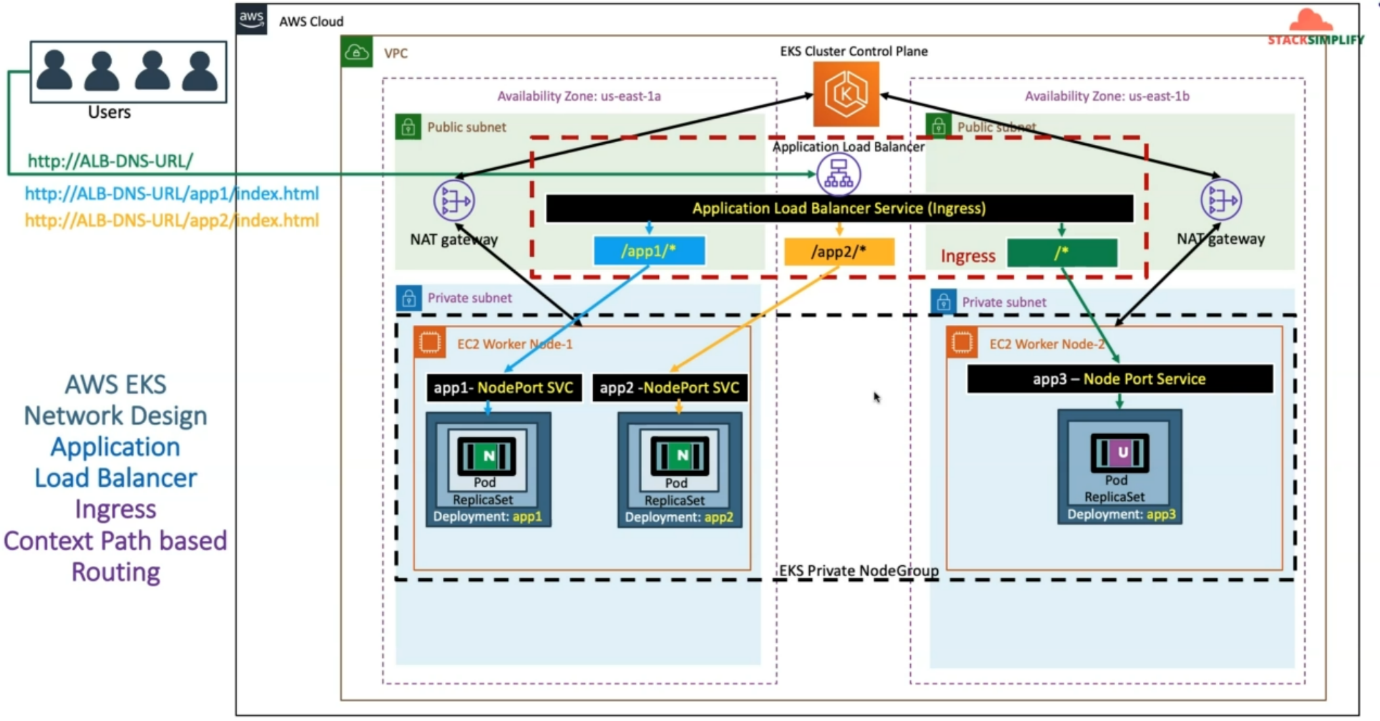
**1. Introduction to Ingress Context Path Based Routing**

--- **note** – in this lesion, we will implement kubernetes context-based routing in application load balancer.

**Network design for kubernetes context-based routing**



--- we have already created VPC and inside of VPC, we have created public and private subnet.

--- we already created a EKS cluster control plane and also EKS private node group in the private subnet. In addition to that, we also installed the ingress controller.

--- So now we are going to deploy three apps. So, for those three apps, we are going to have those three-deployment manifest and node port service manifest. So, those are nothing but **app1-Nodeport** service and **app1 deployment**, **app2-NodePort** service and **app2 deployment**, **app3-NodePort** service and **app3 deployment**.

--- we are also going to define ingress service with context-based routing. wherein we are going to define rules like /app1/\* goes to **app1-Nodeport** Service, which eventually goes to the app1 pod and /app2/\* goes to **app2-NodePort** service, which eventually goes to that app2 pod and anything with /\* context goes to app3 via **app3-NodePort** service.

--- So as a user, whenever I access my application with ALB DNS with root context(<http://ALB-DNS-URL/>). The request comes to the ingress application load balancer from there it will go to app3 pod via app3-NodePort service.

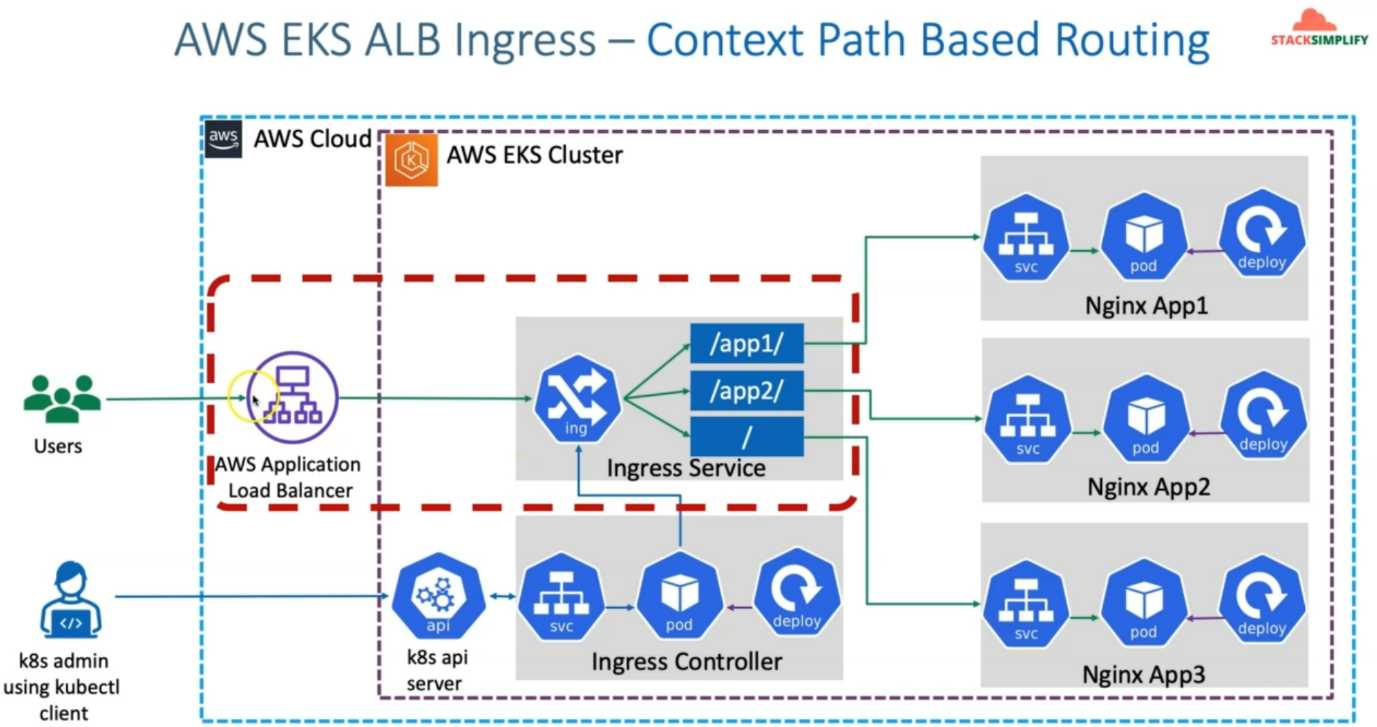
--- whenever I access my application with ALB DNS with /app1/index.html context(<http://ALB-DNS-URL/app1/index.html>). The request comes to the ingress application load balancer from there it will go to app1 pod via app1-NodePort service.

--- whenever I access my application with ALB DNS with /app1/index.html context(<http://ALB-DNS-URL/app2/index.html>). The request comes to the ingress application load balancer from there it will go to app2 pod via app2-NodePort service.

--- **note** – this is what happened, whenever we implemented context-based routing.

--- **note** - so far, we have seen the network design for ingress context path-based routing using the aws resources. also see the same architecture in a different way using the Kubernetes objects how it is going to work inside the Kubernetes cluster

**How the same architecture works in kubernetes cluster**



--- you have in AWS, you have created the AWS EKS cluster and in EKS cluster in the Kube system namespace, you have installed the Ingress controller. So, Ingress Controller related deployment, which eventually creates the Ingress Controller pods and also you have the ingress webhook service.

--- whenever as a Kubernetes admin using kubectl Client, whenever you deploy the Kubernetes objects. So, in those objects, whenever you deploy ingress service-related Kubernetes object or resource. this ingress controller keeps watching that k8s API server, which is EKS Cluster API server for finding any ingress services deployed and if it finds any ingress services deployed immediately, it is going to create the ingress service for us and this ingress service is nothing.

--- whenever the ingress service is created, it's equilent AWS application load balancer is created in the cloud and this ingress service already has context path-based routing rules inside that and those also will be defined inside this ingress service and equilent routing rules also will get created in the AWS application load balancer.

--- one important thing is, using kubectl client You'll also deploy the node port services and also deployments for your nginx **app1**, **app2** and **app3**.

--- whenever a user accesses **/app1/\*** context. the request will go nginx app1 via ingress service and it will cross-check the rules and it will go to that app1 service If the context path is **/app1/\*** and go to the node port service and from there it will go to the pod.

--- same applies for the **/app2/\*** and **/\*** context.

--- **note** - Another important thing which we need to be aware of is, AWS application load balancer and ingress service are not two different objects. in other lines, aws application load balancer is not proxying request to ingress service, both are comb inly a single object.

--- **note** – in EKS service, you call it as ingress service and in aws, we call it as application load balancer.