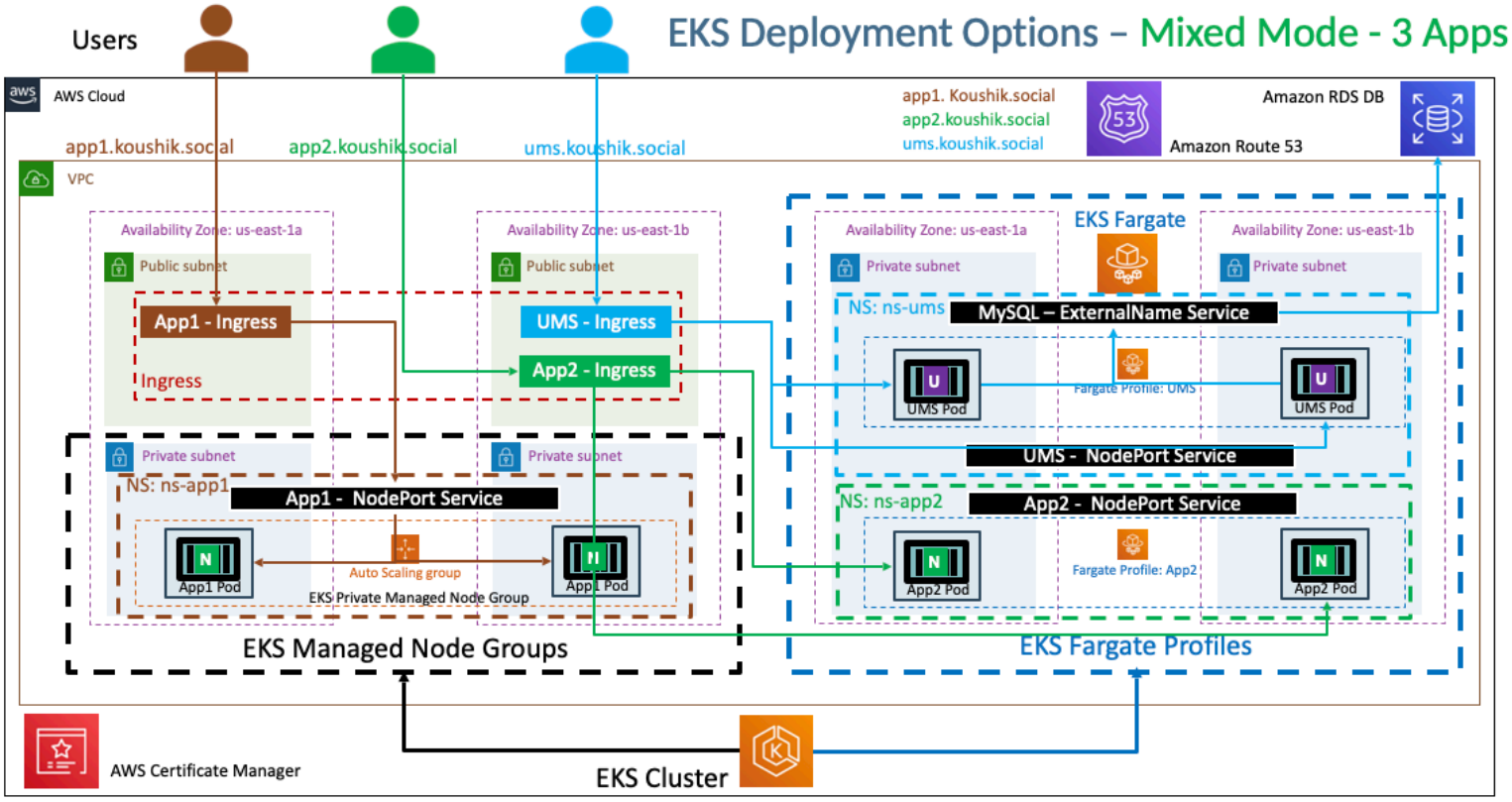
EKS Mixed Mode Deployment - 3 Apps

- Deploy 3 apps in mixed mode
- 2 Apps to 2 different Fargate Profiles
- 1 App to EKS EC2 Manged Node Group



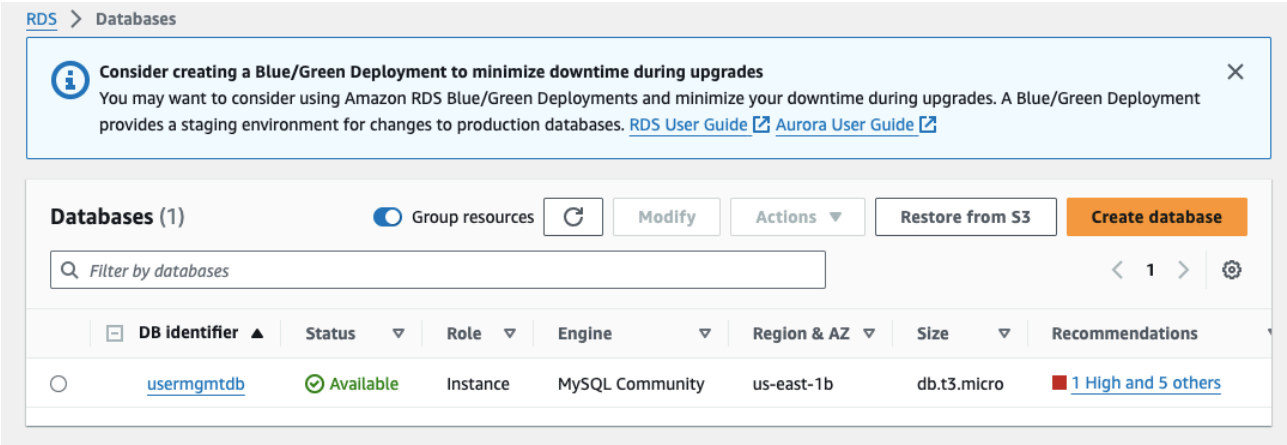
Steps:

- Create Advanced Fargate Profile with yml
- Create Fargate Profiles using YAML files
- Get list of Fargate profiles
- Review App1, App2 and UMS Manifests
- Deploy Apps
- Verify deployed Apps
- Verify using kubectl
- Verify ALB & Target Groups
- Access Applications

Pre req:

- Load-balancer controller should be installed and running in kube-system namespace
- External dns pod should be running in default namespace
- Rds database must be created and running in privater subnet for our usrmgmt microservice

```
NAMESPACE     NAME                                          READY   STATUS    RESTARTS
default       external-dns-6c4576979b-ttd9h                 1/1     Running   0
kube-system   aws-load-balancer-controller-5484f67599-h8t55 1/1     Running   0
kube-system   aws-load-balancer-controller-5484f67599-vrv7c 1/1     Running   0
```



Create Advanced Fargate Profile with yml

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: eksdemo1  # Name of the EKS Cluster
  region: us-east-1
fargateProfiles:
  - name: fp-app2
    selectors:
      # All workloads in the "ns-app2" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: ns-app2
  - name: fp-ums
    selectors:
      # All workloads in the "ns-ums" Kubernetes namespace matching the following
      # label selectors will be scheduled onto Fargate:
      - namespace: ns-ums
        labels:
          runon: fargate
```

Create Fargate Profiles using YAML files

eksctl create fargateprofile -f kube-manifests/01-Fargate-Advanced-Profiles/01-fargate-profiles.yml

Get list of Fargate profiles

List Fargate profiles
eksctl get fargateprofile --cluster eksdemo1

View in yaml format
eksctl get fargateprofile --cluster eksdemo1 -o yaml

Review App1, App2 and UMS Manifests

Check the namespace in yml manifest
ns-app1
ns-app2
ns-ums

```
 # For Fargate
  alb.ingress.kubernetes.io/target-type: ip
```

Deploy Apps

Verify if RDS DB which is required for UMS Service is UP and RUNNING.
Deploy Apps
kubectl apply -R -f kube-manifests/02-Applications/

Verify deployed Apps

Verify using kubectl
Verify Ingress
kubectl get ingress --all-namespaces

Verify Pods
kubectl get pods --all-namespaces -o wide

Verify Fargate Nodes
kubectl get nodes -o wide

Verify ALB & Target Groups

Verify ALB Listeneres, Rules
Verify Target Groups
App1: Should use Target Type as instance

App2, UMS: Should use Target Type as ip

Access Applications
App1: http://app1.koushik.social/app1/index.html
App2: http://app2.koushik.social/app2/index.html
UMS Health Status Page: http://ums.koushik.social/usermgmt/health-status
UMS List Users: http://ums.koushik.social/usermgmt/users

Kubemanifests:

```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: eksdemo1  # Name of the EKS Cluster
  region: us-east-1
fargateProfiles:
  - name: fp-app2
    selectors:
      # All workloads in the "ns-app2" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: ns-app2
  - name: fp-ums
    selectors:
      # All workloads in the "ns-ums" Kubernetes namespace matching the following
      # label selectors will be scheduled onto Fargate:
      - namespace: ns-ums
        labels:
          runon: fargate
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: ns-app1
# Apps deployed in this namespace will run on a EC2 Managed Node Group
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app1-nginx-deployment
  labels:
    app: app1-nginx
  namespace: ns-app1
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"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: app1-nginx-nodeport-service
  labels:
    app: app1-nginx
  namespace: ns-app1
  annotations:
#Important Note:  Need to add health check path annotations in service level if we are planning
    alb.ingress.kubernetes.io/healthcheck-path: /app1/index.html
spec:
  type: NodePort
  selector:
    app: app1-nginx
  ports:
    - port: 80
      targetPort: 80
```

```yaml
# 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
  labels:
    app: app1-nginx
  namespace: ns-app1
  annotations:
    # Load Balancer Name
    alb.ingress.kubernetes.io/load-balancer-name: app1-ingress
    # 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 bal
    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-1:388059815654:certificate/7294efbf-9752-427c-a8b0-0807f3ccd025
    #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: app1.koushik.social
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.
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: ns-app2
# Apps deployed in this namespace will run on a Fargate fp-app2
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app2-nginx-deployment
  labels:
    app: app2-nginx
  namespace: ns-app2
spec:
  replicas: 2
  selector:
    matchLabels:
      app: app2-nginx
  template:
    metadata:
      labels:
        app: app2-nginx
    spec:
      containers:
        - name: app2-nginx
          image: stacksimplify/kube-nginxapp2:1.0.0
          ports:
            - containerPort: 80
          resources:
            requests:
              memory: "128Mi"
              cpu: "500m"
            limits:
              memory: "500Mi"
              cpu: "1000m"
---
apiVersion: v1
kind: Service
metadata:
  name: app2-nginx-nodeport-service
  labels:
    app: app2-nginx
  namespace: ns-app2
  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: /app2/index.html
    # For Fargate
    alb.ingress.kubernetes.io/target-type: ip
spec:
  type: NodePort
  selector:
    app: app2-nginx
  ports:
    - port: 80
      targetPort: 80
```

```yaml
# Annotations Reference: https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app2-ingress-service
  labels:
    app: app2-nginx
  namespace: ns-app2
  annotations:
    # Load Balancer Name
    alb.ingress.kubernetes.io/load-balancer-name: app2-ingress
    # 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-1:388059815654:certificate/7294efbf-9752-427c-a8b0-0807f3ccd025
    #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: app2.koushik.social
    # For Fargate
    alb.ingress.kubernetes.io/target-type: ip
spec:
  rules:
    - http:
        paths:
          - path: /app2
            pathType: Prefix
            backend:
              service:
                name: app2-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.
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: ns-ums
# Apps deployed in this namespace will run on a Fargate fp-ums
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    runon: fargate
  namespace: ns-ums
spec:
  type: ExternalName
  externalName: usermgmtdb.cxojydmxwly6.us-east-1.rds.amazonaws.com
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: usermgmt-microservice
  labels:
    app: usermgmt-restapp
    runon: fargate
  namespace: ns-ums
spec:
  replicas: 2
  selector:
    matchLabels:
      app: usermgmt-restapp
  template:
    metadata:
      labels:
        app: usermgmt-restapp
        runon: fargate
    spec:
      initContainers:
        - name: init-db
          image: busybox:1.31
          command: ['sh', '-c', 'echo -e "Checking for the availability of MySQL Server deployment"; while ! nc -z mysql 3306; do sleep 1; printf "-"; done; echo -e "  >> MySQL DB Server ha
      containers:
        - name: usermgmt-restapp
          image: stacksimplify/kube-usermanagement-microservice:1.0.0
          resources:
            requests:
              memory: "128Mi"
              cpu: "500m"
            limits:
              memory: "500Mi"
              cpu: "1000m"
          ports:
            - containerPort: 8095
          env:
            - name: DB_HOSTNAME
              value: "mysql"
            - name: DB_PORT
              value: "3306"
            - name: DB_NAME
              value: "usermgmt"
            - name: DB_USERNAME
              value: "dbadmin"        # RDS DB Username is dbadmin
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-db-password
                  key: db-password
          livenessProbe:
            exec:
              command:
                - /bin/sh
                - -c
                - nc -z localhost 8095
            initialDelaySeconds: 60
            periodSeconds: 10
          readinessProbe:
            httpGet:
              path: /usermgmt/health-status
              port: 8095
            initialDelaySeconds: 60
            periodSeconds: 10
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-db-password
  labels:
    runon: fargate
  namespace: ns-ums
type: Opaque
data:
  db-password: ZGJwYXNzd29yZDEx
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: usermgmt-restapp-nodeport-service
  labels:
    app: usermgmt-restapp
    runon: fargate
  namespace: ns-ums
  annotations:
#Important Note:  Need to add health check path annotations in service level if we are planning to use m
    alb.ingress.kubernetes.io/healthcheck-path: /usermgmt/health-status
spec:
  type: NodePort
  selector:
    app: usermgmt-restapp
  ports:
    - port: 8095
      targetPort: 8095
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ums-ingress-service
  labels:
    app: usermgmt-restapp
    runon: fargate
  namespace: ns-ums
  annotations:
    # Load Balancer Name
    alb.ingress.kubernetes.io/load-balancer-name: ums-ingress
    # 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-1:388059815654:certificate/7294efbf-9752-427c-a8b0-0807f3ccd025
    #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: ums.koushik.social
    # For Fargate
    alb.ingress.kubernetes.io/target-type: ip
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: usermgmt-restapp-nodeport-service
                port:
                  number: 8095
# 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.
```

```
❯ eksctl create fargateprofile -f kube-manifests/01-Fargate-Advanced-Profiles/01-fargate-profiles.yml
2024-02-05 17:06:01 [ℹ]  creating Fargate profile "fp-app2" on EKS cluster "eksdemo1"
2024-02-05 17:06:35 [ℹ]  created Fargate profile "fp-app2" on EKS cluster "eksdemo1"
2024-02-05 17:06:35 [ℹ]  creating Fargate profile "fp-ums" on EKS cluster "eksdemo1"
2024-02-05 17:06:54 [ℹ]  created Fargate profile "fp-ums" on EKS cluster "eksdemo1"
❯ eksctl get fargateprofile --cluster eksdemo1 -o yaml
- name: fp-app2
  podExecutionRoleARN: arn:aws:iam::388059815654:role/eksctl-eksdemo1-fargate-FargatePodExecutionRole-BxqTutXXeVE2
  selectors:
  - namespace: ns-app2
  status: ACTIVE
  subnets:
  - subnet-0d43d09ff2b86cbd1
  - subnet-0fd74930dd5ace459
- name: fp-ums
  podExecutionRoleARN: arn:aws:iam::388059815654:role/eksctl-eksdemo1-fargate-FargatePodExecutionRole-BxqTutXXeVE2
  selectors:
  - labels:
      runon: fargate
    namespace: ns-ums
  status: ACTIVE
  subnets:
  - subnet-0d43d09ff2b86cbd1
  - subnet-0fd74930dd5ace459
```

```
❯ kubectl apply -R -f kube-manifests/02-Applications/
namespace/ns-app1 created
deployment.apps/app1-nginx-deployment created
service/app1-nginx-nodeport-service created
ingress.networking.k8s.io/app1-ingress-service created
namespace/ns-app2 created
deployment.apps/app2-nginx-deployment created
service/app2-nginx-nodeport-service created
ingress.networking.k8s.io/app2-ingress-service created
namespace/ns-ums created
service/mysql created
deployment.apps/usermgmt-microservice created
secret/mysql-db-password created
service/usermgmt-restapp-nodeport-service created
ingress.networking.k8s.io/ums-ingress-service created
```

```
ingress.networking.k8s.io/ums-ingress-service created
❯ kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                          READY   STATUS    RESTARTS   AGE   IP                NODE
              NOMINATED NODE   READINESS GATES
default       external-dns-6c4576979b-ttd9h                 1/1     Running   0          25h   192.168.127.250   ip-192-168-122-9.ec2.interna
l             <none>           <none>
kube-system   aws-load-balancer-controller-5484f67599-h8t55 1/1     Running   0          26h   192.168.66.135    ip-192-168-70-237.ec2.intern
al            <none>           <none>
kube-system   aws-load-balancer-controller-5484f67599-vrv7c 1/1     Running   0          26h   192.168.116.200   ip-192-168-122-9.ec2.interna
l             <none>           <none>
kube-system   aws-node-gkdj7                                2/2     Running   0          26h   192.168.122.9     ip-192-168-122-9.ec2.interna
l             <none>           <none>
kube-system   aws-node-wb5m9                                2/2     Running   0          26h   192.168.70.237    ip-192-168-70-237.ec2.intern
al            <none>           <none>
kube-system   coredns-d9b6d6c7d-llvs4                       1/1     Running   0          28h   192.168.89.55     ip-192-168-70-237.ec2.intern
al            <none>           <none>
kube-system   coredns-d9b6d6c7d-tnkpl                       1/1     Running   0          28h   192.168.88.127    ip-192-168-70-237.ec2.intern
al            <none>           <none>
kube-system   kube-proxy-f8x8k                              1/1     Running   0          26h   192.168.70.237    ip-192-168-70-237.ec2.intern
al            <none>           <none>
kube-system   kube-proxy-l8bc2                              1/1     Running   0          26h   192.168.122.9     ip-192-168-122-9.ec2.interna
l             <none>           <none>
ns-app1       app1-nginx-deployment-859d7bb997-9c4kb        1/1     Running   0          19m   192.168.101.30    ip-192-168-122-9.ec2.interna
l             <none>           <none>
ns-app1       app1-nginx-deployment-859d7bb997-qkqg5        1/1     Running   0          19m   192.168.72.247    ip-192-168-70-237.ec2.intern
al            <none>           <none>
ns-app2       app2-nginx-deployment-76959c4978-cx96t        1/1     Running   0          19m   192.168.115.193   fargate-ip-192-168-115-193.e
c2.internal   <none>           <none>
ns-app2       app2-nginx-deployment-76959c4978-d956t        1/1     Running   0          19m   192.168.64.150    fargate-ip-192-168-64-150.ec
2.internal    <none>           <none>
ns-ums        usermgmt-microservice-6cb448c6fc-8jx9z        1/1     Running   0          19m   192.168.71.41     fargate-ip-192-168-71-41.ec2
.internal     <none>           <none>
ns-ums        usermgmt-microservice-6cb448c6fc-jp27k        1/1     Running   0          19m   192.168.81.219    fargate-ip-192-168-81-219.ec
2.internal    <none>           <none>
~/De/aw/09/09-02-Fargate-Profiles-Advanced-YAML                                                                  ✓ 05:41:57 PM
```

us-east-1.console.aws.amazon.com/route53/v2/hostedzones?region=us-east-1#ListRecordSets/Z08912351BGJ4P6FDSSWT

Route 53

- Dashboard
- Hosted zones
- Health checks
- IP-based routing
  - CIDR collections
- Traffic flow
  - Traffic policies
  - Policy records
- Domains
  - Registered domains
  - Requests
- Resolver
  - VPCs
  - Inbound endpoints
  - Outbound endpoints
  - Rules
  - Query logging
  - Outposts

Records (9) | DNSSEC signing | Hosted zone tags (0)

Records (9) Info

Delete record | Import zone file | Create record

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Filter records by property or value | Type | Routing pol... | Alias

Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...
koushik.social	NS	Simple	-	No	ns-202.awsdns-25.com. ns-1148.awsdns-15.org. ns-883.awsdns-46.net. ns-1678.awsdns-17.co.uk.	172800
koushik.social	SOA	Simple	-	No	ns-202.awsdns-25.com. awsd...	900
_e2afe2d179285a1e...	CNAME	Simple	-	No	_713b387e38248df65ba60e...	300
app1.koushik.social	A	Simple	-	Yes	app1-ingress-735500530.us-...	-
app1.koushik.social	TXT	Simple	-	No	"heritage=external-dns,exter...	300
app2.koushik.social	A	Simple	-	Yes	app2-ingress-1769739653.u...	-
app2.koushik.social	TXT	Simple	-	No	"heritage=external-dns,exter...	300
ums.koushik.social	A	Simple	-	Yes	ums-ingress-2068345435.us...	-
ums.koushik.social	TXT	Simple	-	No	"heritage=external-dns,exter...	300

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:v=3;$case=tags:false%5C,client:false;$regex=tags:false%5C,cli...

EC2 > Load balancers

Load balancers (3)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Actions | Create load balancer

Filter load balancers

Name	DNS name	State	VPC ID	Availability Z
ums-ingress	ums-ingress-2068345435.us-east-1.elb.amazona...	Active	vpc-07b3347a51c776...	2 Availability
app2-ingress	app2-ingress-1769739653.us-east-1.elb.amazona...	Active	vpc-07b3347a51c776...	2 Availability
app1-ingress	app1-ingress-735500530.us-east-1.elb.amazonaw...	Active	vpc-07b3347a51c776...	2 Availability

0 load balancers selected

Select a load balancer above.

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:

Target groups (1/3) Info

Actions | Create target group

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balance
k8s-nsapp1-app1ngin-e999ad8b07	arn:aws:elasticloadbalanci...	32189	HTTP	Instance	app1-ingress
k8s-nsapp2-app2ngin-98650fa389	arn:aws:elasticloadbalanci...	80	HTTP	IP	app2-ingress
k8s-nsums-usermgmt-8b7b1a7b27	arn:aws:elasticloadbalanci...	8095	HTTP	IP	ums-ingress

Target group: k8s-nsapp1-app1ngin-e999ad8b07

Registered targets (2) Info

Anomaly mitigation: Not applicable | Deregister | Register targets

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

Instance ID	Name	Port	Zone	Health status	Health status details	Ano
i-070b8461f33a4a7f3	eksdemo1-eks...	32189	us-east-1b	Healthy	-	
i-019508107720ba9aa	eksdemo1-eks...	32189	us-east-1a	Healthy	-	

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:

EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security

Target groups (1/3) Info

Actions | Create target group

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balance
k8s-nsapp1-app1ngin-e999ad8b07	arn:aws:elasticloadbalanci...	32189	HTTP	Instance	app1-ingress
k8s-nsapp2-app2ngin-98650fa389	arn:aws:elasticloadbalanci...	80	HTTP	IP	app2-ingress
k8s-nsums-usermgmt-8b7b1a7b27	arn:aws:elasticloadbalanci...	8095	HTTP	IP	ums-ingress

Target group: k8s-nsapp2-app2ngin-98650fa389

Registered targets (2) Info

Anomaly mitigation: Not applicable | Deregister | Register targets

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

IP address	Port	Zone	Health status	Health status details	Anomaly detection result
192.168.64.150	80	us-east-1a	Healthy	-	Normal
192.168.115.193	80	us-east-1b	Healthy	-	Normal

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:

EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security

Target groups (1/3) Info

Actions | Create target group

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balance
k8s-nsapp1-app1ngin-e999ad8b07	arn:aws:elasticloadbalanci...	32189	HTTP	Instance	app1-ingress
k8s-nsapp2-app2ngin-98650fa389	arn:aws:elasticloadbalanci...	80	HTTP	IP	app2-ingress
k8s-nsums-usermgmt-8b7b1a7b27	arn:aws:elasticloadbalanci...	8095	HTTP	IP	ums-ingress

Target group: k8s-nsums-usermgmt-8b7b1a7b27

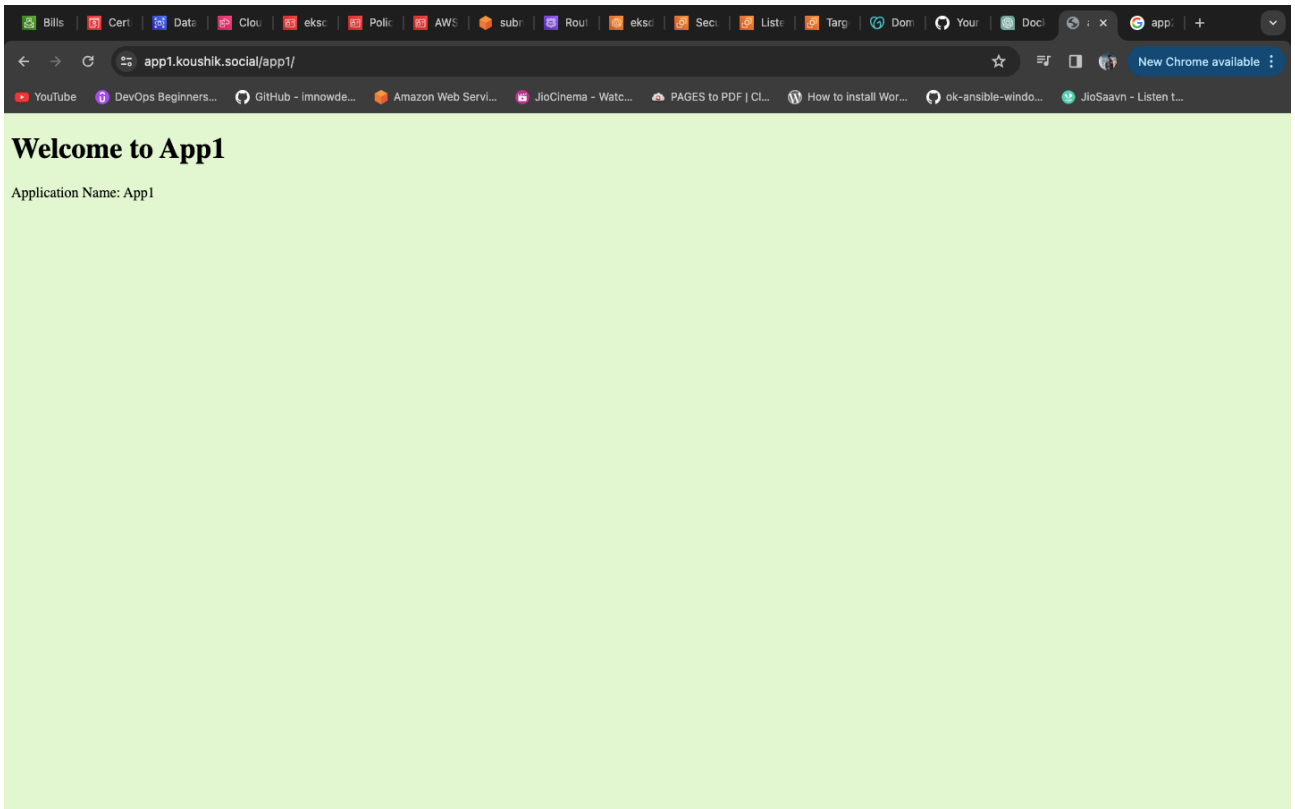
Registered targets (2) Info

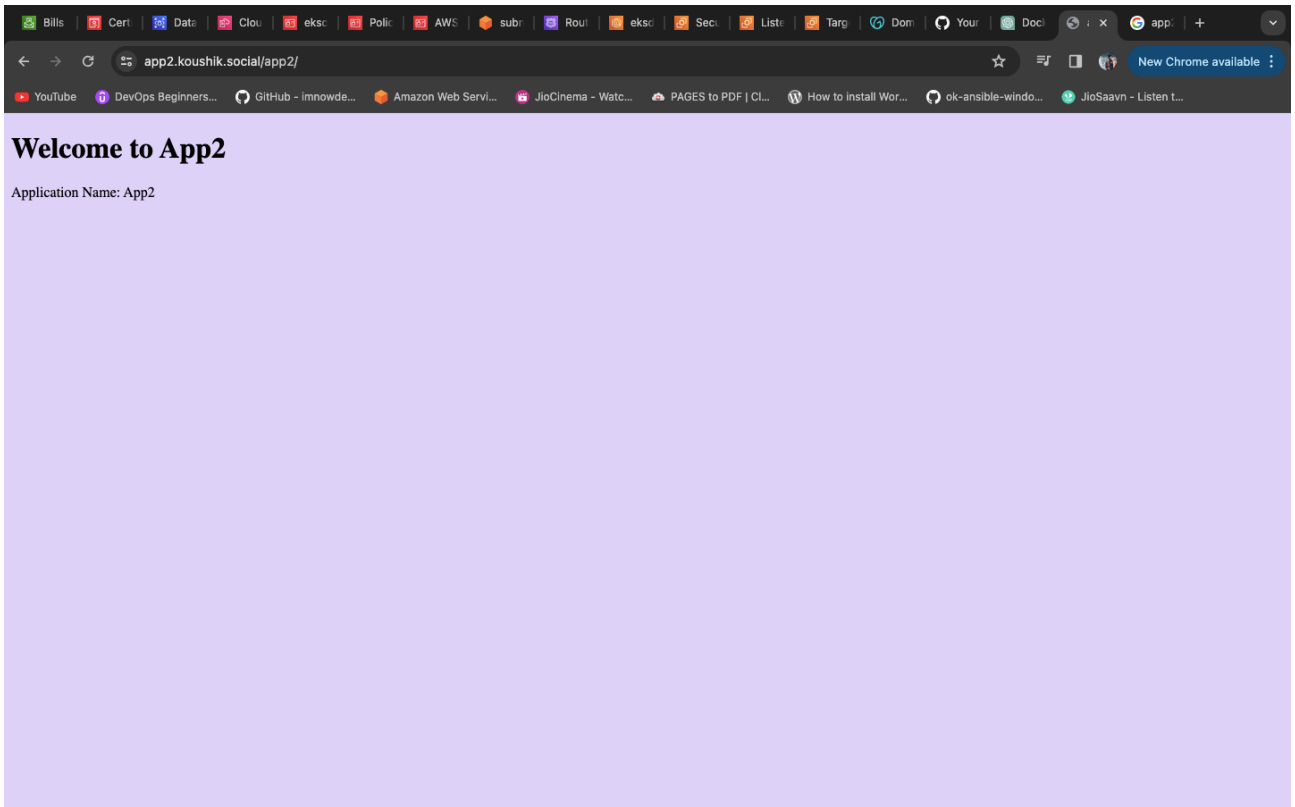
Anomaly mitigation: Not applicable | Deregister | Register targets

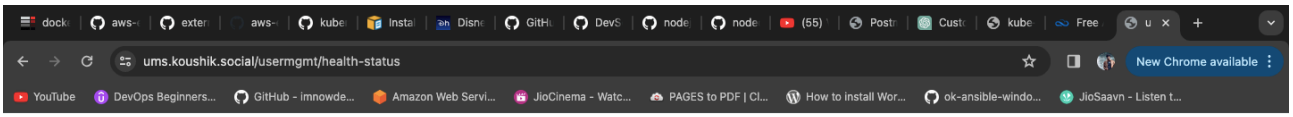
Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

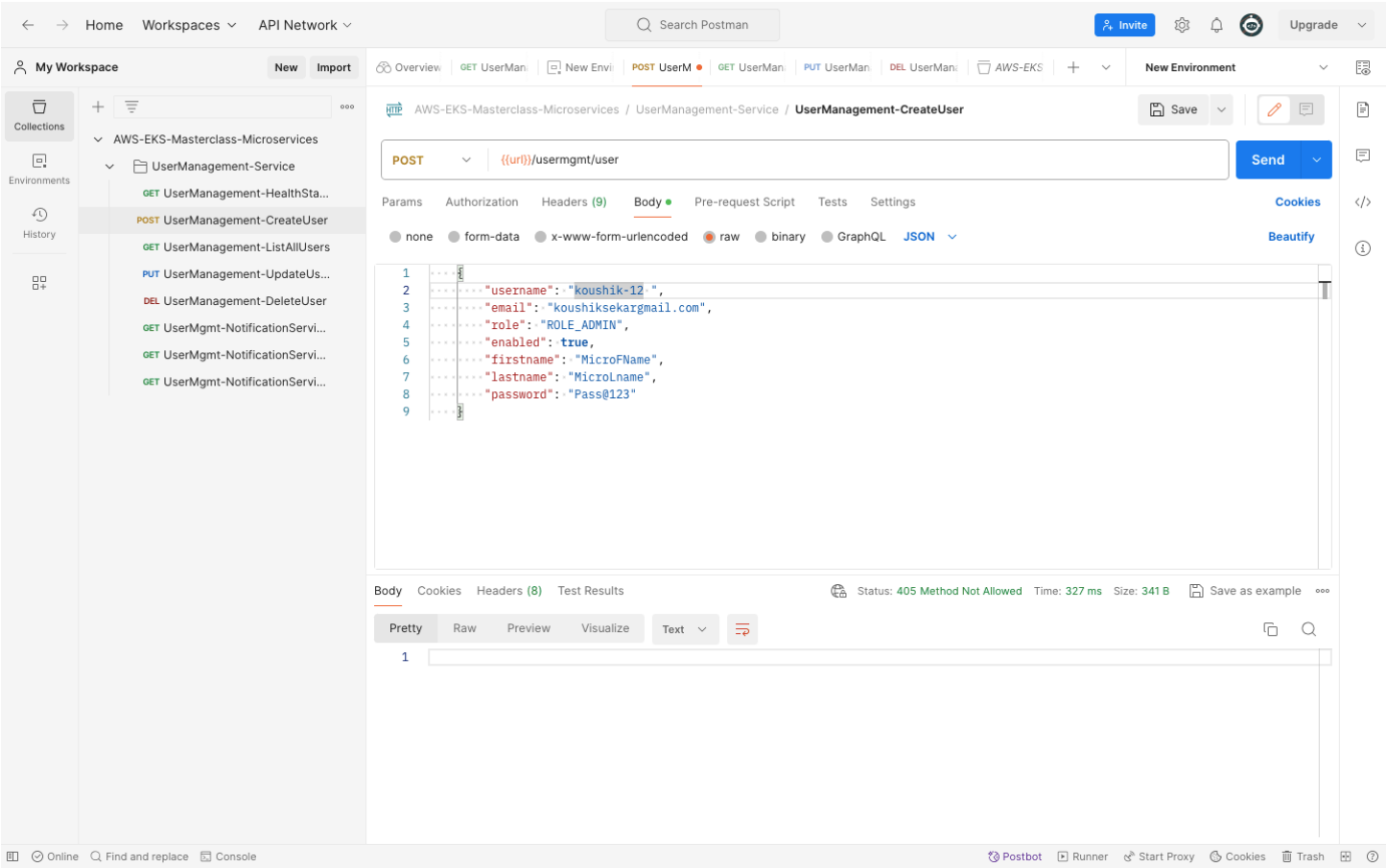
IP address	Port	Zone	Health status	Health status details	Anomaly detection result
192.168.71.41	8095	us-east-1a	Healthy	-	Normal
192.168.81.219	8095	us-east-1a	Healthy	-	Normal


app1.koushik.social/app1/
New Chrome available
Welcome to App1
Application Name: App1


app2.koushik.social/app2/
New Chrome available
Welcome to App2
Application Name: App2



User Management Service UP and RUNNING - V1



```
| sys                |
| usermgmt           |
+--------------------+
6 rows in set (0.00 sec)

mysql> use usermgmt;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables
    -> ^C
mysql> show tables;
+--------------------+
| Tables_in_usermgmt |
+--------------------+
| users              |
+--------------------+
1 row in set (0.00 sec)

mysql> select * from users;
+-------------+-----------------------+---------+------------+------------+-------------------------------------------------------------
--------+------------+
| username    | email                 | enabled | firstname  | lastname   | password
        | role       |
+-------------+-----------------------+---------+------------+------------+-------------------------------------------------------------
--------+------------+
| koushik-12  | koushiksekargmail.com |         | MicroFName | MicroLname | $2a$04$GpYpDxw7ENSyd/ln7oi1wOUZtlW0NKaTUNTeFc2Ee9jpCW2
kVZnXW | ROLE_ADMIN |
| koushik-07  | koushiksekargmail1.com |         | MicroFName | MicroLname | $2a$04$mTDhhloIfc6k7XODs1qgxekwBbeP1g3vw39YnzCT7WYf4d4
vpSgKq | ROLE_ADMIN |
+-------------+-----------------------+---------+------------+------------+-------------------------------------------------------------
--------+------------+
2 rows in set (0.00 sec)

mysql>
```

0.0 kB↓ 0.0 kB↑ 10% 4.1 GB 0.0 kB↓ 0.0 kB↑ 24/1, 3:21 PM