**5. EKS Fargate - Mixed Mode Introduction**

--- In this lecture, we are going to understand about fargate profiles creation with yml.

--- let's see what we are going to learn.

**What are we going to learn?**

--- We are going to learn about writing Fargate Profiles using YAML wherein with YAML we can create multiple fargate profiles at a time.

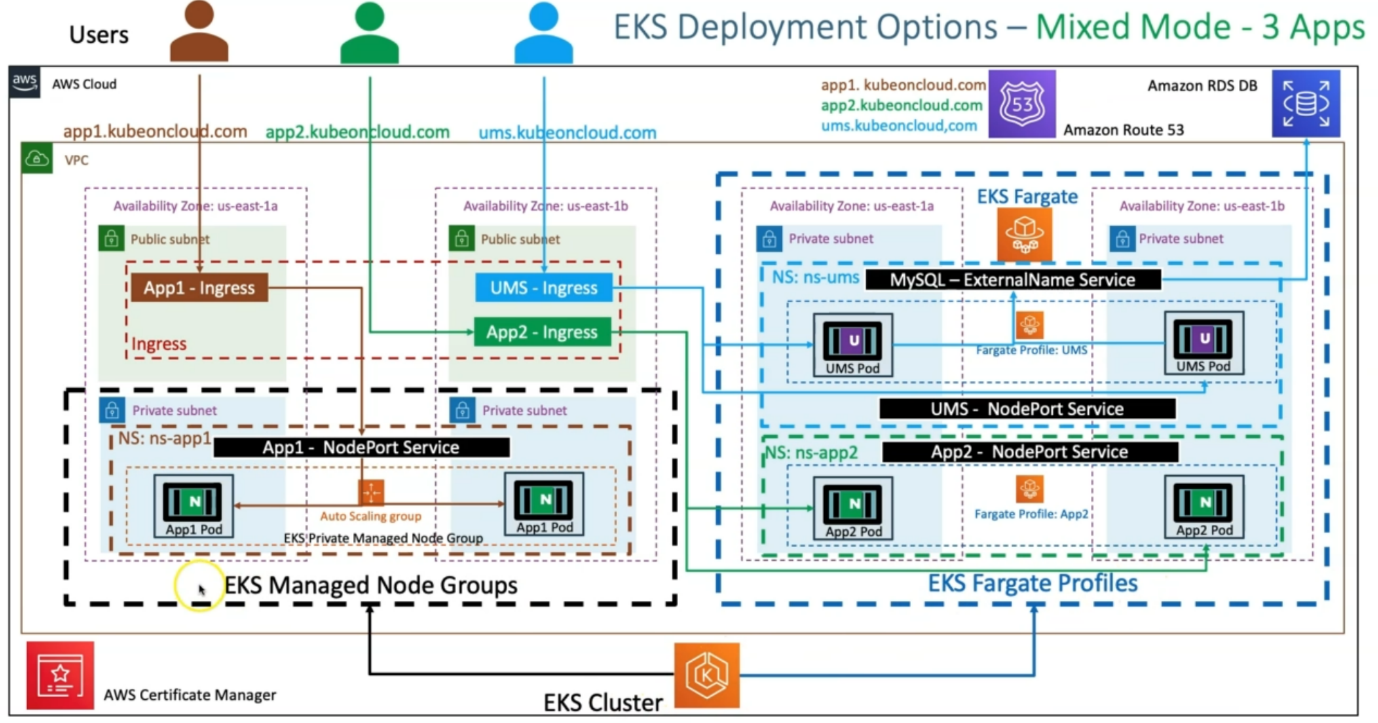
--- Understand about namespaces and labels in fargate profiles

--- Deploy 3 Apps in a mixed Mode

1. 2 Apps to 2 different Fargate Profiles
2. 1 App to EKS EC2 Manged Node Group

--- Test and clean-up

**Deploy 3 applications in visual manner**



--- in AWS cloud, we have VPC and inside VPC, we are going to have both EKS managed node groups and fargate profiles together. we call it as mixed mode.

--- what happens here is like. we have our available zones and then subnets. from fargate perspective, it only supports private subnets.

--- if you see here, here we have only private subnets and managed node groups can be created in both the private and public subnets.

--- as part of this section, we already have an existing private subnet related EKS managed node group. that's the reason I have rounded it to only this part and in this public subnet. we are going to deploy our ingress related services.

--- whenever we bring the fargate into the picture. The Organization of apps is going to be with the name Spaces. how, we're going to organize this with name spaces?

--- for this EKS managed node group, whatever the application going to deploy, for that we have created a namespace ns-aap1, in the same way for app2, we are going to create a namespace app2.

--- for application, ums, which is user management micro service. we are going to have the namespace as ns-ums.

--- this **ns-app2** will be part of one fargate profile and **ns-ums** will be part of another fargate profile and nsapp1 will be part of the managed node groups means like these named spaces are not

--- if we have that namespace type to fargate profile. So, those apps will be scheduled to that respective target profile.

--- app1 related pods I have deployed on the EC2 worker nodes and it will have its own NodePort Service and its own app1 ingress service.

--- in the same way for app2, we'll have a Fargate profile and inside that we will deploy our app2 application and it also have its own separate app2 ingress Service.

--- why we are having separate ingress services here. That we see in the next slide.

--- we have seen the concepts of context path-based routing in **ALB** But why we are not using here when we are using these mixed deployments.

--- there is a reason for that. we'll see in the next slide.

--- another thing is UMS; UMS is nothing but user management micro services. It's ingress service needs to be deployed and UMS deployment will be deployed and its node port service and its external name service and this external name service is going to connect to the RDS database.

--- if this Database is not up and running from previous sessions, whatever we have created. if it is down and then if you want to start it, you can start it up. If the db is not up then the ums pod will keep on failing.

--- one more thing will be there, which is external DNS to register the DNS names and also I associate SSL certificate manager related certificate will be associated in our ingress service.

--- it will resistor 3 DNS names.

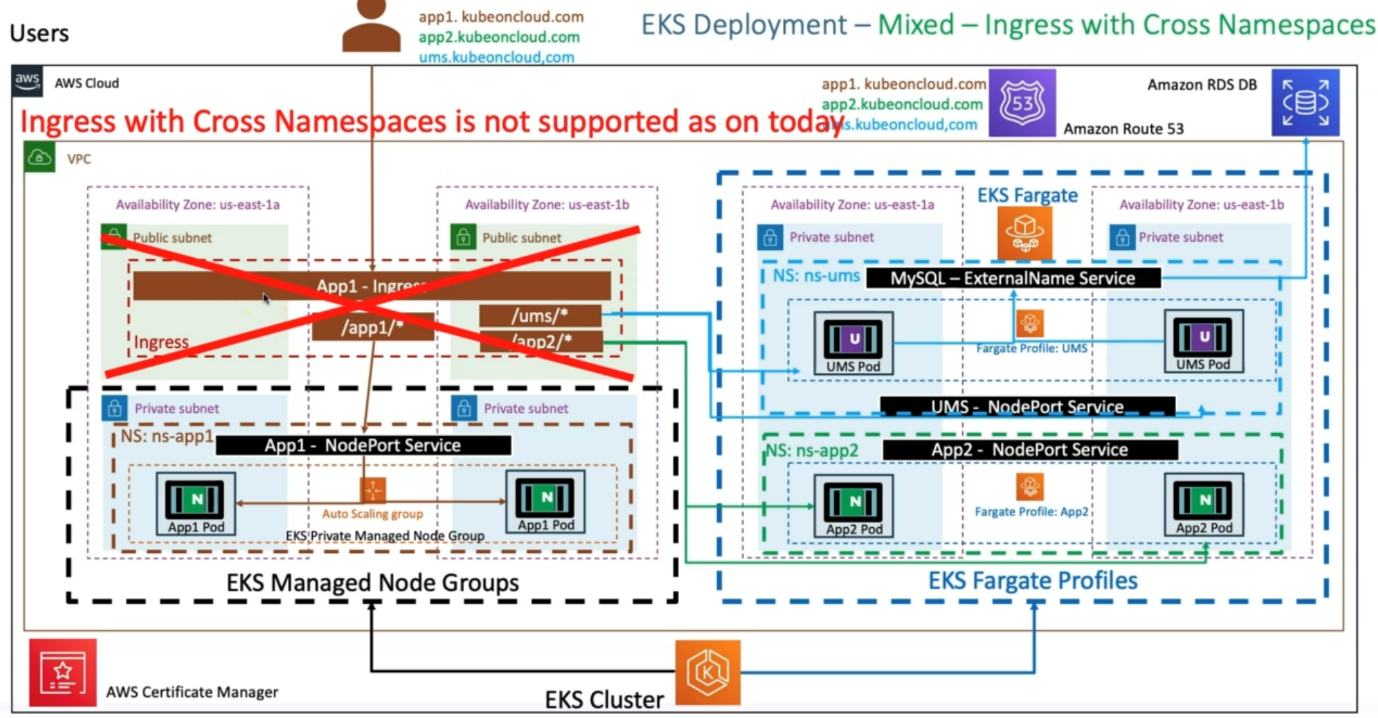
1. app1.kubeoncloud.com
2. app2.kubeoncloud.com
3. Ums.kubeoncloud.com

--- as a user. if I'm using **app1.kubeoncloud.com** then the request will come to app1 ingress and from there it will go to app1 nodeport service then from there it will be load balanced to pods.

--- if we go to **app2.kubeoncloud.com**, it will go to app2 ingress then the request will directly go to the pods because we are going to use fargate profile. For fargate profiles, we going to use target type **IP** in manifest.

--- in the same way for **ums.kubeoncloud.com**. we are going to access it and the request comes to **ums-ingress**, from there it will go to pods. then the request will go to external name service and then it will go to our database and serve it back to the user.

**EKS deployment – mixed – ingress with cross Namespace**



--- Why can't we use the app1-ingress for all these things. Single app1-ingress for /app1/\*, /ums/\* and /app2/\*.

--- if the request is coming to app1-ingress, if it /app1/\* then go this way, if it is /ums\* then go this way and if is /app2/\* then go this way.

--- So why can't we do that? it is nothing but ingress with cross namespace.

Because app1-ingress will be part of the ns-app2 name space and ums-ingress will be part of the

Ns-ums name space and same goes to other one.

--- **note** - ingress with cross name space is not supported today.