**Part -A**

**Ingress deploy part one :-steps**

**1)create a dir name data and enter into that dir by cd data**

**2)after entering in to the dir data here create metal-lb configuration..**

**3)configure the metal-lb by hown below file**

a) kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.13.9/config/manifests/metallb-native.yaml

kubectl -n metallb-system get all

cat <<EOF | kubectl create -f -

apiVersion: metallb.io/v1beta1

kind: L2Advertisement

metadata:

name: example

namespace: metallb-system

EOF

cat <<EOF | kubectl create -f -

apiVersion: metallb.io/v1beta1

kind: IPAddressPool

metadata:

name: first-pool

namespace: metallb-system

spec:

addresses:

- 192.168.1.100-192.168.1.110

EOF

### 4) Create the deployment - which has our server content.

### kubectl create deployment devops-wala --image=nginx --replicas=2 --port=80

### 5) Now, create a new service and assign the above deployment. we can use "kubectl expose" command.

### kubectl expose deployment devops-wala --name=srv-basic-devops-wala

### 6) Check the service. It should be clusterIP.

**kubectl get service**

### 7) Create a Nginx Controller

### kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.0/deploy/static/provider/cloud/deploy.yaml>

### 8) Check the pods, all should be in working condition

### kubectl -n ingress-nginx get deployments.apps

### 9) We should have a new service called ingress-nginx-controller after creating a NGINX controller.

**kubectl get service ingress-nginx-controller --namespace=ingress-nginx**

### 10) You can also check your external IP address, by using below command.

**INGRESS\_EXTERNAL\_IP=`kubectl get svc --namespace=ingress-nginx ingress-nginx-controller -o jsonpath='{.status.loadBalancer.ingress[0].ip}'`**

**echo $INGRESS\_EXTERNAL\_IP**

### 11) It's time to create a Ingress rule (Ingress Resource). Here, we are going to open the FQDN (devops-wala.com) and it should redirect to devops-wala deployment's pods.

#### Syntax==> kubectl create ingress INGRESS-RESOURCE-NAME --rule host/path=svcname:svcport[,tls[=secret]]

**kubectl create ingress devops-wala1 --class=nginx --rule devops-wala.com/=srv-basic-devops-wala:80**

### 12) We can also describe this ingress resources.

**kubectl describe ingress/devops-wala1**

### 13) Let's try to open the website. FQDN

**curl devops-wala.com**

### Part B:- Ingress type path based…

### 1) Create the deployment - which has our server content.

**kubectl create deployment devops-wala --image=nginx --replicas=2 --port=80**

**kubectl create deployment cka --image=nginx --replicas=2 --port=80**

**kubectl create deployment ckad --image=nginx --replicas=2 --port=80**

### 2) Now, create a new services and assign the above deployments. we can use "kubectl expose" command.

**kubectl expose deployment devops-wala --name=srv-path-devops-wala**

**kubectl expose deployment cka --name=srv-path-cka**

**kubectl expose deployment ckad --name=srv-path-ckad**

### 3) Check the services. It should be clusterIP.

**kubectl get service**

### 4) Check the pods, all should be in working condition.

**kubectl -n ingress-nginx get deployments.apps**

### 5) It's time to create an Ingress rule (Ingress Resource).

**cat << EOF | kubectl apply -f -**

**apiVersion: networking.k8s.io/v1**

**kind: Ingress**

**metadata:**

**name: devops-wala**

**annotations:**

**kubernetes.io/ingress.class: nginx**

**nginx.ingress.kubernetes.io/rewrite-target: / # For path base**

**spec:**

**rules:**

**- host: "devops-wala.com"**

**http:**

**paths:**

**- pathType: Prefix**

**path: "/"**

**backend:**

**service:**

**name: srv-path-devops-wala**

**port:**

**number: 80**

**- pathType: Prefix**

**path: "/cka"**

**backend:**

**service:**

**name: srv-path-cka**

**port:**

**number: 80**

**- pathType: Prefix**

**path: "/ckad"**

**backend:**

**service:**

**name: srv-path-ckad**

**port:**

**number: 80**

**EOF**

**kubectl describe ingress/devops-wala**

### 5) Try to open the webpage "FQDN"

**curl devops-wala.com**

**curl devops-wala.com/cka**

**curl devops-wala.com/ckad**

**6)** **FQDN** stands for Fully Qualified Domain Name. It represents the complete domain name for a specific host or server, including all levels of the domain hierarchy from the top-level domain (TLD) to the hostname. Here's an example of an FQDN:

Copy code

example.com

In this example:

* example is the hostname.
* .com is the top-level domain (TLD).

A fully qualified domain name includes the hostname and all its parent domain labels. For instance:

Copy code

www.example.com

In this case:

* www is the hostname.
* example.com is the parent domain.

**Why FQDNs are Used:**

* **Precision:** They provide an unambiguous way to specify a particular domain and its exact position in the DNS hierarchy.
* **Global Reach:** FQDNs are universally understood across the internet, ensuring consistency in naming and addressing.
* **Routing and Resolution:** They are crucial for DNS (Domain Name System) to correctly route requests and resolve to the appropriate IP addresses.

**In Kubernetes and Networking:**

* **Ingress:** When configuring Ingress in Kubernetes, you often specify an FQDN (like example.com) to route traffic based on hostname rules.
* **Service Discovery:** FQDNs help in identifying and accessing services within Kubernetes clusters and across networks.
* **Load Balancing:** FQDNs can be associated with load balancers to distribute traffic across multiple backend services or pods.

When setting up applications, services, or networking configurations, ensuring correct usage and configuration of FQDNs helps maintain consistency and reliability in communication across systems and networks.

**Part C:- Ingress type host based based :-**

### 1) Create the deployment - which has our server content.

**kubectl create deployment devops-wala --image=nginx --replicas=2 --port=80**

**kubectl create deployment example --image=nginx --replicas=2 --port=80**

### 2) Now, create a new services and assign the above deployments. we can use "kubectl expose" command.

**kubectl expose deployment devops-wala --name=srv-devops-wala**

**kubectl expose deployment example --name=srv-example**

### 3) Check the services. It should be clusterIP.

**kubectl get service**

### 4) Check the pods, all should be in working condition.

**kubectl -n ingress-nginx get deployments.apps**

### 5) I have no public IP @ thus, I am going to create a local DNS entry. Here, I will point my load balancer IP with FQDN.

**echo "$INGRESS\_EXTERNAL\_IP devops-wala.com www.example.com" >> /etc/hosts**

### 6) It's time to create an Ingress rule (Ingress Resource). We can also create Ingress resources through command line.

### a) kubectl create ingress different-url --class=nginx --rule devops-wala.com/=srv-devops-wala:80 --rule www.example.com/=srv-example:80 -o yaml --dry-run=client

ubuntu@ip-10-0-0-190:~$ kubectl create ingress different-url --class=nginx --rule devops-wala.com/=srv-devops-wala:80 --rule www.example.com/=srv-example:80 -o yaml --dry-run=client

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

creationTimestamp: null

name: different-url

spec:

ingressClassName: nginx

rules:

- host: devops-wala.com

http:

paths:

- backend:

service:

name: srv-devops-wala

port:

number: 80

path: /

pathType: Exact

- host: www.example.com

http:

paths:

- backend:

service:

name: srv-example

port:

number: 80

path: /

pathType: Exact

status:

loadBalancer: {}

### b) kubectl create ingress different-url --class=nginx --rule devops-wala.com/=srv-devops-wala:80 --rule [www.example.com/=srv-example:80](http://www.example.com/=srv-example:80) allready created

### kubectl describe ingress/different-url

In Kubernetes, the **--dry-run=client** flag is used with kubectl apply or kubectl create commands to simulate the resource creation or update without actually persisting the changes to the cluster. This feature is particularly useful for testing and verifying Kubernetes resource manifests before applying them for real. Here are the key reasons why --dry-run=client is used:

### 1. Simulation and Validation

* **Prevention of Errors:** By using **--dry-run=client**, you can catch potential errors or conflicts in your resource manifests before deploying them. This helps in identifying issues early in the development or deployment process.
* **Validation of Syntax:** It checks the syntax and structure of your YAML or JSON files against the Kubernetes API server without making any changes. This ensures that the manifests are correctly formatted and compliant with Kubernetes specifications.

### 2. Resource Impact Assessment

* **Impact Assessment:** Simulating resource creation or update helps you understand the impact on existing resources. It allows you to assess how the changes would affect other components in the cluster, such as pods, services, and deployments.
* **Avoidance of Unintended Changes:** It mitigates the risk of unintentional changes to critical resources by providing a safe way to preview modifications before applying them.

### 3. Operational Safety

* **Risk Reduction:** It minimizes the risk of disrupting production environments or causing downtime due to misconfigured or incompatible resource specifications.
* **Deployment Confidence:** Operators and developers can gain confidence in their deployments by testing configurations in a controlled, non-destructive manner.

### Example Usage:

bash

Copy code

kubectl apply -f deployment.yaml --dry-run=client

In this example, kubectl apply simulates applying the deployment.yaml file to the cluster without actually creating or updating any resources. It checks the feasibility of the operation and provides feedback on whether the resource would be applied successfully.

### Considerations:

* **Limitations:** While --dry-run=client is effective for syntax and basic validation, it doesn't validate against all cluster-specific conditions or dynamic changes that could occur during real deployments.
* **Subsequent Validation:** Even after successful simulation, it's advisable to perform thorough testing and validation in a staging environment before deploying changes to production.

Using --dry-run=client ensures safer and more reliable Kubernetes resource management by allowing operators to verify configurations and anticipate potential issues before committing changes to a live environment.

### 7) Try to open the webpage "FQDN"

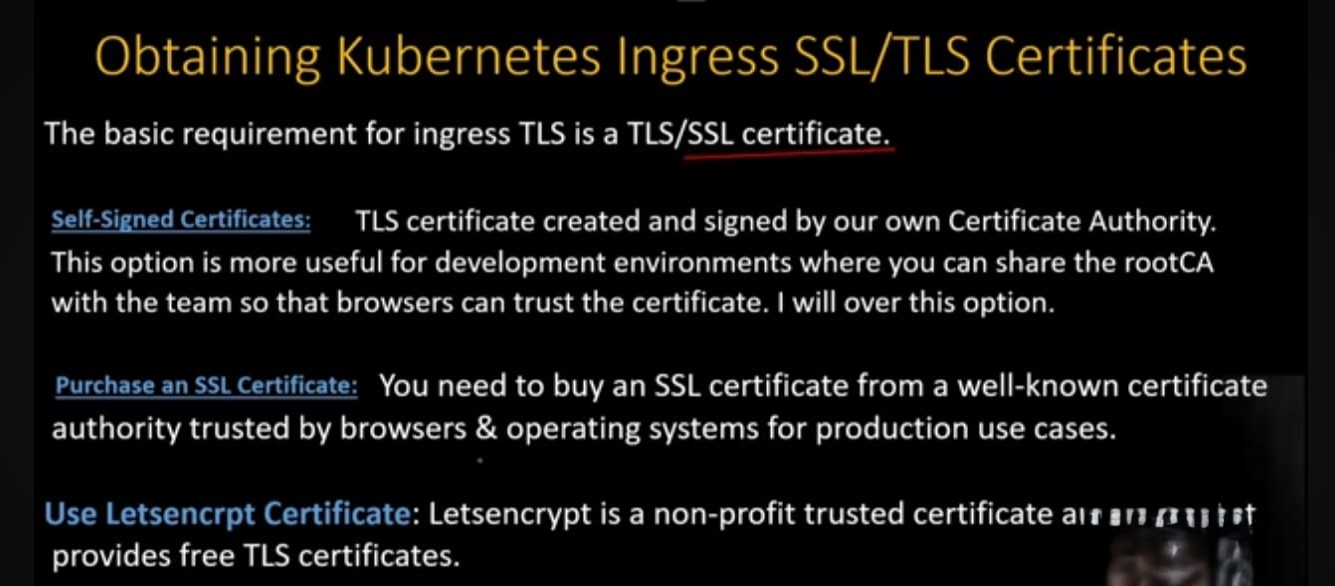
**curl devops-wala.com**

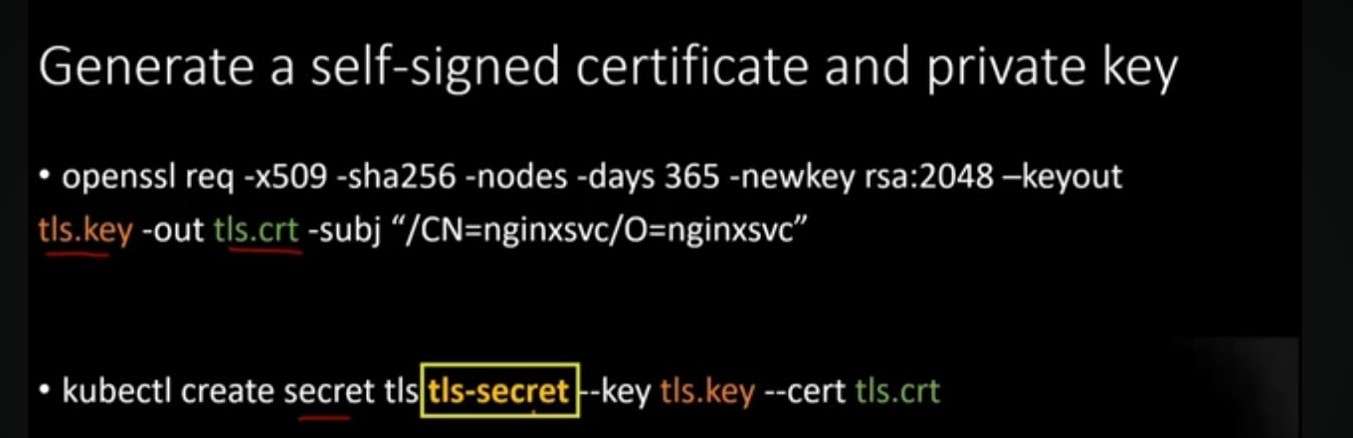
**curl www.example.com**

**Part D:- Ingress type SSL termination based :-**

### 1) In order to achive SSL, we need to create TLS key & Certificate and save as a form of Secret.

### Also, we need to generate the CA key & Certificate, Server Key & Certificate and Client key & Certificate.

****

****

**openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out tls.crt -subj "/CN=nginxsvc/O=nginxsvc"**

### 2) Create the secret and add the key and crt as key/value.

**kubectl create secret tls tls-secret --key tls.key --cert tls.crt**

### 3) Create the deployment - which has our server content.

**kubectl create deployment devops-wala --image=nginx --replicas=2 --port=80**

### 4) Now, create a new services and assign the above deployments. we can use "kubectl expose" command.

**kubectl expose deployment devops-wala --name=srv-devops-wala**

### 5)Check the services. It should be clusterIP.

**kubectl get service**

### 6) I have no public IP @ thus, I am going to create a local DNS entry. Here, I will point my load balancer IP with FQDN.

**echo "$INGRESS\_EXTERNAL\_IP devops-wala.com" >> /etc/hosts**

### 7) It's time to create an Ingress rule (Ingress Resource). We can also create Ingress resources through command line.

cat << EOF | kubectl apply -f -

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx-test-ssl

spec:

tls:

- hosts:

- devops-wala.com

secretName: tls-secret

ingressClassName: nginx

rules:

- host: devops-wala.com

http:

paths:

- path: /new-path

pathType: Prefix

backend:

service:

name: srv-devops-wala

port:

number: 80

EOF

**kubectl describe ingress/nginx-test-ssl**

### 8) Try to open the webpage "FQDN"

**curl https://devops-wala.com**

**curl -k -v https://devops-wala.com**