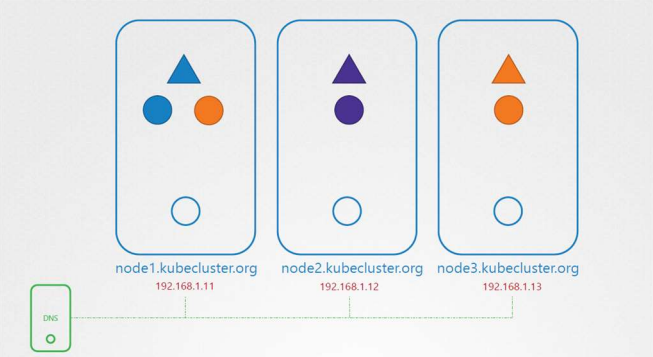
In this lecture, we will discuss DNS in the Kubernetes cluster. We will see what names are assigned to what objects, what are service DNS records, pod DNS records, what are the different ways you can reach one pod from another. And in the next lecture, we will see how Kubernetes implements DNS in the cluster.

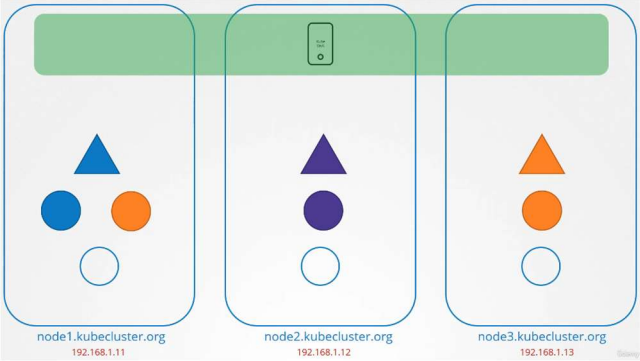
- ❑ What names are assigned to what objects?
- ❑ Service DNS records
- ❑ POD DNS Recrods

So we have a three-node Kubernetes cluster with some pods and services deployed on them. Each node has a node name and IP address assigned to it. The node names and IP addresses of the cluster are probably registered in a DNS server in your organization. Now, how that is managed, and who accesses them, are not of concern in this lecture. In this lecture, we discuss DNS resolution within the cluster, between the different components in the cluster, such as pods and services.



Kubernetes deploys a built-in DNS server by default when you set up a cluster. If you set up Kubernetes manually, then you do it by yourself. We will see how that is done and how it is configured in the next lecture. As far as this lecture is concerned, we will see how it helps pods resolve other pods and services within the cluster. So we don't really care about nodes. We focus purely on pods and services within the cluster.

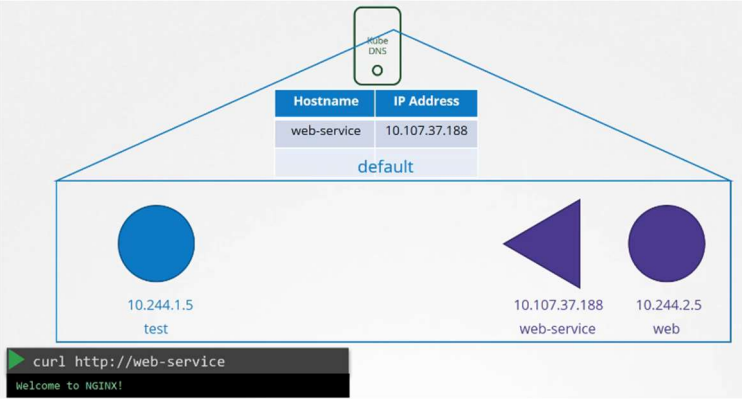
As long as our cluster networking is set up correctly, following the best practices we learn so far in this section, and all pods and services can get their own IP address and can reach each other, we should be good.



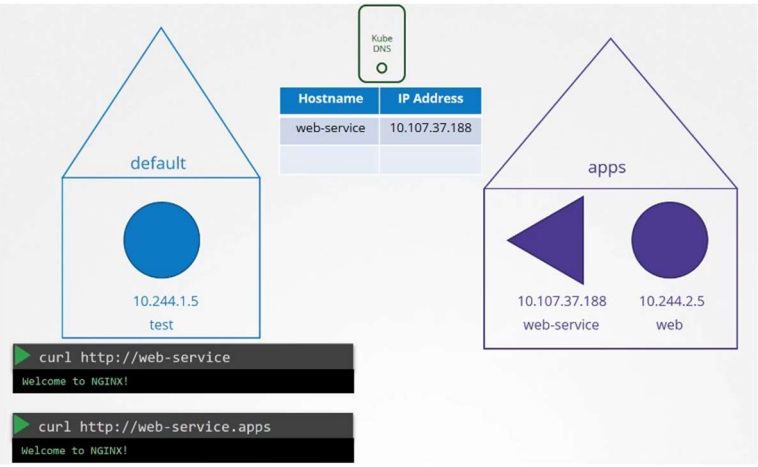
Let's start with just two pods and a service. I have a test pod on the left with the IP set to 10.2.44.105, and I have a web pod on the right with the IP set to 10.2.44.2.5. Looking at their IPs, you can guess that they're probably hosted on two different nodes, but that doesn't really matter.

As far as DNS is concerned, we assume that all pods and services can reach each other using their IP addresses. To make the web server accessible to the test pod, we create a service. We name it "web service." The service gets an IP 10.10737.188. Whenever a service is created, the Kubernetes DNS service creates a record for the service. It maps the service name to the IP address, so within the cluster, any pod can now reach this service using its service name.

Remember we talked about namespaces earlier, that everyone within the namespace addresses each other just with their first names, and to address anyone in another namespace, you use their full names. In this case, since the test pod and the web pod, and its associated service are all in the same namespace, the default namespace, you were able to simply reach the web service from the test pod using just the service name "**Web-Service**"



Let's assume the web service was in a separate namespace named "Apps." Then to refer to it from the default namespace, you would have to say "Web Service.apps." The last name of the service is now the name of the namespace. So here, "Web Service" is the name of the service, and "Apps" is the name of the namespace. For each namespace, the DNS server creates a subdomain.

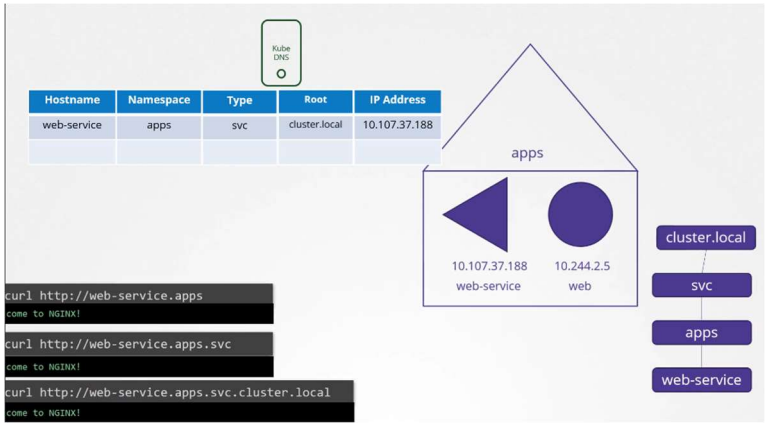


All the services are grouped together into another subdomain called "SVC." So what was that about? Let's take a closer look.

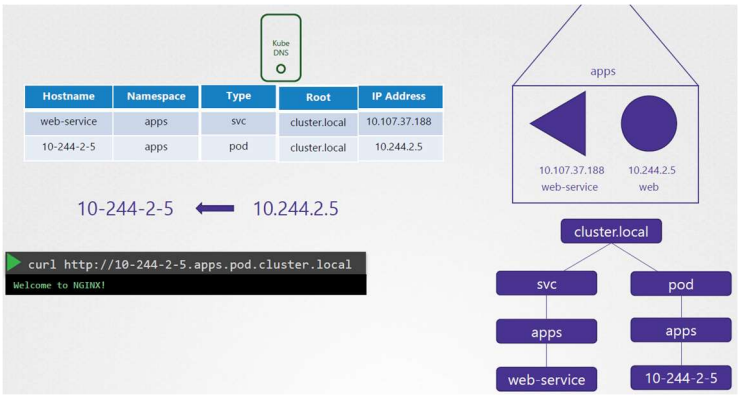
"Web service" is the name of the service, and "Apps" is the name of the namespace. For each namespace, the DNS server creates a subdomain with its name. All pods and services for a namespace are thus grouped together within a subdomain in the name of the namespace.

All the services are further grouped together into another subdomain called "SVC." So you can reach your application with the name "webservice.apps.svc."

Finally, all the services and pods are grouped together into a root domain for the cluster, which is set to "cluster.local" by default. So you can access the service using the URL "webservice.apps.svc.cluster.local," and that's the fully qualified domain name for the service. So that's how services are resolved within the cluster.



What about pods? Records for pods are not created by default, but we can enable that explicitly. We will see that in the next lecture. Once enabled, records are created for pods as well. It does not use the pod name though. For each pod, Kubernetes generates a name by replacing the dots in the IP address with dashes. The namespace remains the same, and the type is set to pod. The root domain is always "cluster.local."



Similarly, the test pod in the default name space gets a record in the DNS server with its IP converted to a dash host name, 10- 244-1-5 and namespace set to default. Type is pod and the route is cluster.local This results to the IP address of the pod.

Hostname	Namespace	Type	Root	IP Address
web-service	apps	svc	cluster.local	10.107.37.188
10-244-2-5	apps	pod	cluster.local	10.244.2.5
10-244-1-5	default	pod	cluster.local	10.244.1.5