**3. Create EKS Profile using eksctl & Review k8s manifests to be deployed**

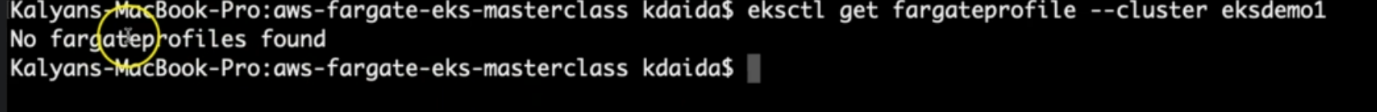
--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/09-EKS-Workloads-on-Fargate/09-01-Fargate-Profile-Basic>

**Create Fargate Profile on cluster eksdemo1**

--- Create Fargate Profile

**# Get list of Fargate Profiles in a cluster**

--- **eksctl get fargateprofile --cluster eksdemo1**



--- no fargate profiles are running inside of eksdemo1 cluster.

**# Template**

--- **eksctl create fargateprofile --cluster <cluster\_name> \**

**--name <fargate\_profile\_name> \**

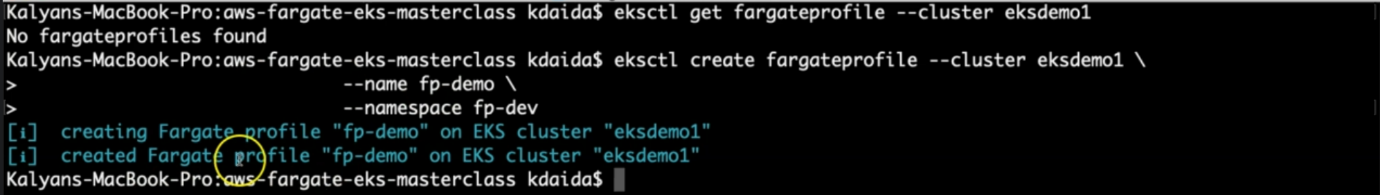
**--namespace <kubernetes\_namespace>**

**# Replace values**

--- **eksctl create fargateprofile --cluster eksdemo1 \**

**--name fp-demo \**

**--namespace fp-dev**



**# output (first deploying output)**

[ℹ]  Fargate pod execution role is missing, fixing cluster stack to add Fargate resources

[ℹ]  checking cluster stack for missing resources

[ℹ]  cluster stack is missing resources for Fargate

[ℹ]  adding missing resources to cluster stack

[ℹ]  re-building cluster stack "eksctl-eksdemo1-cluster"

[ℹ]  updating stack to add new resources [FargatePodExecutionRole] and outputs [FargatePodExecutionRoleARN]

[ℹ]  creating Fargate profile "fp-demo" on EKS cluster "eksdemo1"

[ℹ]  created Fargate profile "fp-demo" on EKS cluster "eksdemo1"

**Review NGINX App1 & Ingress Manifests**

--- We are going to deploy a simple NGINX App1 with Ingress Load Balancer

--- We cannot use Worker Node Node Ports for Fargate Pods for two reasons

1. Fargate Pods are created in Private Subnets, so no access to internet to access
2. Fargate Pods are created on random worker nodes whose information is unknown to us to use NodePort Service
3. But in our case, we are in mixed environment with Node Groups and Fargate, if we create a NodePort service, it will create the service with Node Group EC2 Worker Nodes Ports and it will work but when we delete those Node Groups, we will have an issue.
4. Always recommended to use alb.ingress.kubernetes.io/target-type: ip in ingress manifest for Fargate workloads

**Create Namespace Manifest**

--- This namespace manifest should match the one with we have created the Fargate Profile namespace value **fp-dev**.

--- **01-namespace.yml**

apiVersion: v1

kind: Namespace

metadata:

  name: fp-dev

--- **02-Nginx-App1-Deployment-and-NodePortService.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: app1-nginx-deployment

  labels:

    app: app1-nginx

  namespace: fp-dev

spec:

  replicas: 2

  selector:

    matchLabels:

      app: app1-nginx

  template:

    metadata:

      labels:

        app: app1-nginx

    spec:

      containers:

        - name: app1-nginx

          image: stacksimplify/kube-nginxapp1:1.0.0

          ports:

            - containerPort: 80

          resources:

            requests:

              memory: "128Mi"

              cpu: "500m"

            limits:

              memory: "500Mi"

              cpu: "1000m"

---

apiVersion: v1

kind: Service

metadata:

  name: app1-nginx-nodeport-service

  labels:

    app: app1-nginx

  namespace: fp-dev

  annotations:

#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-path: /app1/index.html

spec:

  type: NodePort

  selector:

    app: app1-nginx

  ports:

    - port: 80

      targetPort: 80

--- **03-ALB-Ingress-SSL-Redirect-with-ExternalDNS.yml**

# Annotations Reference: https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: app1-ingress-service

  namespace: fp-dev

  annotations:

    # Load Balancer Name

    alb.ingress.kubernetes.io/load-balancer-name: ingress-fargatedemo

    # Ingress Core Settings

    #kubernetes.io/ingress.class: "alb" (OLD INGRESS CLASS NOTATION - STILL WORKS BUT RECOMMENDED TO USE IngressClass Resource)

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: traffic-port

    #Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '2'

    ## SSL Settings

    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-1:180789647333:certificate/d86de939-8ffd-410f-adce-0ce1f5be6e0d

    #alb.ingress.kubernetes.io/ssl-policy: ELBSecurityPolicy-TLS-1-1-2017-01 #Optional (Picks default if not used)

    # SSL Redirect Setting

    alb.ingress.kubernetes.io/ssl-redirect: '443'

    # External DNS - For creating a Record Set in Route53

    external-dns.alpha.kubernetes.io/hostname: fpdev101.stacksimplify.com

    # For Fargate

    alb.ingress.kubernetes.io/target-type: ip

spec:

  rules:

    - http:

        paths:

          - path: /app1

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

# Important Note-1: In path based routing order is very important, if we are going to use  "/\*", try to use it at the end of all rules.

**Update All other manifests with namespace tag in metadata section**

  namespace: fp-dev

**Update All Deployment Manifests with Resources in Pod Template**

--- In Fargate, it is super highly recommended to provide the resources.requests, resources.limits about cpu and memory. Almost you can make it mandatory.

--- This will help Fargate to schedule a Fargate Host accordingly.

--- As fargate follows 1:1 ratio Host:Pod, one pod per host concept, we defining resources section in pod template (Deployment pod template spec) should be our mandatory option.

--- Even if we forget to define resources in our Deployment Pod Template, low memory using pods like NGINX will come up, high memory using Apps like Spring Boot REST APIs will keep restarting continuously due to unavailable resources.

          resources:

            requests:

              memory: "128Mi"

              cpu: "500m"

            limits:

              memory: "500Mi"

              cpu: "1000m"

**Update Ingress Manifest**

--- As we are running our pods on Fargate Serverless, we need to change our target-type to IP as there is no dedicated EC2 worker nodes concept in Fargate.

--- Important Note: When we are using same ingress in mixed mode deployments Node Groups & Fargate we can use this annotation at service level.

    # For Fargate

    alb.ingress.kubernetes.io/target-type: ip

**Also update the DNS Names**

    # External DNS - For creating a Record Set in Route53

    external-dns.alpha.kubernetes.io/hostname: fpdev.kubeoncloud.com