EKS WorkShop Summary 👍

- Kubernetes Architecture - 10 m
- Give Introduction to EKS with Below Doc : - 10 m
- Different Ways to Create EKS [Console/Terraform/Eksctl] -10m
- Brief the Basics Master Plane & Explain NodeGroups -20m
- Demo case the EKS cluster creation via Console -30 m
  ==========
- Give Access to a ReadOnly EKS cluster - SSH access - 5 m - **Worksop**
- Demo case the kubectl commands - 20 m - **Worksop**
- Troubleshoot scenario -10m - **Worksop**
  ==========

  Part 2
- Custom NodeGroups AMI -30m
- Monitoring Basics via EFK -20m
- Security Basics -10m
- Brief HA/Cluster Autoscale/3rd party - 20m

Ref:
https://dev.to/rajitpaul_savesoil/read-only-access-to-specific-resources-in-aws-eks-cluster-via-eks-authentication-authorization-2i13
https://antonputra.com/kubernetes/add-iam-user-and-iam-role-to-eks/#add-iam-user-to-eks-cluster

What Is AWS EKS?

If you want to manage production-grade deployments, start, run, and scale the same on AWS Cloud or on-premises, Amazon's Elastic Kubernetes services(EKS) can help you achieve the same.

Amazon EKS is a managed service that makes it easy for you to run Kubernetes on AWS without needing to install and operate your own Kubernetes control plane or worker nodes.

Why EKS?

High Availability :

When you set up EKS on AWS, it gives you a control plane that is available across multiple availability zones, if there is an issue with any of the **control planes** EKS automatically detects and replaces those unhealthy control plane nodes, and provides on-demand, zero downtime upgrades, and patching.

> EKS offers a 99.95% uptime SLA. At the same time, the EKS console provides observability of your Kubernetes clusters so you can identify any issue quickly and get it resolved.

Provision Your Resources For Scale:

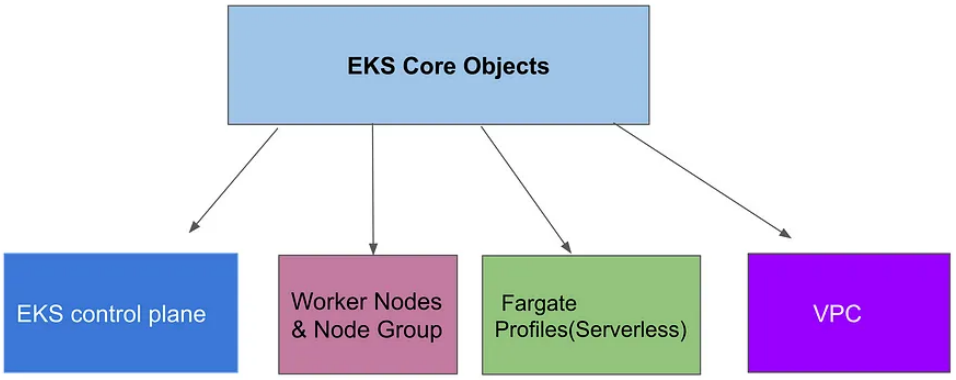
EKS managed services come pre-configured with the required compute (Server resources)provisioning which is designed to scale your K8S app. You don't need to manage those configurations manually.

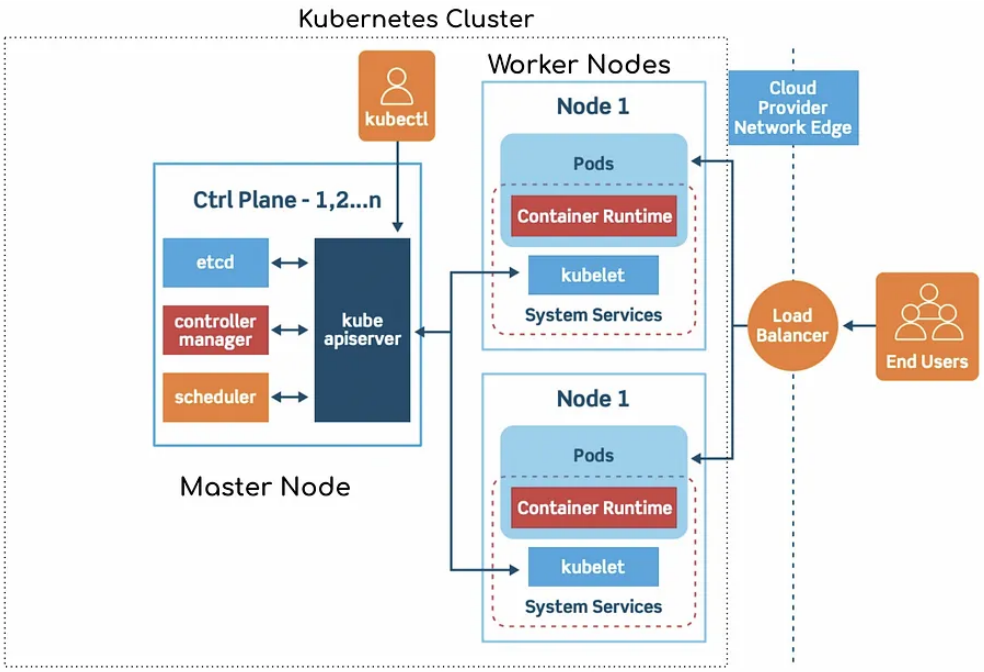
> EKS also supports AWS Fargate to automatically provision on-demand serverless compute for your applications.

Highly Secure K8s Environment :

The Clusters deployed using EKS is highly secured and automatically apply the latest security patches to your **cluster's control plane**.

EKS Core Components



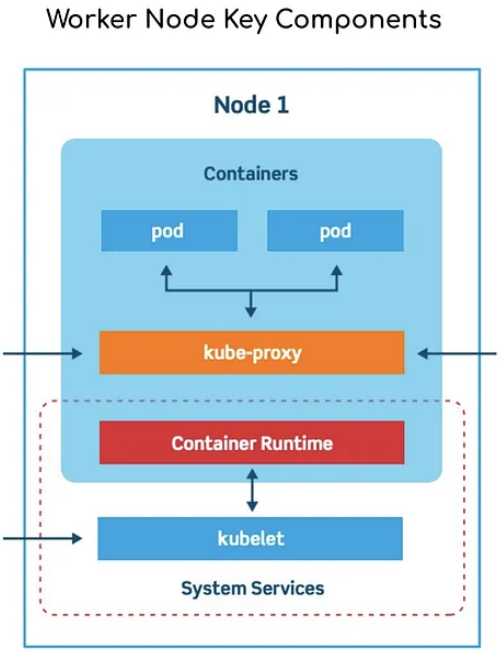


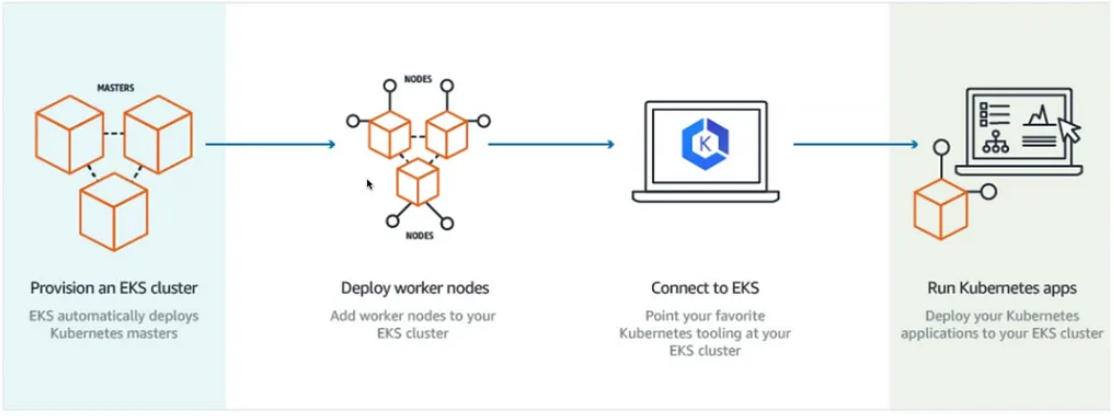
The **master node** is also known as a **control plane** that is responsible to manage worker/slave nodes efficiently. They interact with the worker node to

- Schedule the pods
- Monitor the worker nodes/Pods
- Start/restart the pods

- Manage the new worker nodes joining the cluster
- That being said, certain system components, such as the kube-proxy, may run as pods on the master node. The kube-proxy is responsible for forwarding network traffic to the appropriate pods and services in the cluster. It typically runs as a DaemonSet, which ensures that one instance of kube-proxy runs on each node in the cluster, including the master nodes.
- The EKS control plane is not deployed in the customer's account or VPC. Instead, it is deployed and managed by AWS in an AWS-owned account and VPC. When a new EKS cluster is created, a new control plane is provisioned for the cluster in the AWS account and VPC associated with the cluster.
- The customers interact with the EKS control plane through the Kubernetes API server, which is exposed as an endpoint in the customer's VPC.

The control plane runs in an account managed by AWS, and the Kubernetes API is exposed via the Amazon EKS endpoint associated with your cluster. Each Amazon EKS cluster control plane is single-tenant and unique and runs on its own set of Amazon EC2 instances.





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

- **Kube-apiserver** - The Kubernetes API server is the central component of the Kubernetes control plane that exposes the Kubernetes API, which is a REST [POST/GET]ful API that allows users to manage the Kubernetes cluster. It receives API requests from the kubectl command-line tool, the Kubernetes web UI, and other Kubernetes components, such as the kubelet and kube-proxy.
- **kubectl: kube-controller-manager -** The Kubernetes Control Manager (KCM) is a component of the Kubernetes control plane that is responsible for managing the cluster's desired state and responding to changes in the state of the cluster. [Node Controller/Replication Controller/Deployment Controller etc..]
- **Kube-scheduler -**The Kubernetes Scheduler is a component of the Kubernetes control plane that is responsible for assigning workloads to nodes in the cluster. It watches for new pods that have been created in the Kubernetes API server and selects a suitable node for them to run on, based on various constraints and policies.
- **Etcd -** etcd is a distributed, highly available, and consistent key-value store that is used as a data store for Kubernetes. It is the primary data store for Kubernetes and is used to store and retrieve configuration data, state data, and metadata for the cluster.