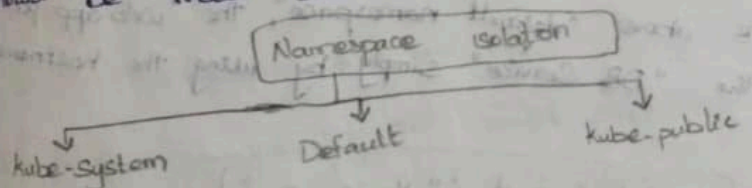
40. Namespaces :-

> So far, we've created pods, deployments, replicas & services inside a home, This home is called "namespace"

> we've created these pods, deployments, replicas & services inside a namespace called "default". This default namespace will be created by Kubernetes itself, when the cluster is first setup

> K8s creates its own pods & services for its internal purpose, such as those required by the n/w solution, the DNS service, etcetera. To isolate those from the user and to prevent you from accidentally deleting or modifying these services, K8s creates them under another namespace created at cluster startup named "kube-system"

> The third namespace created by K8s automatically is called "kube-public", This is where the resources that should be made available to all users are created



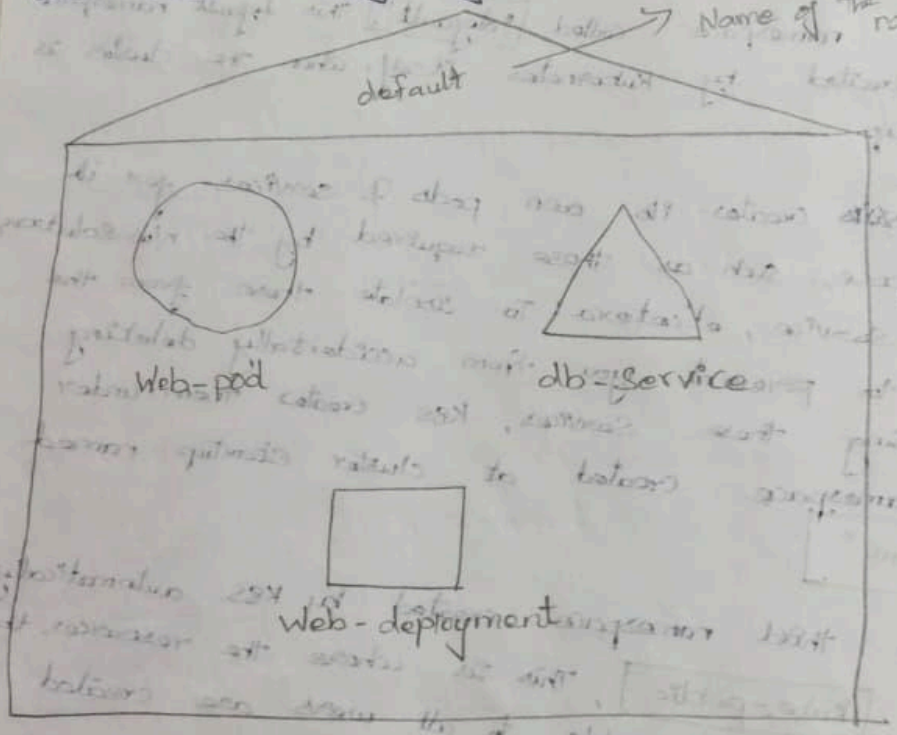
> We can create our own namespaces, if you wanted to use the same cluster for both "dev" & "prod" environment but at the same time, isolate the resources b/w them, you can create a different namespace for each of them

> Each of these namespaces {dev, prod, default} can have its own set of policies that define "who can do what", you can also assign "quota of resources" to each of these namespaces. That way each name space

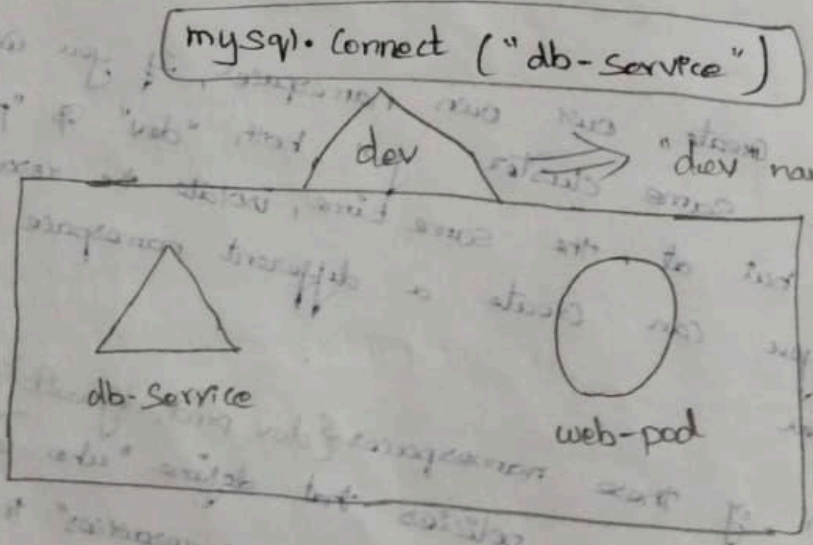
is guaranteed a certain amount and does not use more than its allowed limit

DNS

> The resources within the namespace can reach the each other simply by their names.



From the above "default namespace", the "web app pod" can reach the "DB Service" simply by using the hostname "DB service"



If required, the "web app pod" can reach a service in another namespace as well.

For this we must append the name of the namespace to the name of the service

For eg:- for the "web pod" in the "default namespace" to connect to the "database" in the "dev environment" or "namespace", we the syntax

servicename.namespace.svc.cluster.local format, that would be

```
mysql.connect("db-service.dev.svc.cluster.local");
```

You're able to do this because when the service is created, a DNS entry is added automatically in this format

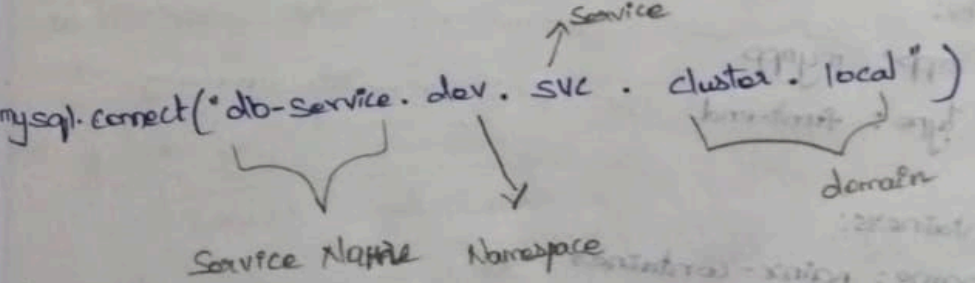
Looking closely at the DNS name of the service, the last part

cluster.local ⇒ This is the default domain name of the k8s cluster

svc ⇒ It is subdomain for service followed by

dev ⇒ Namespace & then the

db-service ⇒ name of the service itself



> Let us now look at some of the operational aspects of namespaces.

> Lets start with kube control commands

`kubectl get pods`

The above command lists the pods in default namespace

Then,

If you want to see/lists the pods in some other namespace, then use `--namespace = <name of the namespace>`

`> kubectl get pods --namespace = kube-system`

* By default, if we explicitly didn't mention the namespace, then the "resource" will get created in "default namespace" (Unless)

We can mention the namespace either in

* Manifest file, under "metadata"
* kubectl create/apply command

`kubectl create -f pod-definition.yml --namespace=dev`

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  namespace: dev
  labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

Create Namespace

- "namespace definition file" method.

```
apiVersion: v1
kind: Namespace
metadata:
    name: dev
```

```
>kubectl create -f namespace-dev.yml
namespace/dev created
```

Imperative command

```
>kubectl create (apply) namespace dev
namespace/dev created
```

Usually, by default, K8S objects will get created in "default" namespace. but what if we want to create "objects" in "dev" namespace. "PERMENANTLY", for which we will use "kubectl config" command to set to "namespace" in the current context to "dev"

```
>kubectl config set-context $(kubectl config current-context)
    --namespace = dev
```

Now, if we do,

```
kubectl get pods
```

The we'll see all the pods under "dev" namespace

Similarly we can switch to "prod" namespace the same way.

To view resources (pods) in all namespaces, we this cmd

```
kubectl get pods --all-namespaces
```

This will list all the pods in all of the namespaces

Understanding the kubectl config command.

```
>kubectl config set-context $(kubectl config current-context)
--namespace=dev
```

First, the command understands the current context & then sets the namespace to the desired one.

{ Contexts are used to manage multiple clusters & multiple environments from the same management system This is completely different topic, which we will discuss later }

Resource Quota

To limit resources in namespace, create a resource quota

To create resource quota we manifest file.

specify the namespace for which you want to create the quota. and specify the limits. under "spec" such as 10 pods, 10 cpu units, 10 GiB byte of memory, etc.,

Compute - quota . yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
    name: compute-quota
    namespace: dev  # namespace for which you want to create quota

spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: 5Gi
    limits.cpu: "10"
    limits.memory: 10Gi
```

kubectl ~~create~~ apply -f Compute - quota . yaml