**EKS CLUSTER CREATION**

1. **Create IAM role For the Cluster:-**

Roles name: - **LTI-AAM-EKS-Role**

AmazonEKSServicePolicy

AWS managed policy

AmazonRoute53FullAccess

**2.) To Create VPC and Subnet:-**

**Create VPC**:- 10.0.0.0/16 - V-LTI-VPC

**Create Subnet**: -

Private1:- 10.0.1.0/24 - 1a - V-LTI-Private

Private2:- 10.0.2.0/24 - 1b - V-LTI-Private

Public :- 10.0.3.0/24 - 1a - V-LTI-Public

**Create RouteTable**:-

***R1:- V-LTI-RT-Public*** :-

Routes -> 10.0.0.0/16 (local)

Routes -> 0.0.0.0/0 - (IGW)

***R2:- V-LTI-RT-Private:-***

Routes -> 10.0.0.0/16 (local)

Routes -> 0.0.0.0/0 - (Nat Gateway)

**Create Internet gateway**:-

V-LTI--IGW :- Attach to our VPC

**Create Elastic IP:-** 3.213.153.84

**Create NatGateway**

V-LTI-NAT

Subnet - V-LTI-Public

Attach Elastic IP to NatGateway

**3.) Now create EKS Cluster through EKS Service**

Cluster Name: **LTI-EKS-AAM-Cluster-New**

Attach IAM role

Attach VPC

Select Security group

Below are the Configuration of Cluster: -

VPC - Select your VPC

Subnets – Select your Subnet

Security groups – select Security Group

API server endpoint access

Private access: - Disabled

Public access: - Enabled

**4.) To Create EC2 instance for the following step: -**

1. Choose AMI- Amazon Linux
2. Choose Instance Type:- t2.micro
3. Configure Instance :- Select VPC amd Public subnet
4. Add Storage
5. Add Tags :-
6. Configure Security Group :- Create a New security group, with SSH - Port - 22 / HTTP /HTTPS/ All TCP / All traffic
7. Review and Launch

**5.)** **Install Kubectl & awsIAMAuthenticator in Our EC2 Machine:-**

***Download and Install kubectl, follow the folling steps:-***

https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html

* curl -o kubectl <https://amazon-eks.s3-us-west-2.amazonaws.com/1.13.7/2019-06-11/bin/linux/amd64/kubectl>
* chmod +x ./kubectl
* mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
* echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
* kubectl version --short --client

***Download and Install awsIAmAuthenticator, follow the following steps:-***

https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html

- curl -o aws-iam-authenticator <https://amazon-eks.s3-us-west-2.amazonaws.com/1.13.7/2019-06-11/bin/linux/amd64/aws-iam-authenticator>

- chmod +x ./aws-iam-authenticator

- mkdir -p $HOME/bin && cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:$PATH

- echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

- aws-iam-authenticator help

**6.) To check Cluster install properly or Not in our EC2 machine**

**kubectl version --short –client -** To check Kubectl

**aws-iam-authenticator -**  To check iam authenticator

**aws –version -** To check aws Version

**aws eks update-kubeconfig --name LTI-EKS-AAM-Cluster-New**

**7.) To create WorkerNode**

Worker Node: - **LTI-EKS-WorkerNode**

we can create this through CloudFormation, use below link

URL: - https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-02-11/amazon-eks-nodegroup.yaml

https://docs.aws.amazon.com/eks/latest/userguide/eks-optimized-ami.html

Node Image ID: - ami-08c4955bcc43b124e

Added 3 subnet, while creating worker node.

Added ARN in auth file

**arn: aws:iam::796470167759:role/LTI-EKS-WorkerNode-J-NodeInstanceRole-OARIP7V06TTW**

**8.) create aws-auth-cm.yaml**

# **kubectl apply -f aws-auth-cm.yaml**

================================================================================

apiVersion: v1

kind: ConfigMap

metadata:

name: aws-auth

namespace: kube-system

data:

mapRoles: |

- rolearn: <ARN of instance role (not instance profile)> # Put WorkerNode ARN

username: system:node:{{EC2PrivateDNSName}}

groups:

- system:bootstrappers

- system:nodes

===============================================================================

**9.)kubectl get nodes** --- **to check attach to cluster**

**Note:- Not should be in Ready State.**

**10.) Kubectl get svc – we can view our services**

**Environment Details**

**Cluster Name:- LTI-EKS-AAM-Cluster-New**

**Worker Node Name:- LTI-EKS-WorkerNode**

**EC2 Instance Name:- LTI-AAM-LINUX-Machine**

**IAM Role Name :- LTI-AAM-EKS-Role**

**S3 Bucker Name:- ipfspoc**

**Deployment Process: -**

1. Download out Project File
2. Open .net solution file in Visual Studio
3. Enable Docker Support
4. Build Docker file and create the Image.
5. Open PowerShell: -

To Push Docker Images into Docker hub: -

***docker images***

***docker tag c7e53cdaa52d 9221484790/ipfs:v1***

***docker push 9221484790/ipfs:v1***

1. Login to Dockerhub and check your Image has been pushed or not
2. Connect to EC2 instance with; ***ec2-user***
3. Connect with root privileges:-

***sudo su –***

1. Edit Yaml file --- (attached Yaml File)
2. ***Kubectl apply -f webui.yaml***
3. ***Kubectl get svc***
4. To go into the container :- ***docker exec -it 9c014042c57d bash***
5. To Delete Yaml File:- ***Kubectl delete -f webui.yaml***

**Kubernetes Commands**

***kubectl get all --all-namespaces -o wide***

***kubectl get pod -o wide***

***kubectl get svc -o wide***

***==========================================================================***

**YAML File (Deployment / Service)**

apiVersion: apps/v1

kind: Deployment

metadata:

name: ipfsweb7

spec:

selector:

matchLabels:

app: ipfsweb7

replicas: 1

template:

metadata:

labels:

app: ipfsweb7

spec:

containers:

- name: ipfsweb7

image: 9221484790/ipfswebui:v7

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: ipfsweb7

spec:

type: NodePort

selector:

app: ipfsweb7

ports:

- name: client

protocol: TCP

port: 80

nodePort: 32085

selector:

app: ipfsweb7

type: LoadBalancer

===========================================================================

**Referred URL**

1. https://www.youtube.com/watch?v=6H5sXQoJiso -- youtube URL to Deploy nginx application through EKS
2. https://kubernetes.io/docs/tasks/tools/install-kubectl/ -- official link for kubernetes
3. https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html -- Amzon EKS Documentation
4. https://www.edureka.co/blog/amazon-eks/ -- Step wise procedure to deploy nginx application