**Kubernetes**

EKS add-ons are pre-packaged, managed software components (like CoreDNS, kube-proxy, VPC CNI) that are essential for a Kubernetes cluster to function correctly. They simplify cluster management by letting AWS handle the installation, version updates, and security patches. This ensures compatibility with EKS versions and reduces manual configuration errors

Here’s a breakdown of each EKS add-on—CoreDNS, kube-proxy, VPC CNI, and Metrics Server.

**1.CoreDNS:**

CoreDNS is the DNS server for Kubernetes that translates service into IP addresses. It runs as a Deployment in the kube-system namespace and communicates with the Kubernetes API to discover services and endpoints. When a pod makes a DNS request, CoreDNS intercepts it and resolves it to the correct cluster IP. CoreDNS ensures pods can easily discover each other by name, which is essential for service-to-service communication.

**2.kube-proxy:**

kube-proxy manages network rules on each node to enable communication between services and pods. It uses iptables (or IPVS) to route traffic to the correct backend pod for a given Kubernetes service. kube-proxy watches the Kubernetes API for changes to Services and Endpoints and updates routing rules accordingly. It ensures that when a pod tries to access a ClusterIP or NodePort, traffic is routed efficiently and correctly. kube-proxy is crucial for maintaining consistent service behavior across the cluster.

**3.VPC CNI (VPC Container network interface)**:

VPC CNI plugin allows each pod to receive an IP address from the VPC, enabling native VPC networking. This means pods can directly communicate with other AWS services, EC2 instances, and the internet without NAT or overlay networks. The CNI plugin runs as a DaemonSet on each node and interfaces with the EC2 instance metadata and AWS APIs to allocate and attach ENIs (Elastic Network Interfaces). It communicates with the Kubernetes API to manage pod network attachment. VPC CNI provides high-performance pod networking and simplifies security group and routing setup in AWS.

**4.Metrics Server**:

The Metrics Server collects resource usage data—like CPU and memory—from the Kubelet on each node. It aggregates this information and makes it available to the Kubernetes API for use by tools like the Horizontal Pod Autoscaler (HPA). It runs as a Deployment in the kube-system namespace and communicates securely with the API server and Kubelets over HTTPS. Unlike Prometheus, Metrics Server stores only recent metrics in memory and is intended for lightweight, short-term monitoring. It plays a key role in enabling auto-scaling based on live resource usage.

**Horizontal Pod Autoscaler (HPA)**

Horizontal Pod Autoscaler (HPA) is a Kubernetes feature that automatically scales the number of pods in a deployment (or replica set or stateful set) based on observed CPU utilization (or other select metrics).

**This is crucial for:**

Handling varying loads efficiently

Maintaining performance during spikes

Saving resources during idle times

**How HPA Works**

Monitors Metrics (like CPU, memory) via metrics-server

Adjusts the replicas count in your deployment dynamically

Example: If CPU usage goes beyond 50%, HPA might increase replicas from 1 to 3 to balance the load.

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

labels:

app: nginx

spec:

replicas: 1

selector:

matchLabels:

app: nginx

template:

metadata:

name: nginxpod

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

resources:

limits:

cpu: 10m