT,PROD name space which act differently to each other .

* - > Each namespace should have their own configMap to refer DB or other services .
* - > Each namespace should have their own secret they can not share between namespaces
* -> volume and node live globally , can not isolate them .

How to create namespace ?

Ans: 1st: we can create namespace mentioning in under configmap’s metadata like : namespce: my-namespce

Need to do more research

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INGRESS IN K8S ( Ingress and internal service configuration)

If we use external service to connect k8s pod or application then we will be using ip address and port of that service which is not good practice : <http://127.0.0.0:8080>

So to takeover this demerits, ingress comes into picture . ingress will handle the client side request and pass to service and service will pass to application . Ingress uses the DNS (domain name service)



|  |  |
| --- | --- |
| Ingress | Service |
| apiVersion: networking.k8s.io/v1  kind: Ingress  metadata:  name: myapp-ingress  spec:  rules:  - host: myapp.com  http:  paths:  - backend:  serviceName: mongo-express-service  servicePort: 8080 | apiVersion: v1  kind: Service  metadata:  name: mongo-express-service  spec:  selector:  app: mongo-express  type: LoadBalancer  ports:  - protocol: TCP  port: 8080  targetPort: 8081 |

Now here ingress controller will know which service is handling it’s forwarded request . meaning ingress receives the request from client and forward this request to internal service which is mongo-express-service running on port 8080 . now this service will mach the selectors (Mongo-express) with deployment and forward the same request to available pod running on port 8081 .

Note : with internal service we don’t need nodeport . can see in above yaml file where we have given nodeport as we are using that services as external service

######## what is ingress controller ? ##########

Ans: ingress controller evaluates all the rules of ingress yaml file , manages redirections . It acts like entrypoint to cluster

Flow : for bareware cluster

Client request - - > (outside cluseter: Proxy server - -> (inside cluster : Ingress controller -- > ingress -- > internal service 🡪 pod )

----- in cloud environment like AWS , google cloud etc ------flow will be like this

Cloud Load Balancer 🡪 (inside cluster : -> ingress controller - - > ingress 🡪 internal service - -> pod)

\*\*\*\*\*\*\*\*\*How to add or enables ingress controller ?\*\*\*\*\*\*\*\*\*\*\*\*\*

Commands:

minikube start (to start minikube we should have docker service running in our machine – so open docker desktop)

minikube addons enable ingress (command)

kubectl get pod -n kube-system ( we get pod by the name of nginx-ingress-controller)

kubectl get all -n Kubernetes-dashboard ( it will give all the information for namespace Kubernetes-dashboard)

note: we need to add ip address and DNS in /etc/hosts by the root user access:-

sudo vim /etc/hosts

---- why we nee this configuration in etc/hosts : external request comes to minikube cluster ip address and it checks the configuration ip address and here ip adderess get resolved with DNS .

--- we can also use tls or https in ingresss - > by using secret

######### HELM ########

Why we need or use in Helm ?

Helm provides the total structure of Kubernetes yaml file structure .

We need to install the helm in linux or windows :

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

**$** chmod 700 get\_helm.sh

**$** ./get\_helm.sh

helm create mychart -> it will create the yaml file structure

Ans: package manager for Kubernetes - > it is used to package yaml file and distribute them in public and private repositories

--- > example: let we have already deployed our service in k8s cluseter - > latter on we need to deploy Elastic Stack or any kind of database or monitoring apps like promethueus for logging -- > for this we need : statefulset , service , configmap , secret , k8s user with permission - -> if we make it by mannualy or searching in internet is tedious job -- > here : someone has already created thos yaml file and packaged it and store in repository which we can use and this bundle of yaml file is called Helm Chart .

What is templating engine ?

Ans:

Need to read more !

############### What is the persisting data and volume in k8s###### need to understand more still

K8s Admin or system engineer or devoops engineer sets up and maintain the cluster . Storage resource is provisioned by Admin . K8s User deploys application in cluster . So application developer explicitly mention pv in yaml file to use provisioned storage .

1, Persistent volume : it is like cluster resources RAM or CPU . Suppose application read or write file from direcotory ( session file or certificate etc) . it can be configured by yaml file ( kind: persistentVolume ). Persistent volume can be accessed by all namespace available in cluster . So PersistentVolume is globally access with in cluster regardless of namespace .

Local vs Remote Volume types: Local volume is not preferable because of its inconsistency . so use remote storage .

Question : who does this storage come from and who makes it available to cluster ?

Ans: this storage can be local disk , cloud storage , nfs server(network filse system) . we can use different type storage in Kubernetes cluster as storage . An application can use multiple storage at a time .

2, persistent volume claim 3, Storage class

Consider : we have application working with mysql . when mysql pod get restarted or died then it will lose it’s data . so if we have storage with mysql , whenever mysql will pod die the whole data can be found in storage and pod will pick it up from there where it was left or died .

###Storage Requirement#####

---- 🡪 Storage that does not depend on the pod lifecycle . But pod keeps dying and restarting .

------ --🡪 Storage must be available on all nodes , because one node’s storage does not support to other node .

------- --🡪 Storage needs to survive even if cluster crashes , because all the data which persisted before cluster crashes should be preserved and picked up when cluster or pod get started .