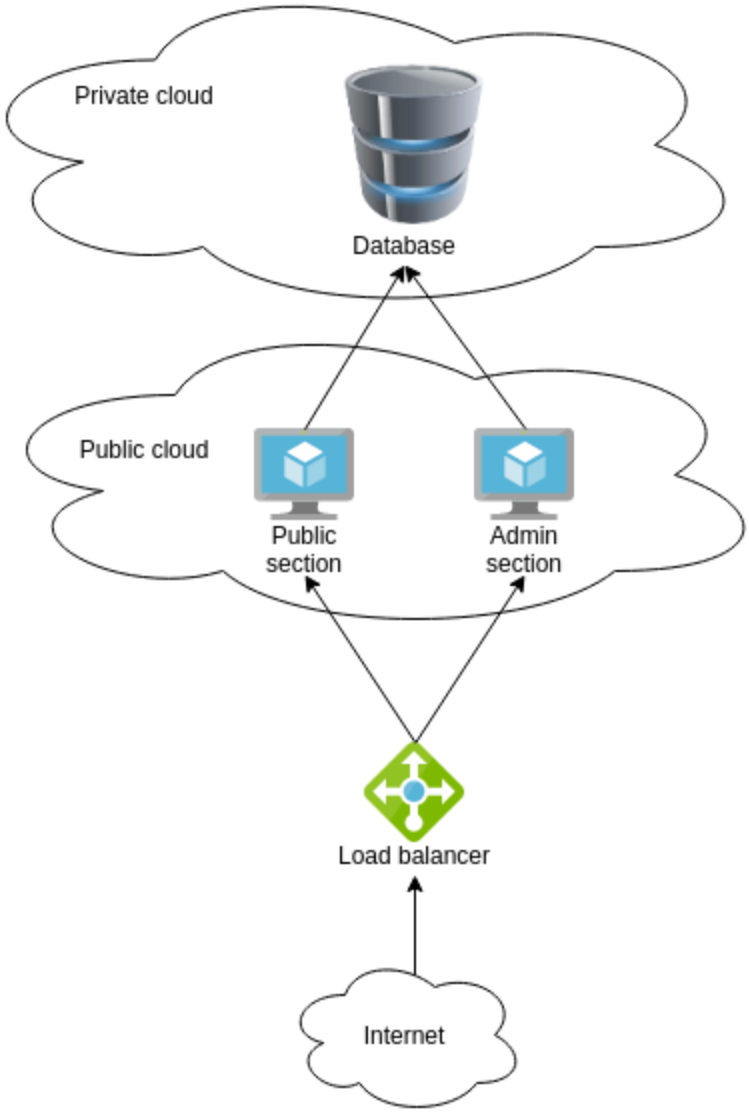
Deploying a opstycoon application on Kubernetes involves organizing your system into three distinct layers for optimal functionality and management.



Check NOTE.txt for details

Pre-requisites

Knowledge On Basic understanding on how these cloud tools and services works

Terraform

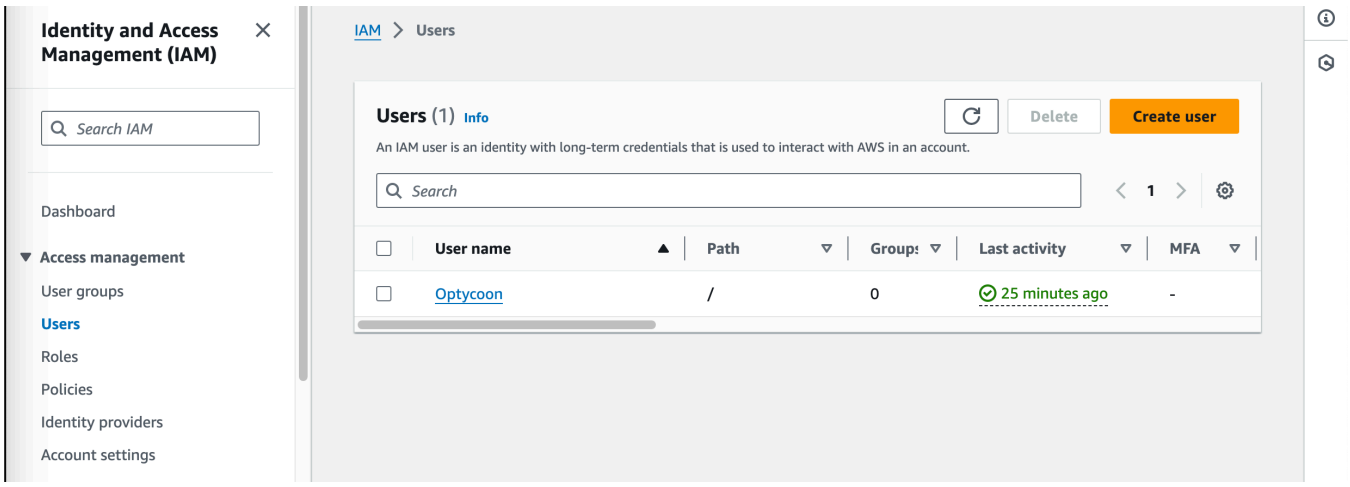
SSH

Amazon EC2

IAM (Identity and Access Management)

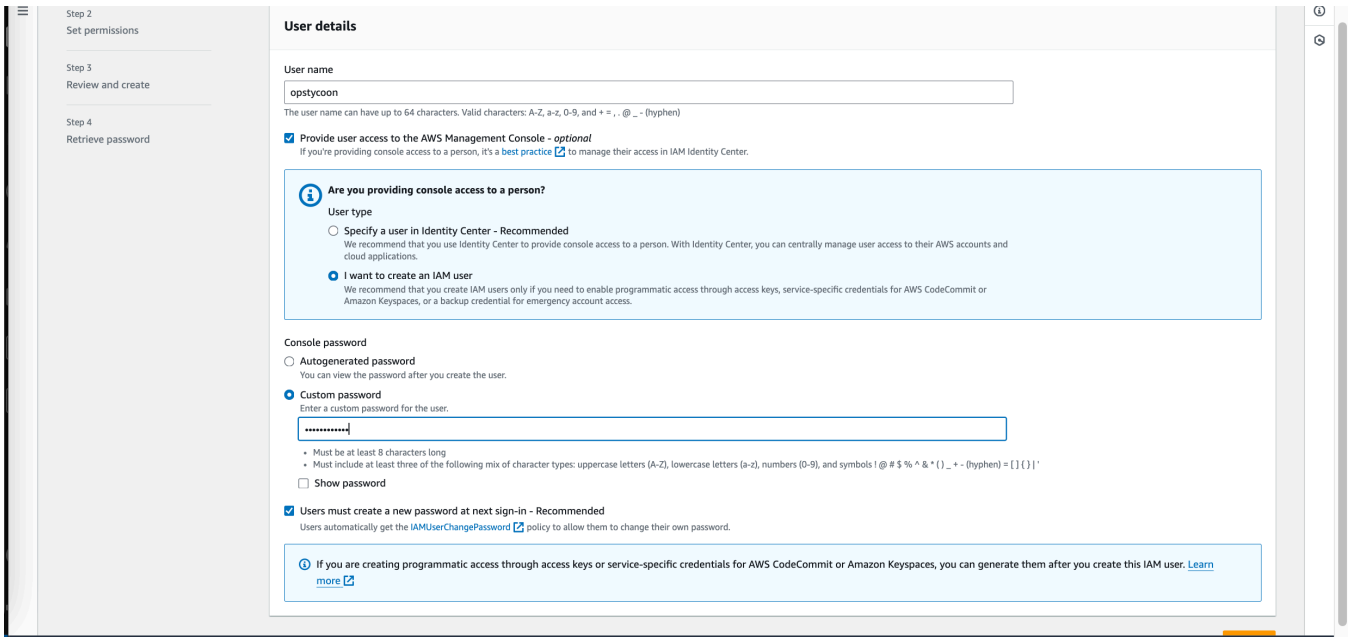
Amazon ECR (Elastic Container Registry)
Kubernetes Cluster
Helm

Step 1. Create an IAM user

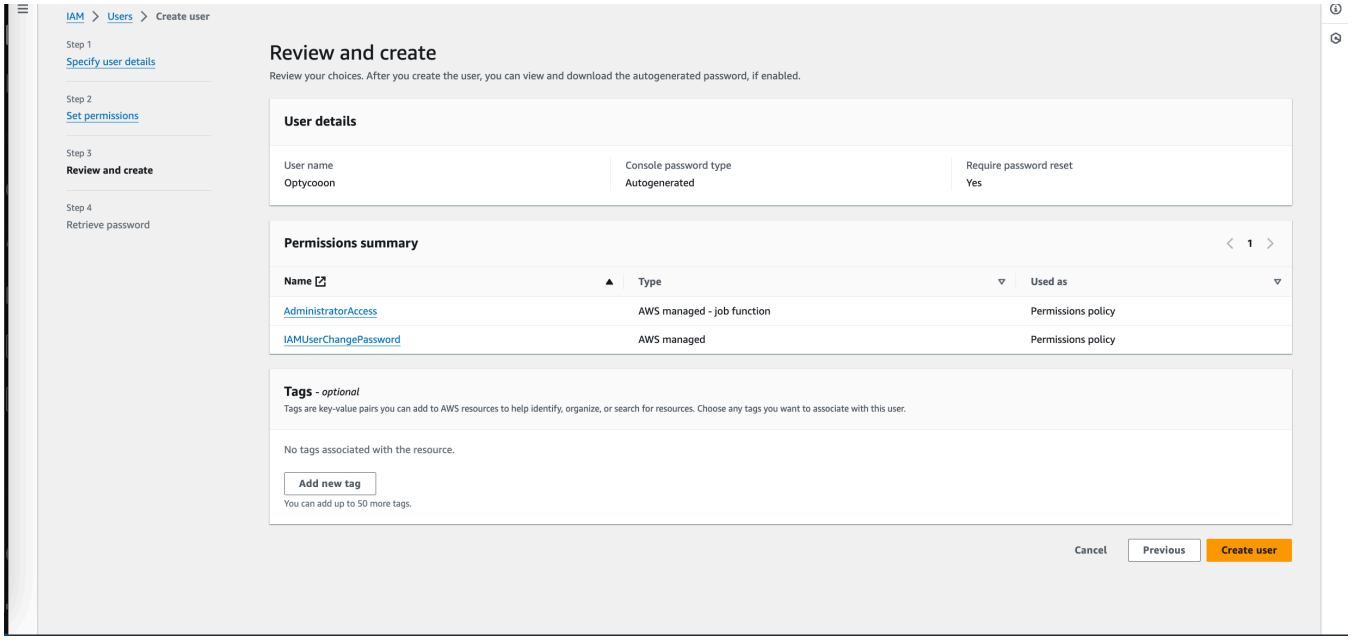


Click on user →create user, Give a name to your user and tick on provide user access to management console and then click on I want an IAM user option

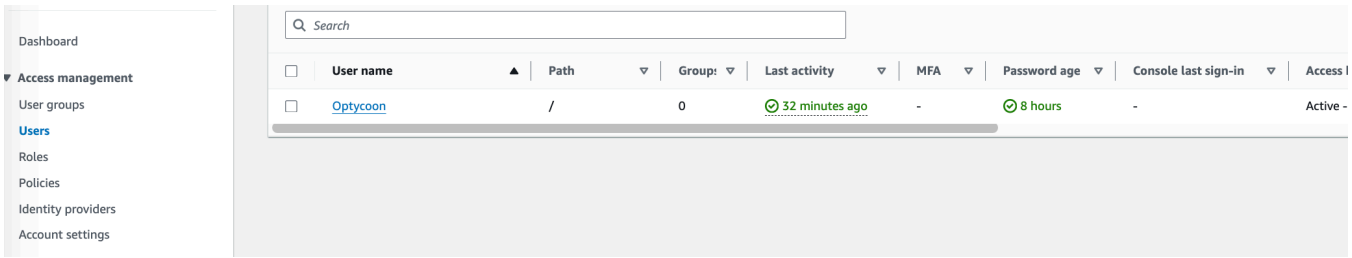
Choose a password for your user →click next, Attach the policies directly to your iam user → click next



Note →I will provide the administrator access for now but we careful while attaching the policies at your workspace



Review and create user, click on create user, download your password file if it is autogenerated otherwise it is your's choice



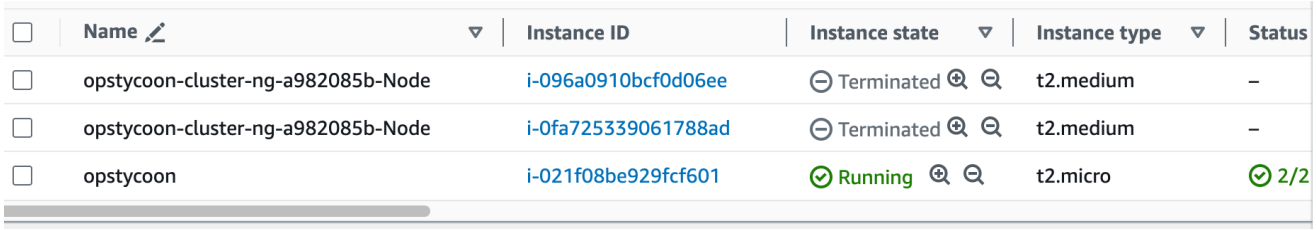
Now click on your IAM user →security credentials, scroll down to access keys and create an access keys

choose aws cli from the options listed

click next and download you csv file for username and password, I choose to copy this in a notepad instead, DON'T Copy this in your readme (Your account could be banned or restricted) if publicly exposed unknowningly

open your aws console and navigate to ec2 and click on launch ec2

Give it a name you want, I choose opstycoon,



I created a terraform to spine up the the ec2 instance instantly, fast and manage it easily, and easily destroy it after use.

If you choose to create take the same approach which i recommended to make use of IaC (Infrastructure-as-Code)

I created a main.tf

terraform > main.tf > terraform

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "eu-central-1"
}

resource "aws_instance" "instance_server" {
  ami           = "ami-04f9a173520f395dd"
  instance_type = "t2.micro"
  key_name      = "opstycoon-key"

  tags = {
    Name = "opstycoon"
  }
}

# # Create the first ECR repository
# resource "aws_ecr_repository" "opstycoon_fro
#   name = "opstycoon-frontend"
```

After this I configure my aws,

aws configure

(This will prompt you to insert the access key and the secret key you saved during the IAM creation)
enter your region and the format is json

Or You can use the AWS console if you choose

Connect to your instance and run the following commands

I ssh into my instance from terminal instead of using the console

`(ssh -i "my-key.pem" ubuntu@public-ip-here)`

Lets run
(sudo su - will allow you to work as a super-user;root-user), then update and create the folder

```
$ sudo su
$ apt update
$ mkdir opstycoon
$ cd opstycoon
```

fetch the code from github by git clone

```
git clone https://github.com/olawaleoyg/opstycoon.git
```

Cd to the folder

Run

```
ls -latr
```

To see the what is inside the repo

```
}'
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# cd ..
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-# ls
README.md  backend  frontend  k8s_manifests
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-# cd backend/
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backend# ls
Dockerfile  db.js  index.js  models  package-lock.json  package.json  routes
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backend# cd ..
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-# cd frontend/
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/frontend# ls
Dockerfile  package-lock.json  package.json  public  src
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/frontend# cd ..
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-# ls
README.md  backend  frontend  k8s_manifests
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-# cd k8s_manifests/
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# ls
backend-deployment.yaml  backend-service.yaml  frontend-deployment.yaml  frontend-service.yaml  full_stack_lb.yaml  iam_po
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# cd mongo/
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests/mongo# kubectl delete -f .
deployment.apps "mongodb" deleted
secret "mongo-sec" deleted
service "mongodb-svc" deleted
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests/mongo#
```

→ Setup aws cli, docker, kubectl and eksctl

AWS CLI is a tool that allows you to interact with AWS services using commands

run the following commands to install aws cli

```
snap install aws-cli --classic
```

→ configure aws by the command →

aws configure
insert the access key, secret key you got from the when you created your IAM user.

Enter your region and set the format to json

→ Let Install docker

```
apt install docker.io
usermod -aG docker $USER # Replace with your username e.g 'ubuntu'
newgrp docker
sudo chmod 777 /var/run/docker.sock
which docker
```

→ Let install kubectl

This is a command-line tool used in managing and interacting with Kubernetes clusters

```
To install kubectl run the following commands
snap install kubectl --classic
```

→ Install eksctl

It is a command-line tool used for managing Amazon EKS (Elastic Kubernetes Service) clusters.
To install eksctl tool run the following commands

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/el
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

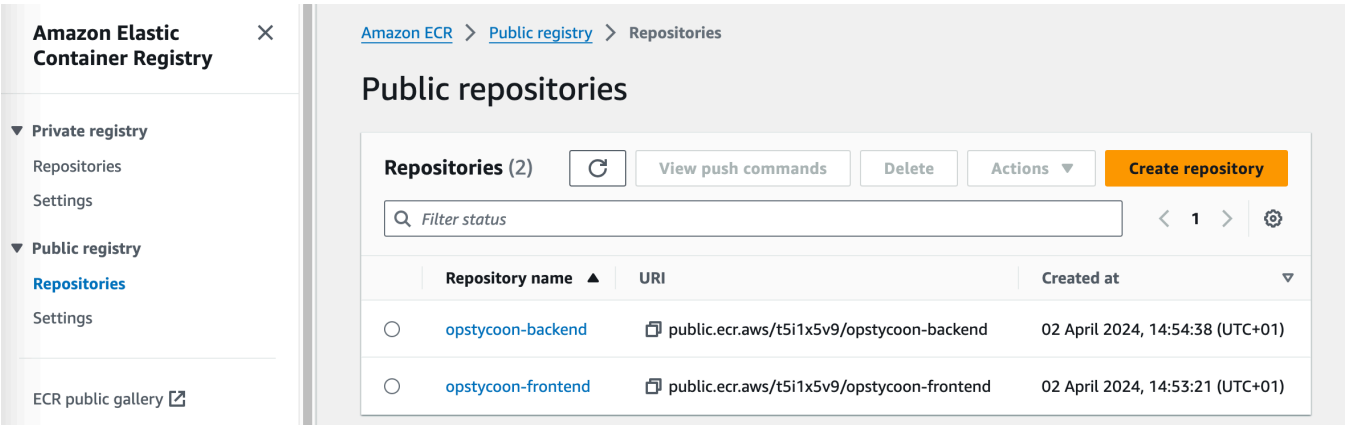
→ Phase 2 →Built frontend and backend images

You can create this from the console or addthe code to your terraform, make sure its public registry.

→ setup Elastic container registery (ECR)

It is similiar to dockerhub where we stored the docker images

Go to your aws console and search for ECR
click on create repository for frontend and select the public option



Look above to see mine.

→ Setup frontend

In terminal go to (cd) to frontend directory and run ls command

→ Go to your ecr repo and click on view push commands

This will give you command to log in, build, tag and push

You could check the Dockerfile to see the setup of what we are building.

→ Run the above command one by one to build the frontend image and push to ecr repository

NB: my image name could be different from yours, also the region I used for mine is eu-central-1

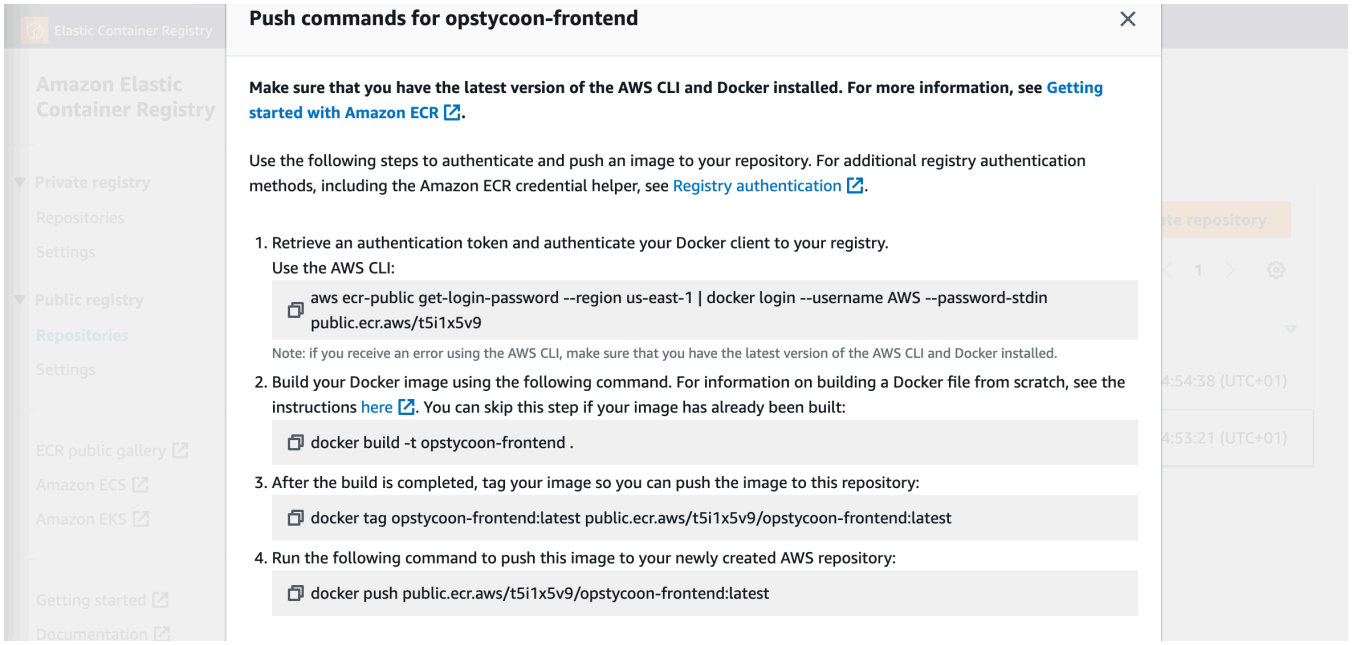
I make use of two terminals

```
root@ip-172-31-22-12: /home/ubuntu/opstycoon/Three-tier-Applica
root@ip-172-31-22-12: /home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k... ⌘1
ervice "mongodb-svc" deleted
oot@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_mani
-region eu-central-1
024-04-02 16:19:43 [ℹ]  deleting EKS cluster "opstycoon-cluster"
024-04-02 16:19:43 [ℹ]  will drain 0 unmanaged nodegroup(s) in cluster "opstycoon-clus
024-04-02 16:19:43 [ℹ]  starting parallel draining, max in-flight of 1
024-04-02 16:19:43 [ℹ]  deleted 0 Fargate profile(s)
024-04-02 16:19:44 [✓]  kubeconfig has been updated
024-04-02 16:19:44 [ℹ]  cleaning up AWS load balancers created by Kubernetes objects o
024-04-02 16:19:45 [ℹ]
 sequential tasks: { delete nodegroup "ng-a982085b",
   2 sequential sub-tasks: {
       2 sequential sub-tasks: {
           delete IAM role for serviceaccount "kube-system/aws-load-balancer-controlle
           delete serviceaccount "kube-system/aws-load-balancer-controller",
       },
       delete IAM OIDC provider,
   }, delete cluster control plane "opstycoon-cluster" [async]

024-04-02 16:19:45 [ℹ]  will delete stack "eksctl-opstycoon-cluster-nodegroup-ng-a9820
024-04-02 16:19:45 [ℹ]  waiting for stack "eksctl-opstycoon-cluster-nodegroup-ng-a9820
024-04-02 16:19:45 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:20:15 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:21:02 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:22:53 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:23:34 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:24:32 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:25:19 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:27:13 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:28:38 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:30:29 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-nod
024-04-02 16:30:30 [ℹ]  will delete stack "eksctl-opstycoon-cluster-addon-iamserviceac
024-04-02 16:30:30 [ℹ]  waiting for stack "eksctl-opstycoon-cluster-addon-iamserviceac
d
024-04-02 16:30:30 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-add
"
024-04-02 16:31:00 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-add
"
024-04-02 16:31:00 [ℹ]  deleted serviceaccount "kube-system/aws-load-balancer-controll
024-04-02 16:31:00 [ℹ]  will delete stack "eksctl-opstycoon-cluster-cluster"
024-04-02 16:31:00 [✓]  all cluster resources were deleted
```

```
root@ip-172-31-22-12: /home/ubuntu/opstycoon/Three-tier-Application-...  ⌥⌘2
  run `npm audit fix` to fix them, or `npm audit` for details
Removing intermediate container 63a6e6e48c92
 ---> cc8020421feb
Step 5/7 : COPY . .
 ---> 34a7e376d67d
Step 6/7 : EXPOSE 8080
 ---> Running in 0e29c85df972
Removing intermediate container 0e29c85df972
 ---> b45673788180
Step 7/7 : CMD [ "node", "index.js" ]
 ---> Running in 21c9a43e924d
Removing intermediate container 21c9a43e924d
 ---> eb0128fb6fc5
Successfully built eb0128fb6fc5
Successfully tagged opstycoon-backend:latest
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backe
nd# docker tag opstycoon-backend:latest public.ecr.aws/t5i1x5v9/opstycoon-backend:la
test
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backe
nd# docker push public.ecr.aws/t5i1x5v9/opstycoon-backend:latest
The push refers to repository [public.ecr.aws/t5i1x5v9/opstycoon-backend]
ee40a8cdb6bc: Pushed
de05abedfda2: Pushed
fddc09a9336b: Pushed
7c44f2a7bffb: Pushed
0d5f5a015e5d: Pushed
3c777d951de2: Pushed
f8a91dd5fc84: Pushed
cb81227abde5: Pushed
e01a454893a9: Pushed
c45660adde37: Pushed
fe0fb3ab4a0f: Pushed
f1186e5061f2: Pushed
b2dba7477754: Pushed
latest: digest: sha256:b75a51e2badf3a0ae9751c7374cc1b47492e9ad5130f79968a89f1c45b16b
4c2 size: 3049
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backe
nd# ls
Dockerfile  db.js  index.js  models  package-lock.json  package.json  routes
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backe
nd# aws cloudformation delete-stack --stack-name eksctl-opstycoon-cluster
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/backe
```

```
aws ecr-public get-login-password --region eu-central-1 | docker login --username AWS --pa
docker build -t opstycoon-frontend .
docker tag opstycoon-frontend:latest public.ecr.aws/l0l7e4u1/opstycoon-frontend:latest
docker push public.ecr.aws/l0l7e4u1/opstycoon-frontend:latest
```

→ Once the image is pushed, goto your ECR on the the console to verify



Welldone! We almost done.

Let’s run a container from the image

docker images -->copy the image name from the list

```
docker run -d -p 3000:3000 Opstycoon-frontend:latest
```

your frontend has setup and your application is now running to see your application you could browse →public-ip:3000

If your app is not showing
(Make sure you open a security group 3000 in your inbound rules)

Now do the same for backend! Try this and come back for the solution.

Let's do it together,

→ Setup backend

Now go to backend directory to setup backend

Go to your ecr repository and click on view push commands of backend repository

run the above command one by one in your terminal

- Login, Build, tag and push like you did in frontend

```
docker build -t opstycoon-frontend .
docker tag opstycoon-backend:latest public.ecr.aws/l0l7e4u1/opstycoon-backend:latest
docker push public.ecr.aws/l0l7e4u1/opstycoon-backend:latest
```

Now your backend image is built successfully and also pushed to Elastic container registry which we used when we create elastic kubernetes service

→ Phase 3 Kubernetes

→ What is Deployment?

Imagine a Factory: Think of a deployment as a factory that produces and manages copies of your software applications.
Multiple Replicas: Just like a factory can produce multiple identical items, a deployment in Kubernetes can create and handle multiple copies (replicas) of your application.
Easy Updates: If you want to change or update your application, the deployment system can smoothly handle that, like swapping out parts in a factory without stopping production.

→ What is Service?

Imagine a Reception Desk: Picture a service in Kubernetes like a reception desk in a building.
Central Point of Contact: The service provides a central point of contact for your applications. Instead of trying to find each application directly, other parts of your system can talk to the service, and it knows how to find the right application.
Stable Address: Just as you have a consistent address for the reception desk, a service has a stable address that other parts of your system can use to communicate with your applications.

→ What is Namespace?

It's like a labeled section within Kubernetes where you can organize and run your applications. Each namespace is like a fenced-off area where your apps can do their thing without stepping on each other's toes.
So, in simpler terms, a namespace in Kubernetes is a way to keep different projects or applications separate and organized, making it easier to manage them in the bustling environment of a Kubernetes cluster

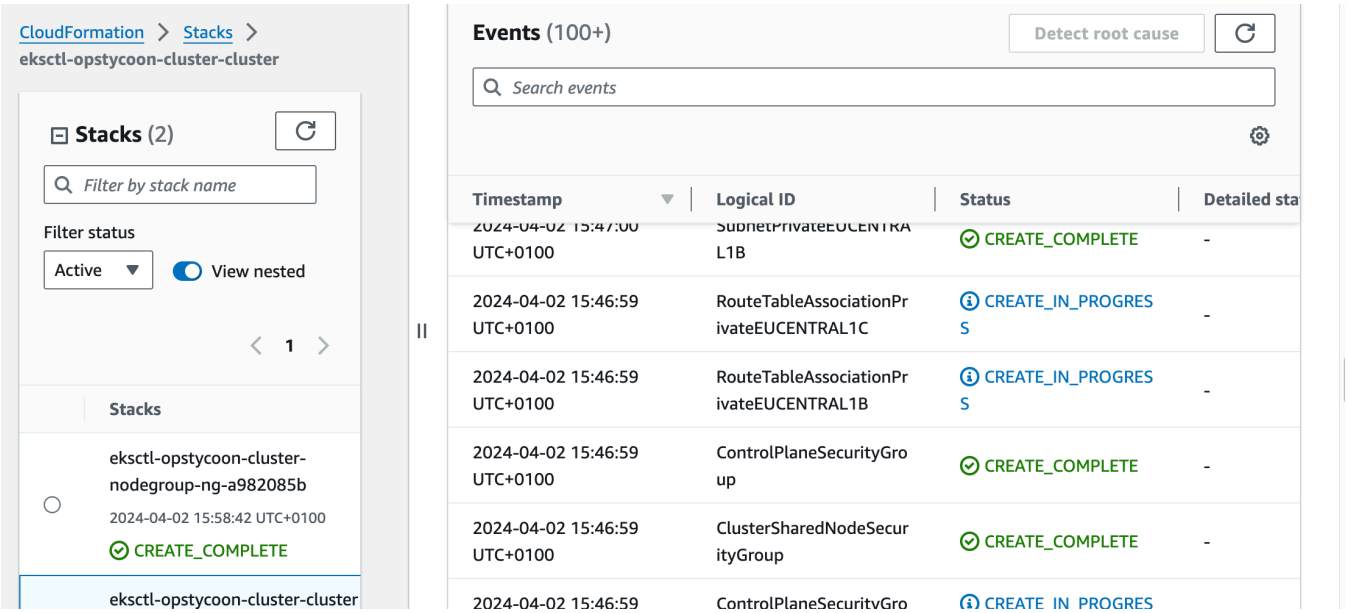
→ Setup EKS Cluster and create a namespace

Run the following command to setup EKS cluster

```
eksctl create cluster --name opstycoon-cluster --region eu-central-1 --node-type t2.mediu
aws eks update-kubeconfig --region eu-central-1 --name opstycoon-cluster
kubectl get nodes
```

It takes 15 to 20 mins to create a cluster

on aws console search for aws cloud formation to view the events happening in creation of EKS cluster



→ **creating Namespace from the following command**

```
kubectl create namespace workshop
kubectl config set-context --current --namespace workshop
```

→ **create a deployment and service for Frontend**

go to k8s_manifests directory there you will find deployment and service files for frontend

- You have to edit the file called frontend-deployment.yaml
- one thing you need to be changed that is your image name (You can see it in the ECR repository)

So, go to your ecr repository → select the frontend repository → click on view public listing and copy the image name and paste inside the frontend-deployment.yaml file

Now run the following commands to create the deployment and service for frontend

```
kubectl apply -f frontend-deployment.yaml
kubectl apply -f frontend-service.yaml
```

→ **Create a deployment and service for Backend**

In the same folder you will find backend-deployment.yaml and backend-service.yaml
you have to edit the file called backend-deployment.yaml
one thing you need to be changed that is your image name
so, go to your ecr repo → select the backend repository → click on view public listing and copy the image name and paste inside the backend-deployment.yaml file
Now run the following commands to create the deployment and service backend

```
kubectl apply -f backend-deployment.yaml
kubectl apply -f backend-service.yaml
kubectl get pods -n workshop
```

Now our two tier is ready that is frontend and backend let's setup the third tier

Let assume you run in trouble where your app is not running, troubleshoot, describe the pod, check the logs, verify if its pulling the right image

```
backend-6c5946ff87-jtklp    0/1     Running            3 (1s ago)    46s
backend-6c5946ff87-jtklp    0/1     CrashLoopBackOff   3 (0s ago)    60s
^Croot@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl get po
NAME                        READY   STATUS             RESTARTS      AGE
backend-6b499c988d-n9rqj    0/1     CrashLoopBackOff   5 (35s ago)   2m30s
backend-6c5946ff87-jtklp    0/1     CrashLoopBackOff   3 (12s ago)   72s
frontend-66b9694d7f-sspxt   1/1     Running            0             26m
mongodb-7f58c5f5d9-vzfrd    1/1     Running            0             6m57s
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# k logs backend-6c5946ff87-jtklp
k: command not found
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl logs backend-6c5946ff87-jtklp

> client@0.1.0 start /usr/src/app
> react-scripts start

ℹ 「wds」: Project is running at http://192.168.45.196/
ℹ 「wds」: webpack output is served from
ℹ 「wds」: Content not from webpack is served from /usr/src/app/public
ℹ 「wds」: 404s will fallback to /
Starting the development server...

Browserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db

Why you should do it regularly:
https://github.com/browserslist/browserslist#browsers-data-updating
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl get po
NAME                        READY   STATUS             RESTARTS      AGE
backend-6b499c988d-n9rqj    0/1     CrashLoopBackOff   5 (80s ago)   3m15s
backend-6c5946ff87-jtklp    0/1     CrashLoopBackOff   5 (2s ago)    117s
frontend-66b9694d7f-sspxt   1/1     Running            0             27m
mongodb-7f58c5f5d9-vzfrd    1/1     Running            0             7m42s
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl delete pod backend-6c5946ff87-jtklp
pod "backend-6c5946ff87-jtklp" deleted
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl get po
NAME                        READY   STATUS             RESTARTS     AGE
backend-6b499c988d-n9rqj    0/1     CrashLoopBackOff   6 (7s ago)   3m42s
backend-6c5946ff87-m64hk    0/1     Running            0            6s
frontend-66b9694d7f-sspxt   1/1     Running            0            28m
mongodb-7f58c5f5d9-vzfrd    1/1     Running            0            8m9s
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl delete pod backend-6c5946ff87-jtklp backend-
```

→ Setup Database tier

Locate the mongo folder that stores deployment, service and secrets manifests

Run the below commands to setup database tier

```
kubectl apply -f .
kubectl get all
```

Now your all three tiers are ready to go but how do you access them for that we have to create a application load balancer to route outside traffic towards cluster and an ingress for in internal routing between our 3 tiers

→ Setup Application Load balancer and ingress

we have to create a application load balancer to route outside traffic towards cluster and an ingress for in internal routing between our 3 tiers

→ **Setup aws load balancer ; installation and attachment it to your EKS cluster**

- Below command fetch the iam policy for your ALB

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.
```

This command create the iam policy in your aws account from iam_policy.json file that is setup in the first command

```
aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document f
```

```
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# kubectl get po -w
NAME                        READY   STATUS    RESTARTS   AGE
backend-7975cb8f6b-bvrdw    1/1     Running   0          14s
frontend-66b9694d7f-sspxt   1/1     Running   0          32m
mongodb-7f58c5f5d9-vzfrd    1/1     Running   0          12m
^Croot@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# curl -O https://raw.githubusercontent.com/kubernet
s-sigs/aws-load-balancer-controller/v2.5.4/docs/install/iam_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8386  100  8386    0     0  44370      0 --:--:-- --:--:-- --:--:-- 44370
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# aws iam create-policy --policy-name AWSLoadBalancerC
ntrollerIAMPolicy --policy-document file://iam_policy.json

{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "ANPATCKATNS5KVXRWW6ZB",
        "Arn": "arn:aws:iam::211125759162:policy/AWSLoadBalancerControllerIAMPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-04-02T16:00:41+00:00",
        "UpdateDate": "2024-04-02T16:00:41+00:00"
    }
}
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# eksctl utils associate-iam-oidc-provider --region=us
east-1 --cluster=three-tier-cluster --approve
Error: unable to describe cluster control plane: operation error EKS: DescribeCluster, https response error StatusCode: 404, RequestID: 163541fa-9
9e-4ac5-9a7e-ddb4dc4b07bc, ResourceNotFoundException: No cluster found for name: three-tier-cluster.
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# aws eks update-kubeconfig --region eu-central-1 --na
e opstycoon-cluster^C
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# eksctl utils associate-iam-oidc-provider --region=eu
central-1 --cluster=opstycoon-cluster --approve
2024-04-02 16:03:17 [ℹ]  will create IAM Open ID Connect provider for cluster "opstycoon-cluster" in "eu-central-1"
2024-04-02 16:03:17 [✔]  created IAM Open ID Connect provider for cluster "opstycoon-cluster" in "eu-central-1"
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
```

This command apply the load balancer policy to your eks cluster so that your eks cluster is working with your load balancer according to the policy

```
eksctl utils associate-iam-oidc-provider --region=eu-central-1--cluster=opstycoon-cluster
```

This command create and attach an service account to your cluster so that your cluster is

allowed to work with load balancer service

please change your aws account number. from the below command otherwise it won't work

```
eksctl create iamserviceaccount --cluster=opstycoon-cluster --namespace=kube-system --name
```

```
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# eksctl create iamserviceaccount --cluster=three-tier
cluster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:ia
::211125759162:policy/AWSLoadBalancerControllerIAMPolicy --approve --region=eu-central-1
Error: unable to describe cluster control plane: operation error EKS: DescribeCluster, https response error StatusCode: 404, RequestID: 7c0cba18-0
5c-45eb-ae63-7196e69ed71d, ResourceNotFoundException: No cluster found for name: three-tier-cluster.
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# eksctl create iamserviceaccount --cluster=opstycoon-
luster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam
:211125759162:policy/AWSLoadBalancerControllerIAMPolicy --approve --region=eu-central-1
2024-04-02 16:05:06 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2024-04-02 16:05:06 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2024-04-02 16:05:06 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
        create serviceaccount "kube-system/aws-load-balancer-controller",
    } }2024-04-02 16:05:06 [ℹ]  building iamserviceaccount stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-c
ntroller"
2024-04-02 16:05:06 [ℹ]  deploying stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-04-02 16:05:06 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controll
r"
2024-04-02 16:05:36 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controll
r"
2024-04-02 16:05:36 [ℹ]  created serviceaccount "kube-system/aws-load-balancer-controller"
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# sudo snap install helm --classic
helm 3.14.3 from Snapcrafters✪ installed
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm repo update eks
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. *Happy Helming!*
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
```

All the policies are attached let's deploy the load balancer

- For this we have to install helm→Helm is a special tool that helps you easily carry and manage your software when you're using Kubernetes, which is like a big playground for running applications.

```
sudo snap install helm --classic
```

```
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# eksctl create iamserviceaccount --cluster=three-tier
cluster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:ia
::211125759162:policy/AWSLoadBalancerControllerIAMPolicy --approve --region=eu-central-1
Error: unable to describe cluster control plane: operation error EKS: DescribeCluster, https response error StatusCode: 404, RequestID: 7c0cba18-0
5c-45eb-ae63-7196e69ed71d, ResourceNotFoundException: No cluster found for name: three-tier-cluster.
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# eksctl create iamserviceaccount --cluster=opstycoon-
luster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam
:211125759162:policy/AWSLoadBalancerControllerIAMPolicy --approve --region=eu-central-1
2024-04-02 16:05:06 [ℹ]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2024-04-02 16:05:06 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2024-04-02 16:05:06 [ℹ]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
        create serviceaccount "kube-system/aws-load-balancer-controller",
    } }2024-04-02 16:05:06 [ℹ]  building iamserviceaccount stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-c
ntroller"
2024-04-02 16:05:06 [ℹ]  deploying stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-04-02 16:05:06 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controll
r"
2024-04-02 16:05:36 [ℹ]  waiting for CloudFormation stack "eksctl-opstycoon-cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controll
r"
2024-04-02 16:05:36 [ℹ]  created serviceaccount "kube-system/aws-load-balancer-controller"
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# sudo snap install helm --classic
helm 3.14.3 from Snapcrafters✪ installed
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm repo update eks
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. ⎈Happy Helming!⎈
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
```

After this we have to add a particular manifest for load balancer that is pre written by someone on eks repo by using helm

```
helm repo add eks https://aws.github.io/eks-charts
```

- update the eks repo using helm

```
helm repo update eks
```

Install the load balancer controller on your eks cluster

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n kube-system
kubectl get deployment -n kube-system aws-load-balancer-controller
```

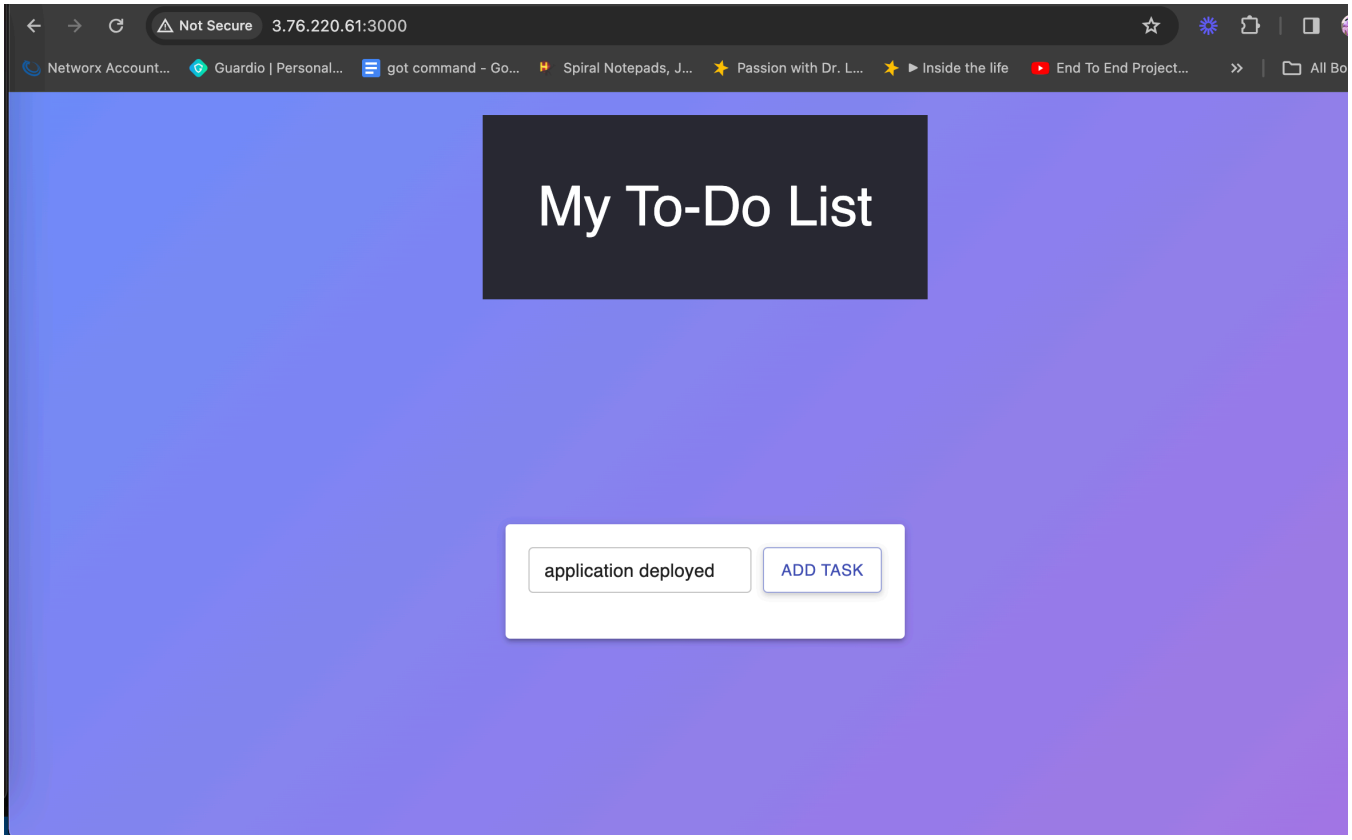
```
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# sudo snap install helm --classic
helm 3.14.3 from Snapcrafters⚙ installed
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm repo add eks https://aws.github.io/eks-charts
"eks" has been added to your repositories
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm repo update eks
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
Update Complete. *Happy Helming!*
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests# helm install aws-load-balancer-controller eks/aws-l
d-balancer-controller -n kube-system --set clusterName=my-cluster --set serviceAccount.create=false --set serviceAccount.name=aws-load-balancer-c
troller

NAME: aws-load-balancer-controller
LAST DEPLOYED: Tue Apr  2 16:08:05 2024
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
root@ip-172-31-22-12:/home/ubuntu/opstycoon/Three-tier-Application-Deployment-/k8s_manifests#
```

Now your Load balancer is working let’s setup Ingress for internal routing

- Look for the full_stack_lb.yaml file

```
kubectl apply -f full_stack_lb.yaml
kubectl get ing -n workshop
```

Go to your Web Browser and paste the above dns address, wait some while



Your application is accessible through load balancer ingress

If it doesn't copy your public Ip:3000 to se if you can access it, if that worked, then the DNS in your ingress should work.

Congratulations!

→ **Destroy Everything**

- On your current folder run
  text

On your current folder run

```
kubectl delete -f .
```

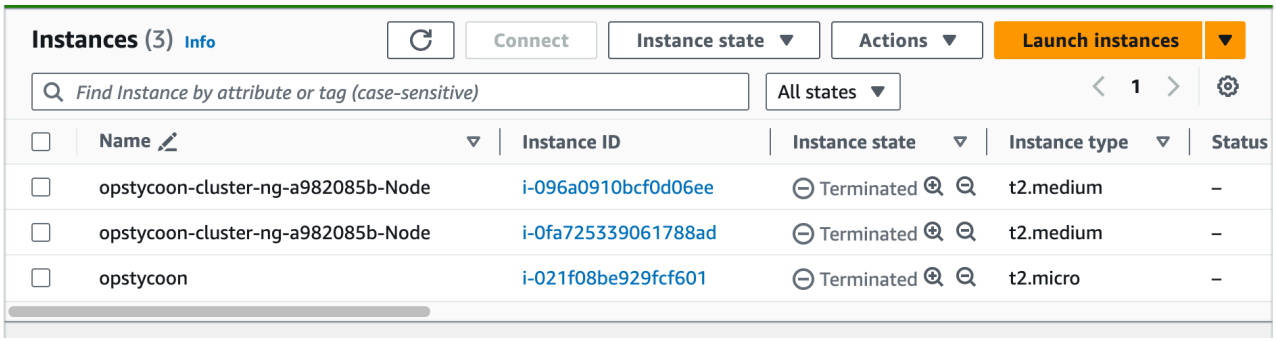
- go to mongo folder to delete database tier

```
kubectl delete -f .
```

- Delete the cluster and the stack of your cloud formation

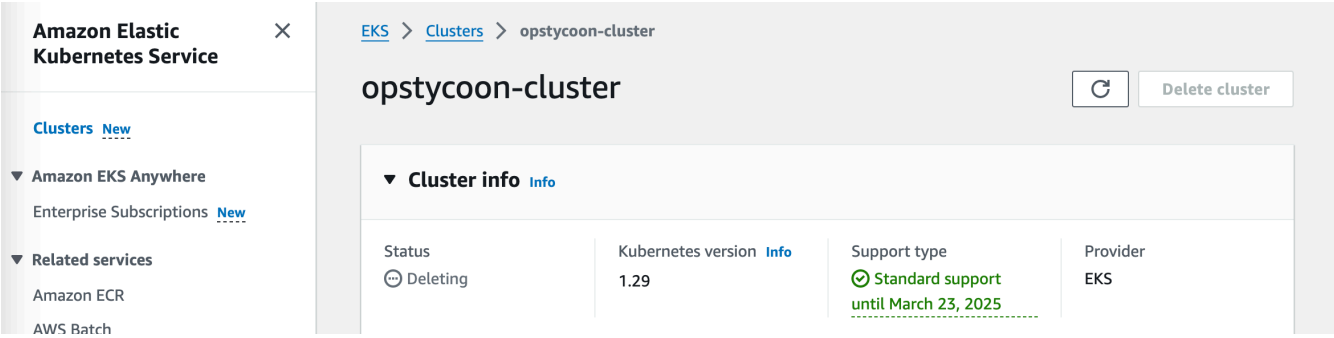
```
eksctl delete cluster --name three-tier-cluster --region us-east-1
aws cloudformation delete-stack --stack-name eksctl-three-tier-cluster-cluster
```

- you could checkout all the changes in cloud formation console of aws



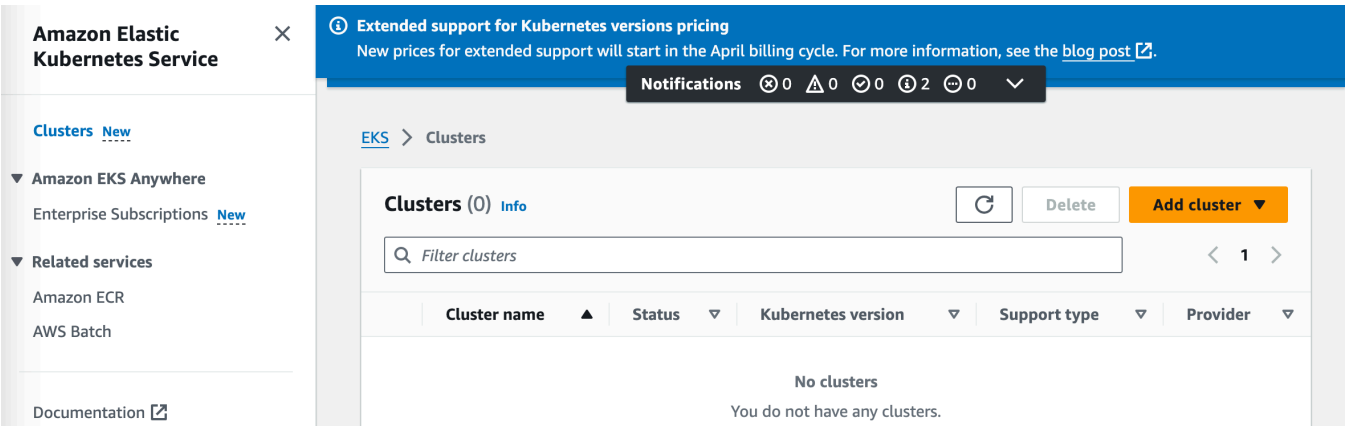
Everything is deleted now thanks me for reducing your aws bill
well our project is completed here if you comes at this point do clap, well done.

Verify from the console to check if they are deleted.



For me I just delete all these with a command by using terraform,

```
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.example_server will be destroyed
  # (because aws_instance.example_server is not in configuration)
  - resource "aws_instance" "example_server" {
      - ami                                  = "ami-04f9a173520f395dd" -> null
      - arn                                  = "arn:aws:ec2:eu-central-1:211125759162:instance/i-021f08be929fcf601" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "eu-central-1a" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-021f08be929fcf601" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - key_name                             = "opstycoon-key" -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-0bb3a7b52e1db8790" -> null
      - private_dns                          = "ip-172-31-22-12.eu-central-1.compute.internal" -> null
      - private_ip                           = "172.31.22.12" -> null
      - public_dns                           = "ec2-3-76-220-61.eu-central-1.compute.amazonaws.com" -> null
      - public_ip                            = "3.76.220.61" -> null
      - secondary_private_ips                = [] -> null
      - security_groups                      = [
```



Don't forget to delete when you are done, else you will be charged for the resources.

Thank you.

- Opstycoon