<https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html>

<https://medium.com/the-programmer/aws-eks-fundamentals-core-components-for-absolute-beginners-part1-9b16e19cedb3>

<https://medium.com/the-programmer/kubernetes-fundamentals-for-absolute-beginners-architecture-components-1f7cda8ea536>

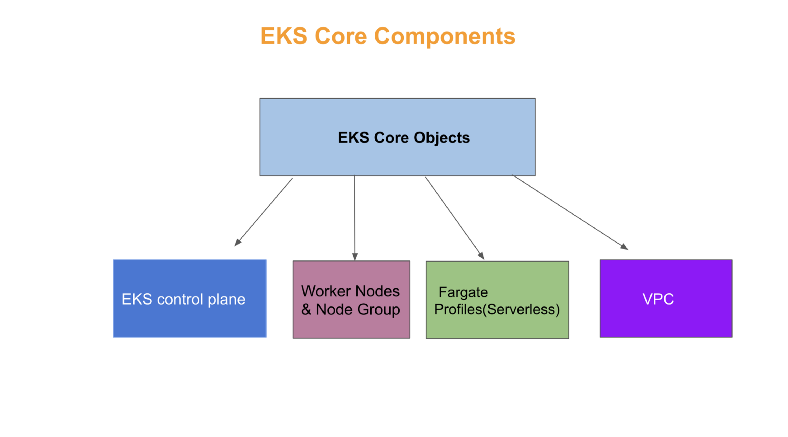
managed service that runs without installing, operating, and maintaining your own Kubernetes control planes or nodes.

Capabilities:

1. Amazon ECR for container images
2. Elastic load balancer for load distribution
3. Authentication
4. Amazon VPC for isolation

Control Plane Architecture:

* Control Plane Architecture is not shared across clusters or AWS accounts
* Available across multiple availability zones
* Zero time upgrades , patching
* EKS supports Fargate to automatically provision on-demand serverless compute for your application.



Diagram

Description automatically generated

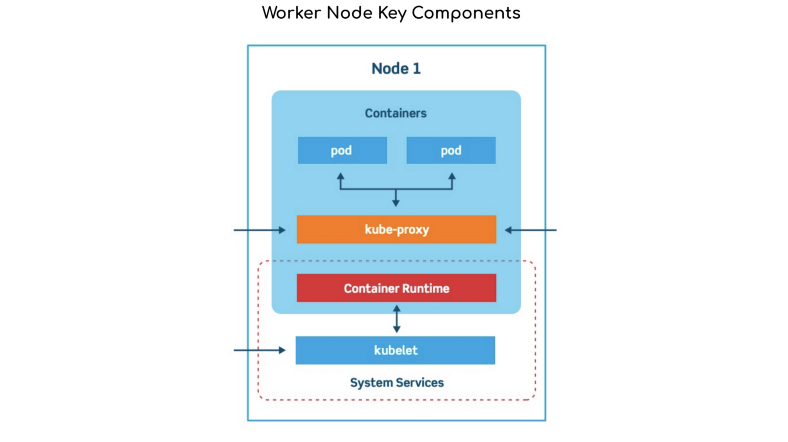
Control Plane :

* Control Plane act as masternode for Kubernetes architecture
* Responsible for mange the worker / slave nodes efficiently . They interact with Worker node to
  + Schedule the PODS
  + Monitor the worker nodes/PODS
  + Start and restart the PODS
  + Manage the new worker nodes joining the cluster
* Kubernetes API is exposed through Amazon EKS endpoint associated with cluster

MAsterNode process :

* Every masternode in the cluster runs the following key processes
  + Kube-apiserver
  + Kubectl-kube-controller-manager
  + Kube-scheduler
  + Etcd

EKS Worker Nodes:



Amazon EKS hosts worker nodes :

* All the workloads ( pod /deployments /service ) deployed in form of yaml file.
* Nodes are EC2 provided by AWS
* Each Amazon EC2 instance is deployed in one subnet
* Each node is assigned one private address from CIDR block assigned to each subnet
* WE often deal with worker nodes

*If a subnet is created using Amazon EKS-provided AWS Cloud formation templates, then the node deployed to the public subnet are automatically assigned to public IP address by subnet.*

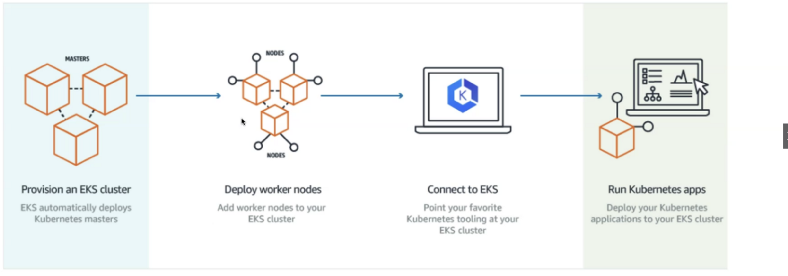
Worker Node has PODS

PODS are abstraction of containerized application

* Every worker runs these 3 key process
  + Container Runtime
  + Kubelet
  + Kube-proxy
* Instead of EC2 nodes we can use AWS Fargate , our workloads are deployed in private subnet only .
* Each pod is assigned a private IP address from the CIDR block assigned to the subnet. Fargate does not support all pod networking options

EKS – VPC

* Used to secure POD
* Compute resources will not shared with other customers



Provision an EKS cluster:

* EKS automatically deploy the master node
* This master node will provide required controle plane for managing networking and resource requirements of worker nodes

Deply worker nodes:

We need to provide worker nodes in cluster ,which is EC2 instances

Connect to EKS :

* IN order to deploy our workload to worker nodes , we need to connect to EKS . we rely on Kubectl 🡪 used to create and deploy workloads ( yaml file )
* Once workloads are deployed in worker nodes ,it is up and ready to exposed outside the cluster through service (Node Port ) of consumed within cluster

Installing EKS Cluster on AWS using eksctl :

* Eksctl is single command line utility for creating and managing Kubernetes cluster in Amazon AWS

Prerequisites:

* Installation and configuring AWS CLI
* Kubectl – command line for working with Kubernetes cluster
* Eksctl – command line for working with EKS cluster that automates many individual task
* Configure IAM configuraton : Amazon EKS IAM roles and service linked roles , AWS Cloud formation ,VPC and related resources

Install AWS CLI

$ curl "<https://awscli.amazonaws.com/AWSCLIV2-2.0.30.pkg>" -o "AWSCLIV2.pkg"$ sudo installer -pkg AWSCLIV2.pkg -target /

Verify suing aws –version

$ aws configureAWS Access Key ID [None]: ABCDEFGHIAZBERTUCNGG (Replace your access key when prompted)

AWS Secret Access Key [None]: uMe7fumK1IdDB094q2sGFhM5Bqt3HQRw3IHZzBDTm (Replace your secret key when prompted)

Default region name [None]: ap-south-1(you can put your own availability zone here )

aws ec2 describe-vpcs – to verify the aws cli working after configuring IAM security cred



Install Kubernetes

curl -o kubectl <https://amazon-eks.s3-us-west-2.amazonaws.com/1.21.2/2021-07-05/bin/darwin/amd64/kubectl>

Apply permission to binary

$chmod +x ./kubectl

Copy the binary to a folder in your PATH. If you have already installed a version of, then we recommend creating a $HOME/bin/kubectl and ensuring that $HOME/bin comes first in your $PATH.

$mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH

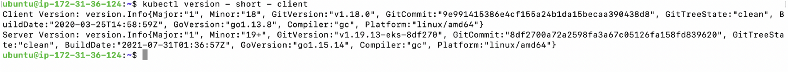
Add the $HOME/bin path to your shell initialization file so that it is configured when you open a shell.

$echo ‘export PATH=$PATH:$HOME/bin’ >> ~/.bash\_profile

After you install kubectl, let’s check the Kubectl version

$kubectl version — short — client

That means you have successfully installed Kubectl CLI .



$ brew install weaveworks/tap/eksctl

$ brew upgrade eksctl && brew link --overwrite eksctl

Create EKS Cluster

* We will make use of eksctl CLI to create our EKS cluster with the command as shown below:
* Create a cluster with the Amazon EKS latest Kubernetes version in your default Region. Replace the <example-values> (including <>) with your own values. You can replace <1.20> with any supported version.

$ eksctl create cluster \  
 --name Democluster\  
 --version <1.21> \  
 --without-nodegroup

* Check if the cluster is created:

$ eksctl get cluster

