<https://github.com/dockersamples/example-voting-app/blob/main/architecture.excalidraw.png>

<https://aws.amazon.com/cli/>

Why k8s in place of docker-swarm?

Docker-swarm is proprietary means, only community edition is free , for organization level usage need to pay whereas k8s is an open source and free.

Docker swarm only supports docker as runtime, but k8s supports all container runtime.

Follows master-slave architecture.

We use API’s to communicate with the controller.

5 things installed in k8s controller/manager:

Cm/ccm:

Manages all the communications happening between different components

within the cluster .

Cm (controller manager) :

If you are using kubernetes on -premises the cm will be active.

Ccm (Cloud controller manager)

If you are using k8s over the cloud ccm will be active.

Etcd

Etcd is key value pair and responsible for remembering the state of the cluster. It stores the information about the cluster.

Scheduler

Whenever a request comes from user , controller will look into the etcd for the cluster information and the scheduler will schedule the tasks to run any of the worker nodes. Scheduler will decide where to run the request. Scheduler will issue the instructions to the kubelet, kubelet will respond back with the status and etcd will be updated.

Api server

API’s used to communicate with the controller. To call the API’s or to communicate using the API’s we use kubectl which we install on our local. We will install the kubectl in our local machine and call the API’s which are hosted in the API Server on the controller.

3 things installed on every node:

Container

Kubelet

Kubelet is the component which is the captain of the node which translates the native runtime in every nodes. Since k8s works with all container runtimes if a request receives for any runtime kubelet handles the translation into the respective container runtimes. Controller will always communicate to Kubelet.

Kubeproxy

Steps to create k8s cluster:

1. Create 2 IAM roles for each nodes one for cluster and one for worker nodes.

A screenshot of a computer

Description automatically generated

A white background with black and white clouds

Description automatically generated

Click create role -> AWS services

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

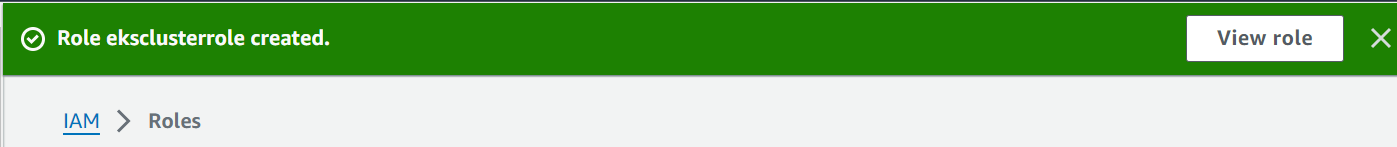
Description automatically generated

A screenshot of a computer

Description automatically generated

A close up of a box

Description automatically generated



To create second role in use case follow below:

A screenshot of a computer

Description automatically generated

Need to select 3 policies:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

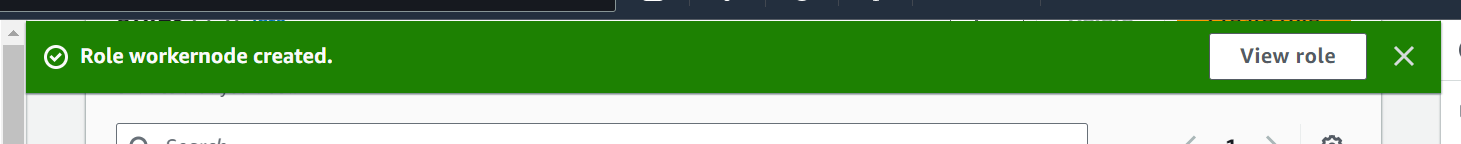
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

1. Create eks

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a sign

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

After the above step all other pages use default click next and review and create.

A screenshot of a computer

Description automatically generated

Hence we created the cluster or controller which will have all the 5 components. Will take 15-20 min to create the cluster. Once cluster created then will add worker nodes in the cluster.

A screenshot of a computer

Description automatically generated

1. Install AWS CLI to communicate.

Install cli on our laptop.

<https://aws.amazon.com/cli/>

A close-up of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

1. Create access keys/tokens to establish connection between aws cli and aws.

Go security credentials -> create access keys.

Use the keys to configure aws cli as below:

A black screen with white text

Description automatically generated

1. Install Kubectl

<https://kubernetes.io/>

go to documentation -> install kubectl

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Download the highlighted version and give the path where it is installed to the environment variable

PATH.

A screenshot of a computer

Description automatically generated

1. Create worker nodes in the eks cluster:

Go to cluster and in compute:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error message

Description automatically generated

Click default next and review and create. Once done we added the worker node to the cluster.

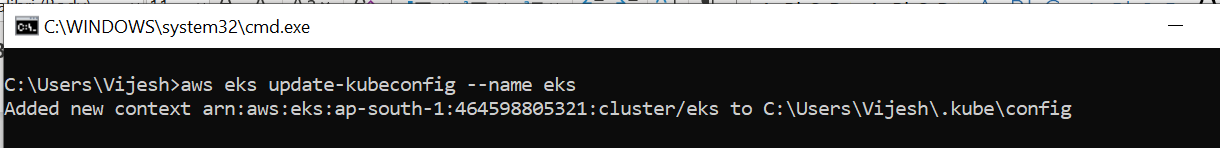
A screenshot of a computer

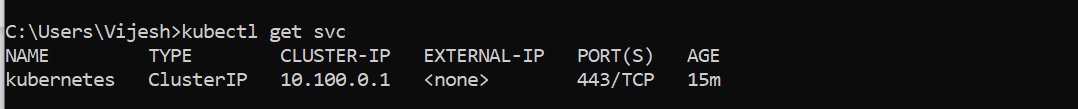
Description automatically generated

A screenshot of a computer

Description automatically generated

1. Configure kubectl to configure with your cluster





The above shows that the Kubernetes cluster is successfully configures and created and the kubectl is also configured to communicate with the cluster.

aws eks update-kubeconfig --name <cluster-name> -- region <regionname>

aws eks update-kubeconfig --name 2048-demo-cluster --region us-east-1

In the above project loadbalancer is not created by default as part of cluster, it is automatically created when we create a service as type loadbalancer.