**2. Fargate Basics Introduction**

--- we are going to understand about fargate profiles basics, so far, we have learned about classic load balancers, network load balancers, application load balancers and then certificate manager, Route53, elastic blocks store and then RDS in combination with Kubernetes and with the elastic kubernetes service on aws.

--- now we are going to start with fargate profiles. it is going to be divided into multiple sections. we start with basics and then move on to advance sections of fargate profiles.

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/09-EKS-Workloads-on-Fargate/09-01-Fargate-Profile-Basic>

**What are we going to learn?**

--- Assumptions:

1. We already have EKS Cluster whose name is eksdemo1 created using eksctl
2. We already have a Managed Node Group with private networking enabled with two worker nodes

--- We are going to create a fargate profile using eksctl on our existing EKS Cluster eksdemo1

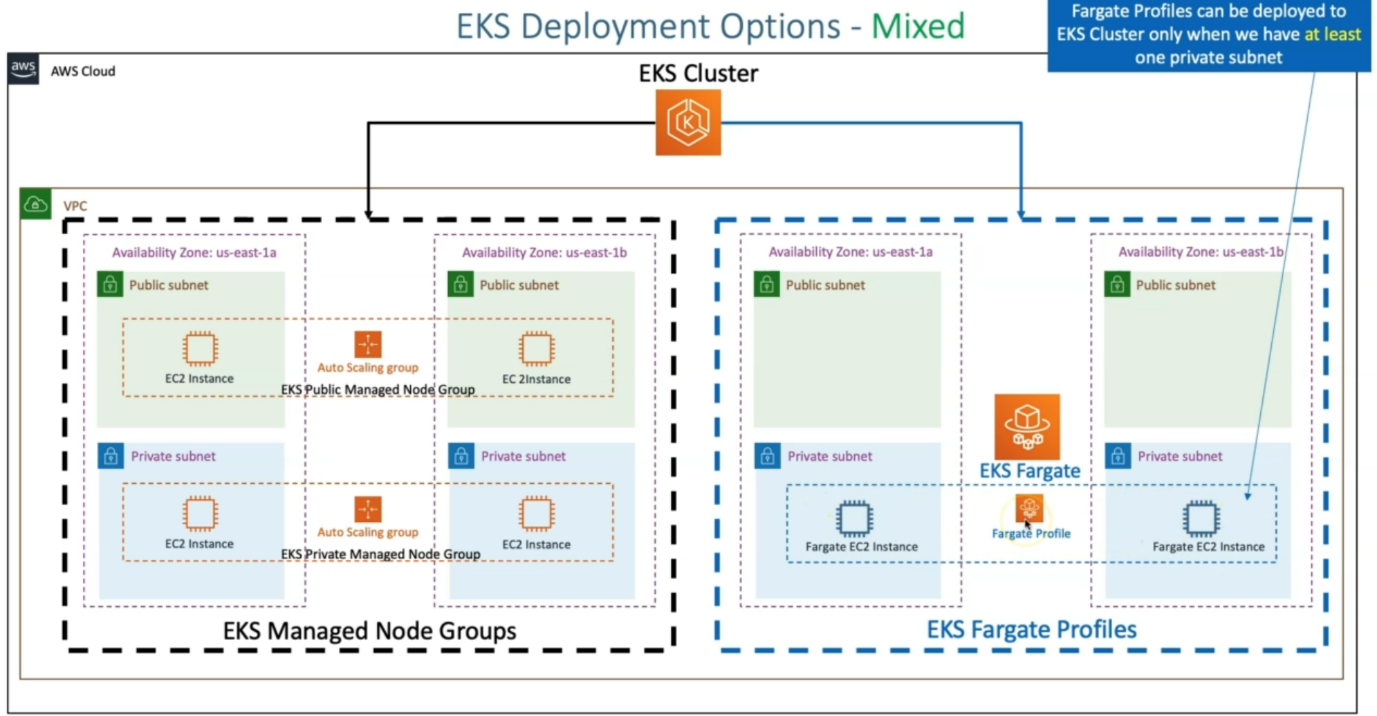
--- We are going to deploy a simple workload

1. Deployment: Nginx App 1
2. NodePort Service: Nginx App1
3. Ingress Service: Application Load Balancer

--- Ingress manifest going to have a additional annotation related to target-type: ip as these are going to be fargate workloads we are not going to have Dedicated EC2 Worker Node - Node Ports

**EKS deployment option – mixed**

--- we already discussed about mixing mode in our introductory section but we'll see the picture one more time.



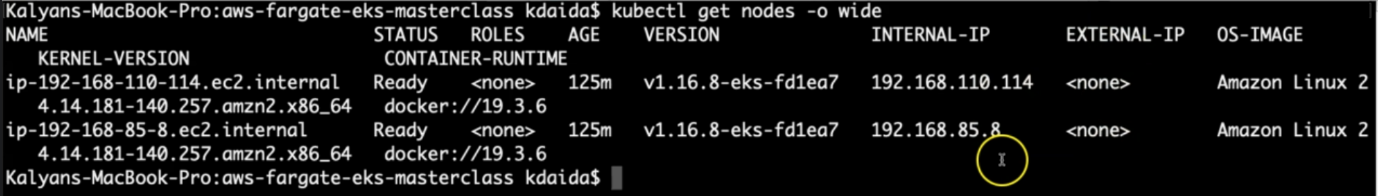
--- So, we have EKS manager managed Nodegroup with 2 instances in a private subnet currently. So, but the depiction here, we are also showing that the public subnet because manager node groups, which we can create on EC2 instances, we can create in both public and private subnet,

--- coming to forget case. we are going to specifically tell that. we can only create Fargate profiles in private subnets.

--- fargate profiles can be deployed to EKS cluster, in our EKS cluster, we have one private subnet. Fargate profiles cannot be deployed to the public subnet. this is very important point, which we should be aware of.

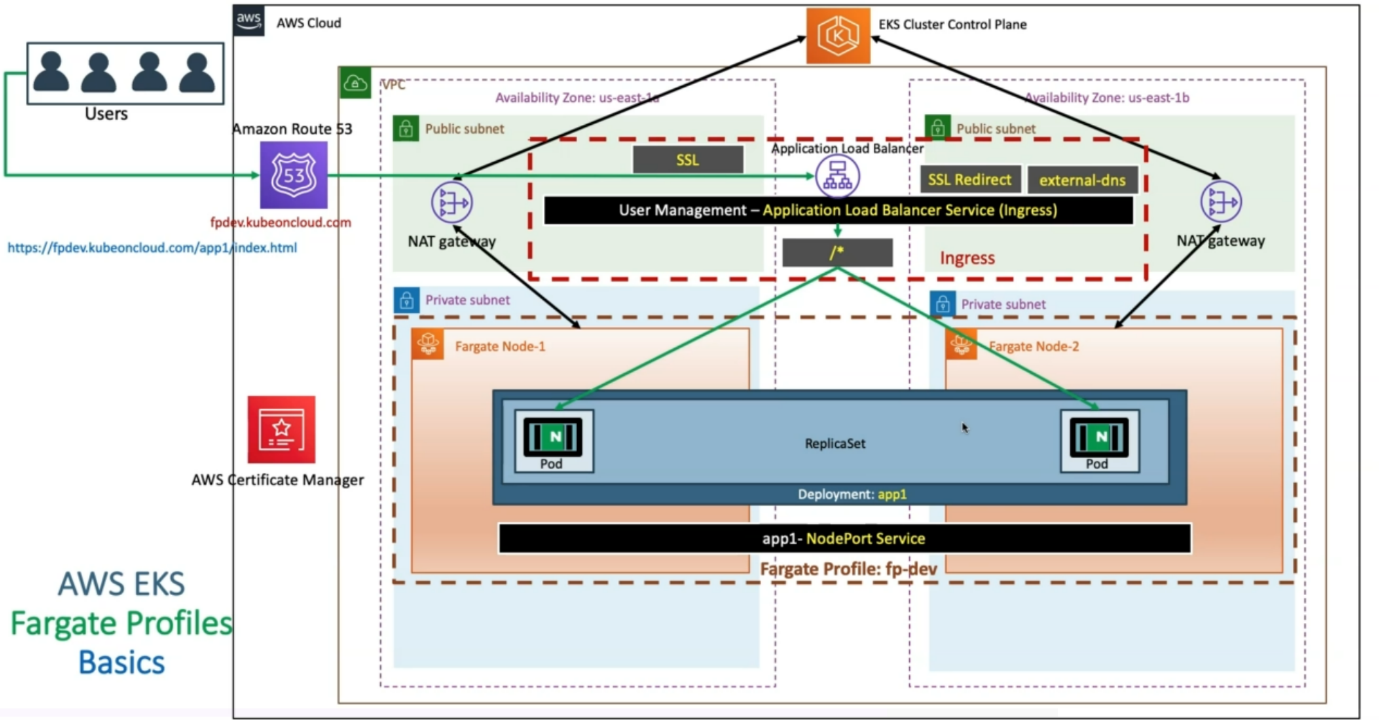
--- we already said that in private subnet, we already have a managed node groups running. let's go ahead and see that.

--- **kubectl get nodes -o wide** – to list node groups.



--- **note** - you can see that these nodes have internal ip but not external ip, so these nodes are in private subnet.

**EKS Fargate Profiles - Basics**



--- as part of the EKS cluster, already nat gateway and then public subnets and then private subnets are created in respected to availability zones. now we are going to create a fargate profile fp-dev with the namespace fp-dev.

--- going to deploy application named app1 nginx. we have deployed it and then it created a replicaset. once it starts creating the pods, those pods will be scheduled, by automatically creating the farmgate nodes.

--- fargate nodes will get created and then these pods get scheduled on those respective nodes.

--- we have given in over manifest the replica as 2. it is going to create two fargate nodes. here one thing we need to understands, here in Fargate the number of hosts is directly proportional to number of pods.

--- if you give 10 pods in your replicaset in your deployment. obviously, you are going to get create 10 fargate nodes.

--- We are going to automatically have an Outbound connection to that EKS cluster plane for this fargate nodes via nat gateway.

--- The next thing is, it's equal nodeport service will get created. So, when we are doing the complete mixed mode with multiple applications few on worker nodes and then few on fargate nodes.

--- the nodeport services will be very helpful. Where we are going to define target, type is **IP** as the annotation at the service level also when required.

--- now we are going to create the ingress service means, whatever the manifest we created, it is

going to create an ingress service with SSL redirect, external DNS, everything and then it will

also associate SSL certificate created here and DNS register **fpdev.kubeoncloud.com** for using external DNS and then whenever the user tries to access **fpdev.kubeoncloud.com/app1/index.html**

--- it comes to application load balancer and for the /\*, it is directly going to the pods. there is no node port service in between. So, from application load balancer target types, it is going directly to the respective pods.

--- Whenever we use the target type annotation with IP, by default it is going to have target type is equals is instance, which we really don't need to define because it is default.

--- very important – fargate does not have a nodeport service.

**Pre-requisites**

---- Pre-requisite Note about eksctl CLI

1. eksctl will have continuous releases with new feature additions to it. It’s always good to be on latest version of eksctl.
2. You can upgrade to latest version using below command on Mac
3. Currently highly evolving space (continuous features and new releases) from Kubernetes in AWS is eksctl and Fargate.
4. eksctl Releases URL: <https://github.com/weaveworks/eksctl/releases>
5. # Check version
6. eksctl version
7. # Update eksctl on mac
8. brew upgrade eksctl && brew link --overwrite eksctl
9. # Check version
10. eksctl version

**Pre-requisite check about ALB Ingress Controller & external-dns**

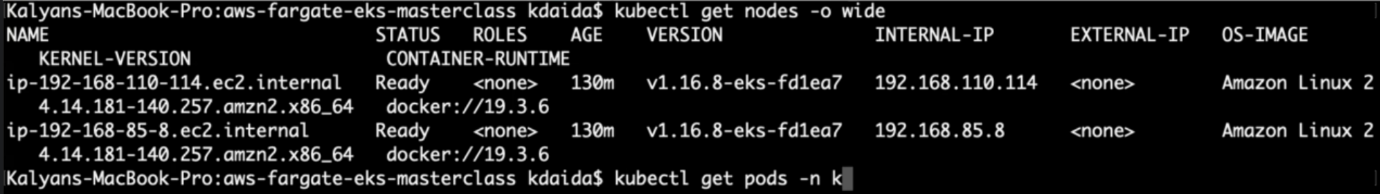
--- We need to have the below two listed components to be already running on our NodeGroup before deploying our application on fargate.

1. ALB Ingress Controller
2. External DNS

--- For our application, in addition to just deploying it we are going to access it via DNS register url fpdev.kubeoncloud.com

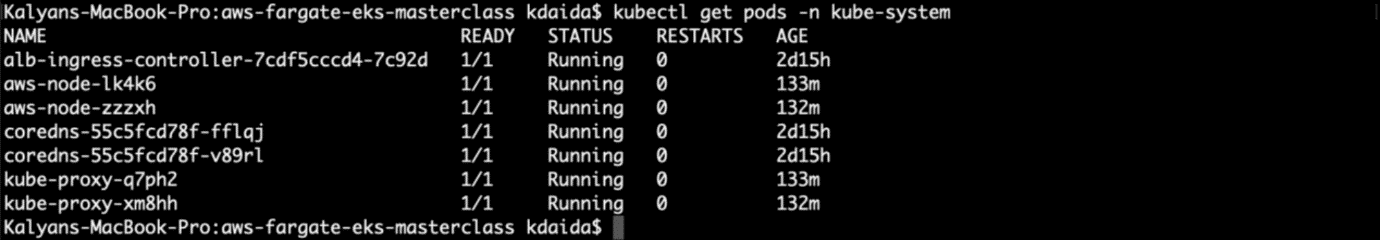
**# Get Current Worker Nodes in Kubernetes cluster**

--- **kubectl get nodes -o wide**



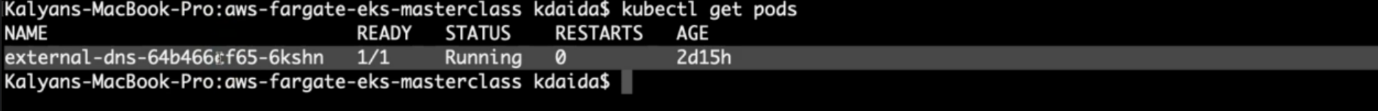
**# Verify Ingress Controller Pod running**

--- **kubectl get pods -n kube-system**



**# Verify external-dns Pod running**

--- **kubectl get pods**



--- our external pod is running.