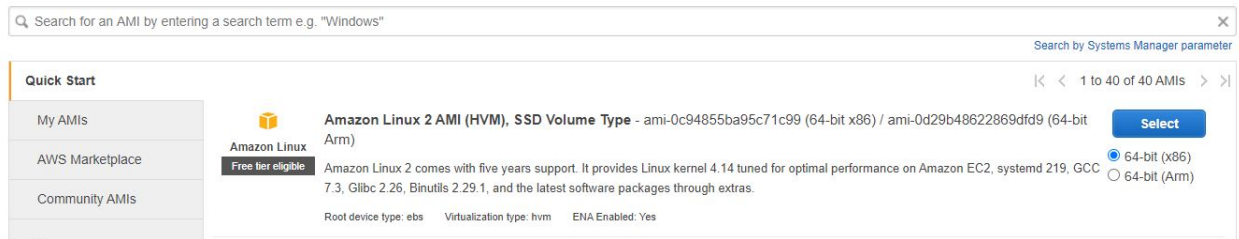


1. Launch Linux EC2 instance in AWS

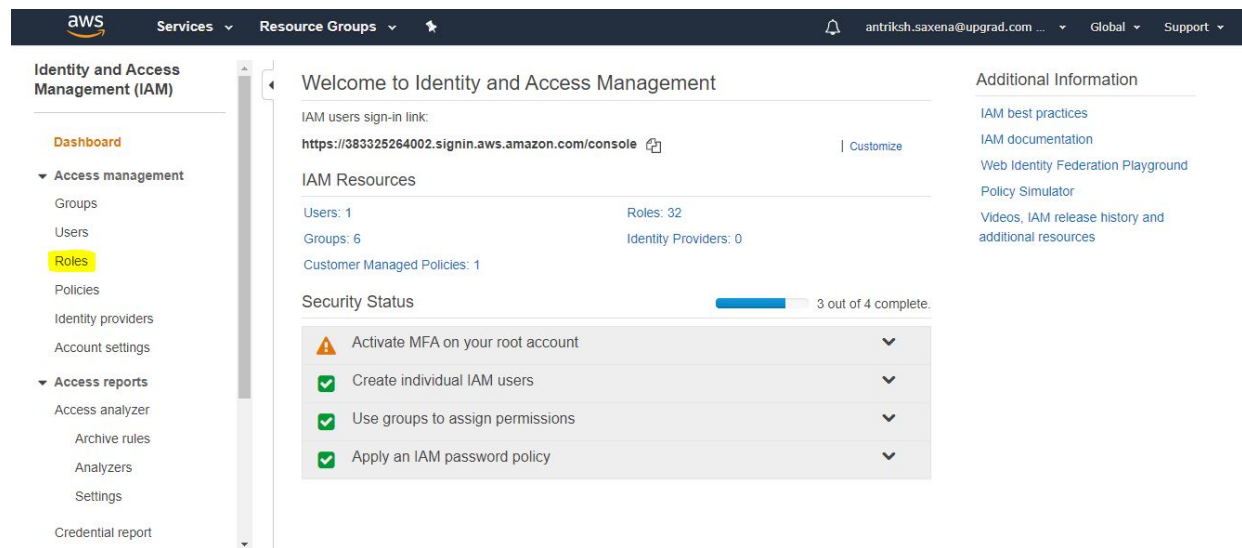


2. Create and attach the relevant policies through the IAM role to EC2 Instance. Kops need permissions to access, S3, EC2, VPC, Route53, Autoscaling, etc.

(Note: You can directly attach the [AdministratorAccess](#) policy to the role)

Create an IAM role:

- Search for IAM in the services and go to the IAM policy page and click to the roles.



- Next, you need to create a role.

Identity and Access Management (IAM)

- Dashboard
- Access management
 - Groups
 - Users
 - Roles**
 - Policies
 - Identity providers
 - Account settings
- Access reports
 - Access analyzer
 - Archive rules
 - Analyzers
 - Settings
 - Credential report

Roles

What are IAM roles?

IAM roles are a secure way to grant permissions to entities that you trust. Examples of entities include the following:

- IAM user in another account
- Application code running on an EC2 instance that needs to perform actions on AWS resources
- An AWS service that needs to act on resources in your account to provide its features
- Users from a corporate directory who use identity federation with SAML

IAM roles issue keys that are valid for short durations, making them a more secure way to grant access.

Additional resources:

- [IAM Roles FAQ](#)
- [IAM Roles Documentation](#)
- [Tutorial: Setting Up Cross Account Access](#)
- [Common Scenarios for Roles](#)

[Create role](#) [Delete role](#)

c. Select the use cases as EC2:

Create role

1 2 3 4

Select type of trusted entity

- AWS service**
EC2, Lambda and others
Allows AWS services to perform actions on your behalf. [Learn more](#)
- Another AWS account
Belonging to you or 3rd party
- Web identity
Cognito or any OpenID provider
- SAML 2.0 federation
Your corporate directory

Choose a use case

Common use cases

- EC2**
Allows EC2 instances to call AWS services on your behalf.
- Lambda
Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

- [API Gateway](#)
- [CodeDeploy](#)
- [EMR](#)
- [KMS](#)
- [Rekognition](#)

* Required

[Cancel](#) [Next: Permissions](#)

d. In the next step, check the AdministratorAccess policy.

Choose one or more policies to attach to your new role.

Create policy ↻

Filter policies ▼ Showing 708 results

	Policy name ▼	Used as
<input type="checkbox"/>	AccessAnalyzerServiceRolePolicy	None
<input checked="" type="checkbox"/>	AdministratorAccess	Permissions policy (5)
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	None
<input type="checkbox"/>	AlexaForBusinessFullAccess	None
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	None
<input type="checkbox"/>	AlexaForBusinessLifesizeDelegatedAccessPolicy	None
<input type="checkbox"/>	AlexaForBusinessNetworkProfileServicePolicy	None
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	None

Set permissions boundary

* Required Cancel Previous Next: Tags

e. Enter a role name and press create role.

Create role 1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name*
Use alphanumeric and "+=, @-_" characters. Maximum 64 characters.
 A role named "kops-iam" already exists

Role description
Maximum 1000 characters. Use alphanumeric and "+=, @-_" characters.

Trusted entities AWS service: ec2.amazonaws.com

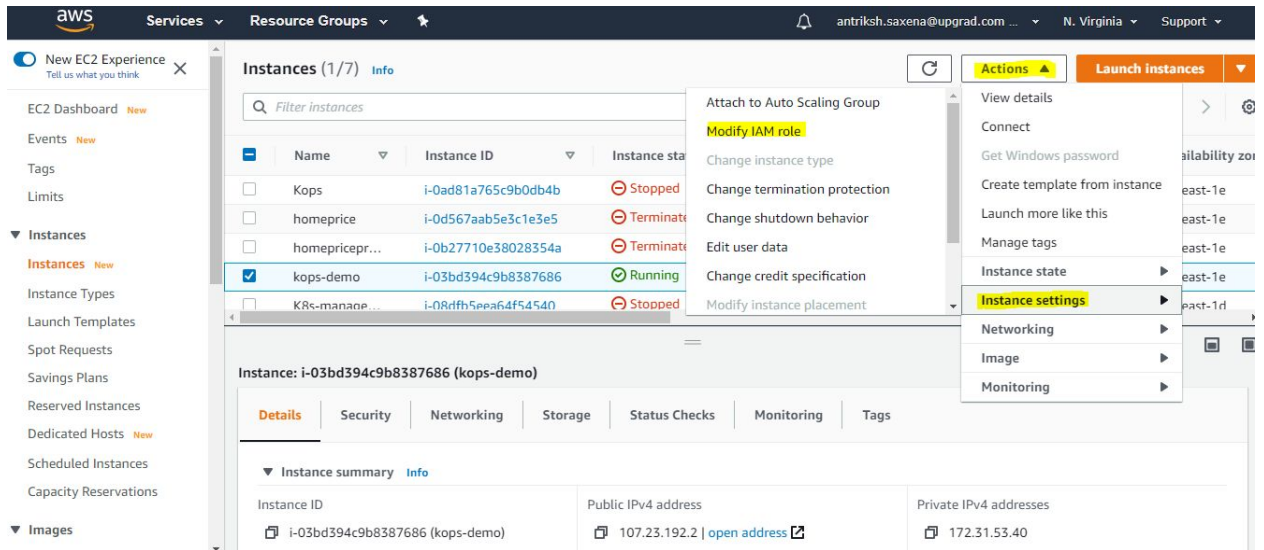
Policies AdministratorAccess [↗](#)

Permissions boundary Permissions boundary is not set

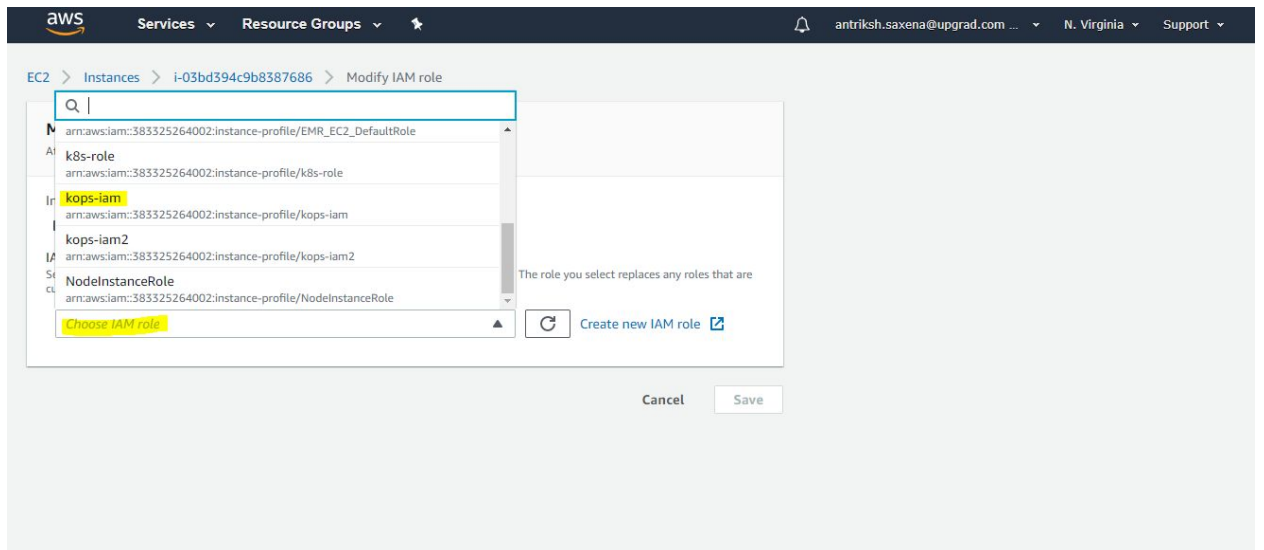
* Required Cancel Previous Create role

The next step is to attach this IAM role to the already created EC2 instance.

a. Go to the instance settings and then modify the IAM role.



b. Choose the appropriate IAM role and save it.



3. Install Kops on EC2

a) Download kops binary

```
curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)/kops-linux-amd64
```

b) Execute permissions for binary

```
chmod +x kops-linux-amd64
```

c) Move binary to usr/local/bin so the command is in path

```
sudo mv kops-linux-amd64 /usr/local/bin/kops
```

d) Check installation

```
kops version
```

The final output should come out as shown below

```
[ec2-user@ip-172-31-47-90 ~]$ kops version  
Version 1.18.0 (git-698bf974d8)
```

4. Install kubectl

a) Download kubectl binary

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl  
-s  
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/l  
inux/amd64/kubectl
```

b) Execute permissions for binary

```
chmod +x ./kubectl
```

c) Move binary to usr/local/bin so the command is in path

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

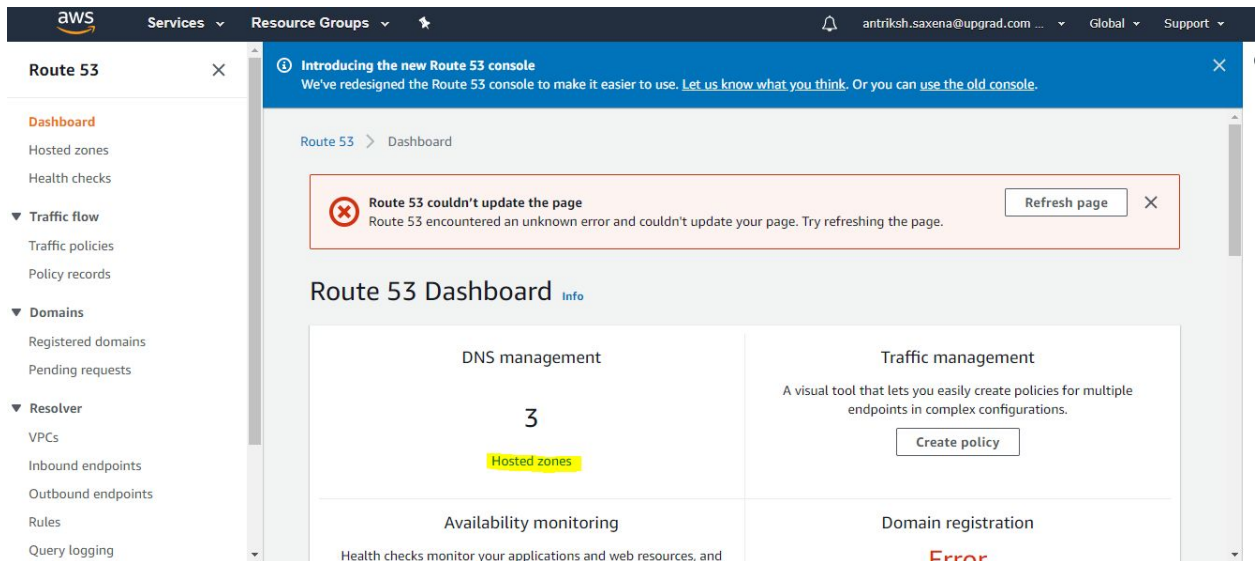
5. Create S3 Bucket in AWS

- S3 bucket is used by kubernetes to persist cluster state
- **Note:** Make sure you choose bucket name that is unique across all AWS accounts

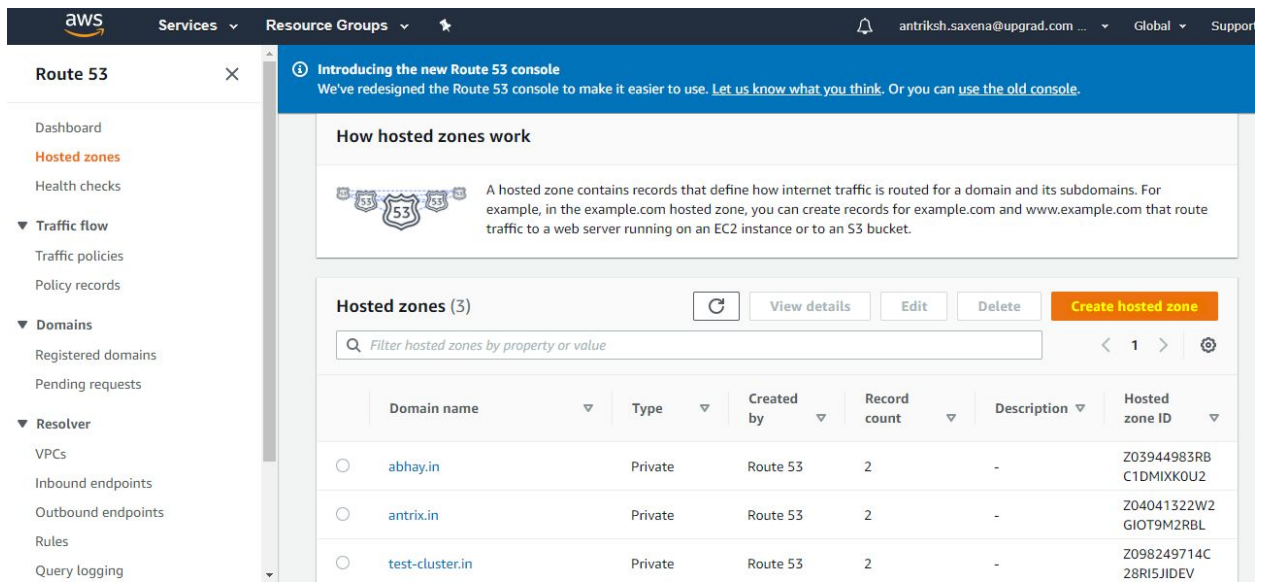
```
aws s3 mb s3://kops-bucket.in.k8s --region us-east-1
```

6. Create private hosted zone in AWS Route53

- a. Head over to aws Route53 and create hostedzone.



Click on the create hosted zone:



- b. Choose a name for example (test-home.in).
c. Choose type as a **private hosted zone** for VPC.

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain name Info
This is the name of the domain that you want to route traffic for.
test-cluster.in
Valid characters: a-z, 0-9, !"#\$%&'()*+,-./:;<=>?@[\\]^_`{|}.,~

Description - optional Info
This value lets you distinguish hosted zones that have the same name.
The hosted zone is used for...
The description can have up to 256 characters. 0/256

Type Info
The type indicates whether you want to route traffic on the internet or in an Amazon VPC.

☐ Public hosted zone
A public hosted zone determines how traffic is routed on the internet.

☒ Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.

- d. Select default vpc in the region you are setting up your cluster.

VPCs to associate with the hosted zone Info
To use this hosted zone to resolve DNS queries for one or more VPCs, choose the VPCs. To associate a VPC with a hosted zone when the VPC was created using a different AWS account, you must use a programmatic method, such as the AWS CLI.

For each VPC that you associate with a private hosted zone, you must set the Amazon VPC settings enableDnsHostnames and enableDnsSupport to true.

Region Info
US East (N. Virginia) [us-east-1]

VPC ID Info
vpc-a1a1d2db

Add VPC Remove VPC

Tags Info
Apply tags to hosted zones to help organize and identify them.

- e. Press the create button.

7. Configure environment variables.

Open .bashrc file

```
vi ~/.bashrc
```

Add the following content into .bashrc, you can choose any arbitrary name for the cluster and make sure the bucket name matches the one you created in step 5.


```
export KOPS_CLUSTER_NAME=test-cluster.in
export KOPS_STATE_STORE=s3://kops-bucket.in.k8s
```

Then run the following command to reflect the variables added to .bashrc

```
source ~/.bashrc
```

8. Create ssh key pair

This keypair is used for ssh into Kubernetes cluster

```
ssh-keygen
```

To which it will return

```
[ec2-user@ip-172-31-47-90 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6tDQTSauficOXOsSqYtHMSe2myrcjMyMJBAjQa+nZLY ec2-user@ip-172-31-47-90.ec2.internal
The key's randomart image is:
+----[RSA 2048]-----+
|oo                    |
|+ .                   |
|.o . . . o           |
|. * .o =             |
|. * B..+ S           |
|=. *.o= o            |
|OE++=.+.             |
|oB=+. =o .           |
|+o...++o             |
+----[SHA256]-----+
```

9. Create a Kubernetes cluster definition using the following command

```
kops create cluster \
--state=${KOPS_STATE_STORE} \
--node-count=2 \
--master-size=t2.micro \
--node-size=t2.micro \
--zones=us-east-1b \
--name=${KOPS_CLUSTER_NAME} \
```



```
--dns private \  
--master-count 1
```

10. Now create the kubernetes cluster using the following command

```
kops update cluster --yes
```

Above command may take some time to create the required infrastructure resources on AWS. Execute the validate command to check its status and wait until the cluster becomes ready

```
kops validate cluster
```

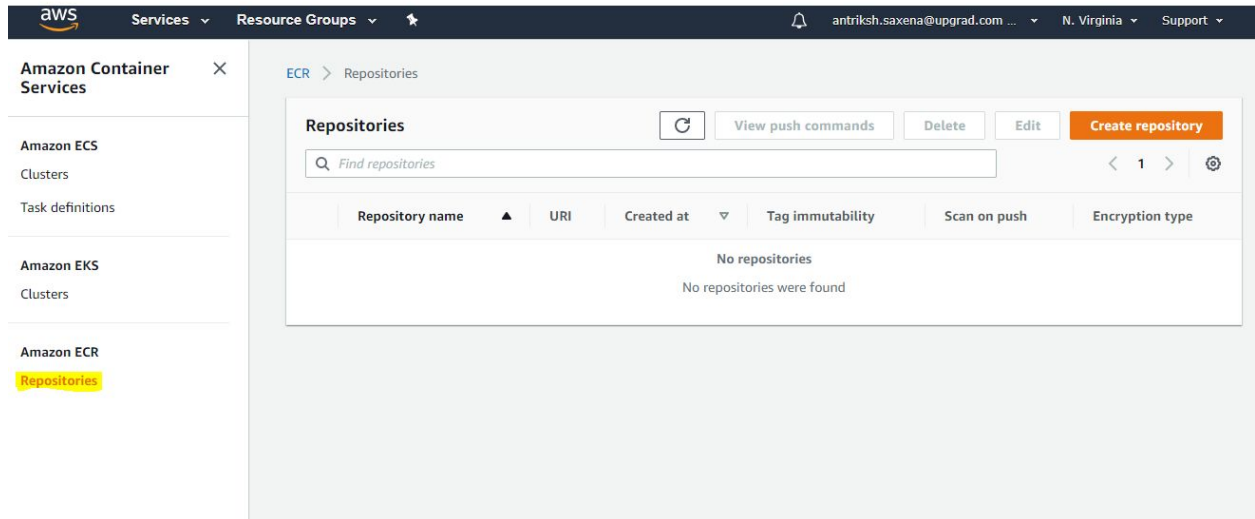
For the above command, you might see validation failed errors initially when you create a cluster and it is an expected behaviour. This means you will have to wait for some time as the cluster is still in its formation stage and you can try and run this command again after sometime.

Once the cluster is ready, the result of the command will be as follows:

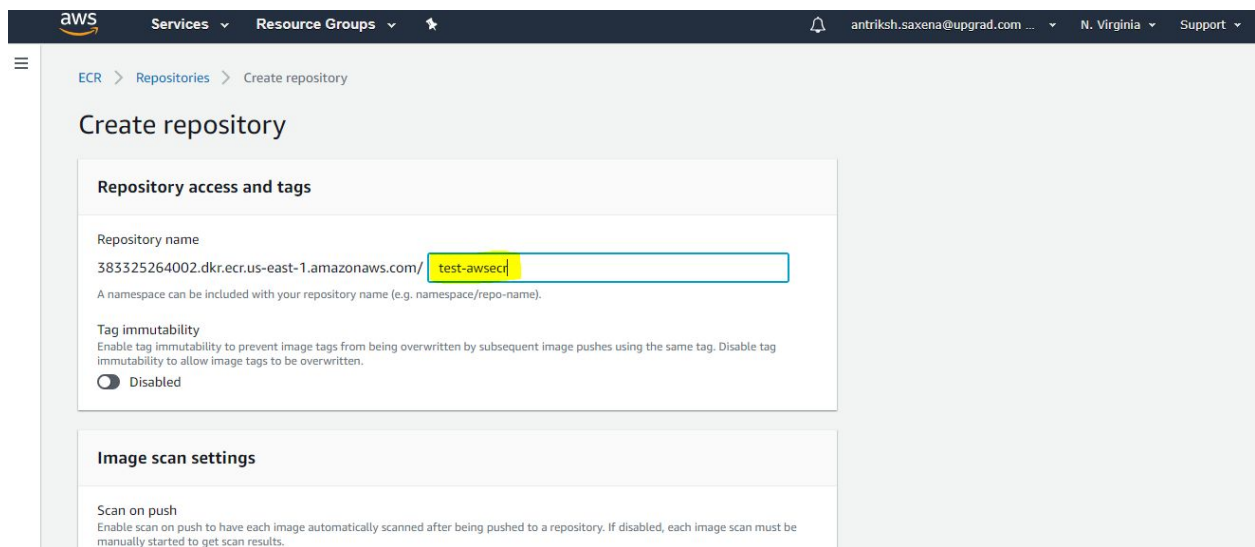
```
Validating cluster test-cluster.in  
  
INSTANCE GROUPS  
NAME                ROLE    MACHINETYPE    MIN    MAX    SUBNETS  
master-us-east-1b   Master  t2.micro        1      1      us-east-1b  
nodes               Node    t2.micro        2      2      us-east-1b  
  
NODE STATUS  
NAME                                ROLE    READY  
ip-172-20-35-90.ec2.internal       node    True  
ip-172-20-45-123.ec2.internal       node    True  
ip-172-20-53-104.ec2.internal       master  True  
  
Your cluster test-cluster.in is ready
```

11. In the next step, you need to create Amazon ECS (Elastic Container Services).

- a. Search for ECR (Elastic Container Registry) in services.



b. Enter a name and create the repository.



As you have seen in this demonstration, we will be using a public git repository available on the following [link](#). Before cloning the git repository, you will first have to install git. This can be done by executing the following commands.

a) First you have to install git using the following command

```
sudo yum install git
```

- b) The next step is to clone the codes into your instance. This is done using git clone command. For this step, you can perform the cloning by executing,

```
git clone https://github.com/antoinemertz/deploy-ml-flask.git
```

- c) Go into the particular folder to access the contents from the git repository

```
cd deploy-ml-flask/
```

Sometimes, there are issues with AWS while trying to use 'Load Balancer', hence you should add this additional code for attaching the elastic load balancing role

```
aws iam create-service-linked-role --aws-service-name  
"elasticloadbalancing.amazonaws.com"
```

Before starting the docker service, you will first need to install docker using the following command.

```
sudo yum install docker
```

Now you can start the docker service using,

```
sudo service docker restart
```

If you run the following command, you will find that
sudo docker images

Now let us build the docker image

```
sudo docker build -t insurance-app:v1 .
```

Validate that the image has been created by running **sudo docker images** again

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
insurance-app	v1	01accf7e20a4	About a minute ago	916MB
python	3.6.3	a8f7167de312	2 years ago	691MB


14. Authenticate to container Registry AWS

sudo \$(aws ecr get-login --no-include-email --region us-east-1)

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ sudo $(aws ecr get-login --no-include-email --region us-east-1)
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Copy the url from ECR

Repository name ▲	URI	Created at ▼	Tag immutability	Scan on push	Encryption type
 test-awsecr	 177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr	08/26/20, 12:52:49 PM	Disabled	Disabled	AES-256

```
sudo docker tag insurance-app:v1
177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:v1
```

Now that you have tagged it, the whole thing needs to be pushed. Hence, we will use sudo docker push followed by the url

```
sudo docker push
177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:v1
```

Once this is done, you will be able to see that it is pushing everything to the ECR

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ sudo docker push 177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr
The push refers to repository [177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr]
44035a629395: Pushed
68a8ebbf3f99: Pushed
cd3f32d47dfa: Pushing [=====>] 160.9MB/224.6MB
873b35128e79: Pushed
43cb5700b5b9: Pushed
c8c8418550a6: Pushed
56388ddcbf7f: Pushed
eddal28d7256: Pushing [=====>] 50.12MB/62.55MB
0870b36b7599: Pushed
8fe6d5dcea45: Pushing [==>] 16.3MB/323.9MB
06b8d020c11b: Pushing [=>] 3.833MB/123MB
b9914afd042f: Waiting
4bcdffd70da2: Waiting
```

Once this is done, you can go to your AWS ECR console to verify that your image has been pushed by finding your image under the image URL.

<input type="checkbox"/>	Image tag	Image URI	Pushed at ▼	Digest	Size (MB) ▼	Scan status	Vu
<input type="checkbox"/>	latest	177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:latest	08/26/20, 01:51:08 PM	sha256:23f61d1cd...	376.19	-	-

The next step is to deploy and manage the whole thing on the cluster. Hence we will use kubectl to create the deployment

```
kubectl create deployment insurance-app
--image=177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:v1
```

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ kubectl create deployment insurance-app --image=177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:latest
deployment.apps/insurance-app created
```

You will see that your insurance-app has been created

18. Expose your application to the internet

By default, the containers you run on are not accessible from the internet because they do not have external IP addresses. Execute the following code to expose the application to the internet:

```
kubectl expose deployment insurance-app --type=LoadBalancer --port 5000
--target-port 5000
```

So you will get the following result

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ kubectl expose deployment insurance-app --type=LoadBalancer --port 5000 --target-port 5000
service/insurance-app exposed
```

To find the url that needs to be used, type the following code

```
kubectl get services
```

You will get the following result.

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
insurance-app  LoadBalancer 100.71.62.89  a40fd2365931d4135ac5ba44984998e0-1408662421.us-east-1.elb.amazonaws.com 5000:32527/TCP 8s
kubernetes    ClusterIP     100.64.0.1    <none>         443/TCP          81m
```

Within this, you will be able to find an

```
[ec2-user@ip-172-31-47-90 deploy-ml-flask]$ kubectl get services
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
insurance-app  LoadBalancer  100.71.62.89    a40fd2365931d4135ac5ba44984998e0-1408662421.us-east-1.elb.amazonaws.com  5000:32527/TCP  8s
kubernetes    ClusterIP      100.64.0.1      <none>           443/TCP          81m
```

You can use '**kubectl describe services**' to check if we are getting any errors. If there is no error, your command will return a result having all the details of the cluster as shown below.

```
Name:          insurance-app
Namespace:     default
Labels:        app=insurance-app
Annotations:    <none>
Selector:      app=insurance-app
Type:          LoadBalancer
IP:            100.71.62.89
LoadBalancer Ingress: a40fd2365931d4135ac5ba44984998e0-1408662421.us-east-1.elb.amazonaws.com
Port:          <unset> 5000/TCP
TargetPort:    5000/TCP
NodePort:      <unset> 32527/TCP
Endpoints:     100.96.1.4:5000
Session Affinity: None
External Traffic Policy: Cluster
Events:        <none>

Name:          kubernetes
Namespace:     default
Labels:        component=apiserver
               provider=kubernetes
Annotations:    <none>
Selector:      <none>
Type:          ClusterIP
IP:            100.64.0.1
Port:          https 443/TCP
TargetPort:    443/TCP
Endpoints:     172.20.53.104:443
Session Affinity: None
Events:        <none>
```

Once your model has been deployed, let us move on to our local machine and try to run the model from there.

Let's open your jupyter notebook in your local machine. And paste the following code to predict the results from the iris dataset.

Use your ML model

Now you can use your model.

```
>>> import json
>>> import requests
>>> url = "http://127.0.0.1:5000/predict"
>>> data = json.dumps({'s1': [5.84, 4.38], 'sw': [3.0, 2.16], 'pl': [3.75, 7.65], 'pw': [1.1, 1.23]})
>>> r = requests.post(url, data)
>>> print(r.json())
```

Once you write the code, make sure to replace the ip address in the url with the external url generated from running the cluster. The point where this url needs to be inserted has been highlighted in red.

```
import json
import requests
url =
"http://a40fd2365931d4135ac5ba44984998e0-1408662421.us-east-1.elb.amazonaws
.com:5000/predict"
data = json.dumps({'s1': [5.84, 4.38], 'sw': [3.0, 2.16], 'pl': [3.75,
7.65], 'pw': [1.1, 1.23]})
r = requests.post(url, data)
print(r.json())
```

If you want to run the application on user interface then you need to follow the following steps:

You have been given all related files-

- requirements.txt
- Dockerfile
- server.py
- model.pckl
- Index.html

You need to add all of such files into a directory named **deploy-ml-flask** on EC2. You already know how to add files on EC2.

Once you have added all the required files on EC2 then follow the same steps as mentioned below that you are already aware of.

- Let's restart Docker.


```
sudo service docker restart
```

- Build an iris application

```
sudo docker build -t iris-app:v1 .
```

Validate that the image has been created by running **sudo docker images** again

- Authenticate to container Registry AWS

```
sudo $(aws ecr get-login --no-include-email --region us-east-1)
```

Copy the url from ECR

Repository name ▲	URI	Created at ▼	Tag immutability	Scan on push	Encryption type
test-awsecr	177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr	08/26/20, 12:52:49 PM	Disabled	Disabled	AES-256

```
sudo docker tag iris-app:v1  
177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:v1
```

Now that you have tagged it, the whole thing needs to be pushed. Hence, we will use **sudo docker push** followed by the url

```
sudo docker push  
177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:v1
```

Once this is done, you will be able to see that it is pushing everything to the ECR

Once this is done, you can go to your AWS ECR console to verify that your image has been pushed by finding your image under the image URL.

<input type="checkbox"/>	Image tag	Image URI	Pushed at ▼	Digest	Size (MB) ▼	Scan status	Vu
<input type="checkbox"/>	latest	177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:latest	08/26/20, 01:51:08 PM	sha256:23f61d1cd...	376.19	-	-

The next step is to deploy and manage the whole thing on the cluster. Hence we will use kubectl to create the deployment

```
kubectl create deployment iris-app
--image=177300670946.dkr.ecr.us-east-1.amazonaws.com/test-awsecr:v1
```

You will see that your iris-app has been created

- Expose your application to the internet

```
kubectl expose deployment iris-app --type=LoadBalancer --port 5000
--target-port 5000
```

- To find the url that needs to be used, type the following code

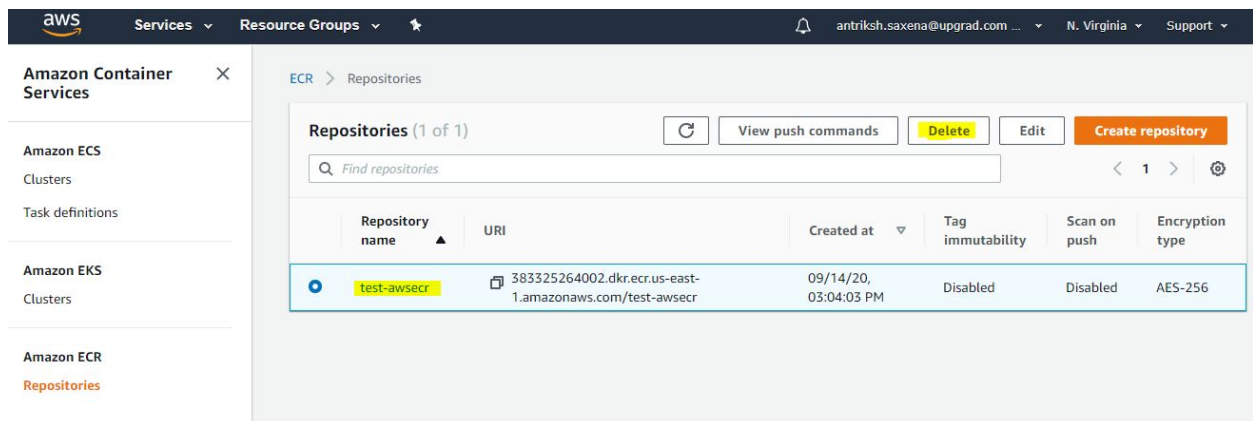
```
kubectl get services
```

TERMINATING THE CLUSTER:

One of the important steps to save the cost.

Once you have done with your entire process, it is very important to delete the entire cluster and delete the ECR repository and private hosted zones. It costs too high if you keep it running.

Deletion of ECR repository:




The screenshot shows the AWS Management Console interface for ECR Repositories. The left sidebar contains navigation links for Amazon Container Services, Amazon ECS, Amazon EKS, and Amazon ECR. The main content area displays the 'Repositories' page with a table listing the repository 'test-awsecr'. The table columns include Repository name, URI, Created at, Tag immutability, Scan on push, and Encryption type. The repository 'test-awsecr' is highlighted with a blue row.

Repository name	URI	Created at	Tag immutability	Scan on push	Encryption type
test-awsecr	383325264002.dkr.ecr.us-east-1.amazonaws.com/test-awsecr	09/14/20, 03:04:03 PM	Disabled	Disabled	AES-256

Deleting Private hosted zone

Hosted zones Info

How hosted zones work


 A hosted zone contains records that define how internet traffic is routed for a domain and its subdomains. For example, in the example.com hosted zone, you can create records for example.com and www.example.com that route traffic to a web server running on an EC2 instance or to an S3 bucket.

Hosted zones (1/2) ↻ View details Edit Delete Create hosted zone

< 1 > ⚙️

Domain name	Type	Created by	Record count	Description	Hosted zone ID
test-cluster.in	Private	Route 53	4	-	Z098249714 C28RI5JIDEV

Deleting Auto Scaling Group:



Services

Resource Groups

Volumes

Snapshots

Lifecycle Manager

▼ Network & Security

Security Groups New

Elastic IPs New

Placement Groups New

Key Pairs New

Network Interfaces

▼ Load Balancing

Load Balancers

Target Groups New

▼ Auto Scaling

Launch Configurations

Auto Scaling Groups

Resources ↻ ⚙️

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Running instances	3	Elastic IPs	0
Dedicated Hosts	0	Snapshots	1
Volumes	15	Load balancers	1
Key pairs	9	Security groups	30
Placement groups	0		

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

EC2 > Auto Scaling groups

Auto Scaling groups (2/2) Refresh Edit Delete Create an Auto Scaling group

Search your Auto Scaling groups

<input checked="" type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min
<input checked="" type="checkbox"/>	nodes.test-cluster	nodes.test-cluster.in-20200912091...	2	-	2	2
<input checked="" type="checkbox"/>	master-us-east-1b	master-us-east-1b.masters.test-clus...	1	-	1	1

Deleting the load balancer:

New EC2 Experience Tell us what you think ×

EC2 Dashboard New

Events New

Tags

Limits

▼ Instances

Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Resources Refresh Settings

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Running instances	3	Elastic IPs	0
Dedicated Hosts	0	Snapshots	1
Volumes	5	Load balancers	4
Key pairs	2	Security groups	15
Placement groups	0		

Create Load Balancer Actions

Filter by tags and attributes

<input checked="" type="checkbox"/>	Name	State	VPC ID	Availability Zones	Type
<input checked="" type="checkbox"/>	ab0a0119d40724159...	724159b57f99...	vpc-002ee582b5c8d5199	us-east-1b	classic
<input checked="" type="checkbox"/>	ab638b74541aa4222...	aa4222923e39...	vpc-002ee582b5c8d5199	us-east-1b	classic
<input checked="" type="checkbox"/>	adf1460b3ae424e47...	424e4799a321...	vpc-002ee582b5c8d5199	us-east-1b	classic
<input checked="" type="checkbox"/>	a9e8ac83558dd42f7...	dd42f7af93437...	vpc-002ee582b5c8d5199	us-east-1b	classic

Edit health check
 Edit subnets
 Edit IP address type
 Edit instances
 Edit listeners
 Edit security groups
 Edit attributes
 Delete

1 to 4 of 4

Resource Groups

phanendra.varma@upgrad.co...N. VirginiaSupport

Launch InstanceConnectActions

Filter by tags and attributes or search

1 to 13 of 13

	Name	Instance ID	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
	master-us-east-1	i-0077bf365bdb0...	us-east-1a	terminated		None	
	nodes.test-cluster	i-04a637b0a865ccb44	us-east-1a	starting-do...		None	ec2-54-89-64-135.com...
	nodes.test-cluster	i-0cb0113b8b45d8b08	us-east-1a	starting-do...		None	ec2-54-82-123-158.co...
	prateek	i-02eb231d9202...	us-east-1a	stopped		None	

Instances: i-04a637b0a865ccb44 (nodes.test-cluster.in), i-0cb0113b8b45d8b08 (nodes.test-cluster.in)

Description

Status Checks

Monitoring

Tags

- i-04a637b0a865ccb44: ec2-54-89-64-135.compute-1.amazonaws.com
- i-0cb0113b8b45d8b08: ec2-54-82-123-158.compute-1.amazonaws.com

- Connect
- Get Windows Password
- Create Template From Instance
- Launch More Like This
- Instance State
- Instance Settings
- Image
- Networking
- CloudWatch Monitoring
- Start
- Stop
- Stop - Hibernate
- Reboot
- Terminate