Amazon EKS Cheat Sheet

- A managed service that allows you to run Kubernetes on AWS without installing, operating, or maintaining your own Kubernetes control plane or nodes.
- Integration with various AWS services to provide scalability and security for your applications:
    - Amazon ECR for container images
    - Elastic Load Balancing for load distribution
    - IAM for authentication
    - Amazon VPC for isolation

Amazon EKS Components

- Clusters
    - An EKS cluster is made up of two main components:
        - EKS control plane
            - It is made up of nodes that run the Kubernetes software (API server &).
            - Each cluster is single-tenant and unique, and runs on its own set of EC2 instances.
            - Cluster control plane is provisioned across multiple AZs and fronted by an ELB Network Load Balancer.
            - Use AWS KMS to encrypt data stored by nodes and associated EBS volumes.
        - EKS nodes
            - A cluster consists of one or more EC2 nodes on which pods are scheduled.
            - Connects to the cluster's control plane via the API server endpoint.
    - The API server endpoint is public to the internet by default, but you can enable private access to keep communication between nodes and the API server within the VPC.
    - EKS supports two autoscaling products:
        - Cluster Autoscaler – uses AWS Auto Scaling groups.
        - Karpenter – works directly with the Amazon EC2 Fleet.
    - By default, cluster control plane logs aren't sent to CloudWatch Logs. In order to send logs for your cluster, you have to enable each log type individually.

    - EKS cluster uses IAM / OIDC for authentication and Kubernetes RBAC for authorization.
- Nodes
    - Nodes must be in the same VPC as the subnets you chose when creating a cluster.
    - From the perspective of the Kubernetes API, nodes represent the compute resources provisioned for your cluster.
    - Taints and tolerations prevent pods from being scheduled on the wrong nodes.
    - Self-managed nodes
        - A cluster can have several node groups.
        - A node group is a collection of one or more EC2 instances deployed in an Amazon EC2 Auto Scaling group.
        - In a node group, instances must have the following characteristics:
            - Same instance type
            - Running the same AMI
            - Uses the same EKS node IAM role
        - Node groups with different instance types and host operating systems can exist in a cluster.
        - There are two methods for updating self-managed node groups in a cluster to use a new AMI:
            - Migrating to a new node group
            - Updating an existing self-managed node group
    - Managed node groups
        - Automates the provisioning and lifecycle management of nodes in EKS clusters.
        - Every managed node is provisioned as part of Amazon EC2 Auto Scaling group.
        - When nodes are launched as part of a managed node group, they are automatically tagged for auto-discovery by Kubernetes Cluster Autoscaler.
        - Use node group to apply Kubernetes labels to nodes.
        - Multiple managed node groups can exist in a single cluster.
        - When you create a managed node group, you have the option of selecting On-Demand or Spot instances.
        - To ensure that your applications remain available, node updates and terminations drain nodes automatically.
    - AWS Fargate
        - You must first define a Fargate profile before scheduling pods on Fargate in your cluster.
        - If a pod matches more than one Fargate profile, Amazon EKS picks one at random.

        - Fargate profiles are immutable and contains the following components:
            - Pod execution role
            - Subnets
            - Selectors
            - Namespace
            - Labels
        - Fargate runs only one pod per node.
        - Pod storage is ephemeral, and data is encrypted with AWS Fargate managed keys.
        - To encrypt ephemeral pod storage, you can use AWS Fargate managed keys.
- Workloads
    - Workloads are deployed in containers and define the applications that run on a Kubernetes cluster
    - A pod can contain one or more containers.
    - Vertical Pod Autoscaler adjusts your pods' CPU and memory reservations.
    - Horizontal Pod Autoscaler adjusts the number of pods in a deployment, replication controller, or replica set based on CPU utilization.
- EKS Connector
    - Enables you to register and connect any Kubernetes cluster to AWS.
    - You can view the status, configuration, and workloads of the cluster in the Amazon EKS console after it has been connected.

Amazon EKS Storage

- Container Storage Interface (CSI) enables third-party storage providers to create and deploy plugins in Kubernetes that provide alternative storage systems without modifying the core Kubernetes code.
- Amazon EBS CSI driver
    - The lifecycle of persistent volumes, such as EBS volumes, is handled by EKS clusters.
    - To make calls to AWS APIs, the EBS CSI plugin requires IAM permissions.
    - Although the Amazon EBS CSI controller can be run on Fargate, volumes cannot be mounted to Fargate pods.
    - You can also manage the EBS CSI driver as an EKS add-on.
- Amazon EFS CSI driver
    - EKS clusters manage the EFS file system lifecycle.
    - Container images based on Windows are incompatible with the EFS CSI driver.

    - Fargate nodes only support static provisioning.
    - A pod running on Fargate automatically mounts an EFS file system.
- Amazon FSx for Lustre CSI driver
    - EKS clusters can also manage the lifecycles of FSx file systems.
    - Fargate does not support the Lustre CSI driver.
- Amazon FSx for NetApp ONTAP CSI driver
    - A storage service for fully-managed ONTAP file systems in the cloud.

Amazon EKS Networking

- There are three ways to create a VPC for an EKS cluster:
    - Private subnets
        - Three private subnets are distributed across different AZs.
        - Nodes have the option of sending and receiving internet traffic via a NAT instance or NAT gateway.
        - The cluster endpoint can only be accessed via your VPC. Traffic from worker nodes to the endpoint will remain within your VPC.
    - Public subnets
        - Three public subnets are distributed across different AZs.
        - Nodes are assigned public IPv4 addresses by default and can send and receive internet traffic via an internet gateway.
        - The cluster endpoint can be accessed from outside your VPC. Traffic from worker nodes will leave your VPC to connect to the endpoint.
    - Public and private subnets
        - Each AZ has one private and public subnet.
            - Nodes are deployed to private subnets.
            - Load balancers are assigned to public subnets to load balance traffic to pods running on nodes.
        - Public IPv4 addresses are automatically assigned to nodes deployed in public subnets.
        - IPv6 addresses can be assigned to nodes in both public and private subnets.
        - A NAT gateway (IPv4) or an egress-only Internet gateway (IPv6) can be used to allow pods to communicate outbound to the internet.
        - The cluster endpoint can be accessed from outside your VPC. Traffic from worker nodes to the endpoint will remain within your VPC.
- The cluster security group manages communication between the control plane and the cluster's compute resources (worker nodes and Fargate pods).
- You can use AWS PrivateLink to privately access the management APIs of Amazon EKS from within your VPC.
- Pod networking

    - Container Network Interface (CNI) is a plugin that assigns a private IPv4/IPv6 address from VPC to each pod.
    - VPC CNI plugin is deployed to each EC2 node in a Daemonset under the name and consists of two components:
        - L-IPAM daemon
            - Creates and attaches network interfaces to EC2 instances.
            - Assigns secondary IP addresses to network interfaces.
            - Maintains a warm pool of IP addresses that will be assigned to pods on each node.
        - CNI plugin
            - Configures the host network and adds the correct network interface to the pod namespace.
    - You can't assign both IPv4 and IPv6 addresses (dual-stacked) to pods and services.
    - With security groups for pods, you can control the inbound and outbound network traffic to and from your pods.
    - Attach multiple network interfaces to a pod using the Multus CNI plugin.
    - CNI metrics helper is a tool that allows you to:
        - Scrape network interface and IP address information.
        - Aggregate metrics at the cluster level.
        - Publish the cluster's CNI metrics to CloudWatch.
- AWS Load Balancer Controller
    - In charge of managing AWS Elastic Load Balancers in a Kubernetes cluster and provisions the following load balancers:
        - ALB when you create a Kubernetes .
        - NLB when you create a Kubernetes service of type .
- CoreDNS
    - A DNS service within EKS clusters that allows individual containers to easily discover and connect to other containers in the cluster.
    - By default, two replicas of the CoreDNS image are deployed to an EKS cluster.
- Kube-proxy
    - Maintains network rules on each Amazon EC2 node.
    - Enables network communication to pods from network sessions inside/outside of the cluster.
- Calico
    - A network policy engine to implement network segmentation and tenant isolation.
    - Pod selectors and labels can be used to assign network policies to pods.

Amazon EKS Security

- By default, IAM users and roles do not have permission to create or modify Amazon EKS resources. An IAM administrator must first create IAM policies and attach them to the IAM users or groups that require those permissions.
- In the Amazon EKS control plane, the IAM user or role that creates the cluster is automatically granted permissions in the cluster's RBAC configuration.
- To grant additional AWS users or roles access to a cluster, edit the within Kubernetes and create a Kubernetes or with the name of a group specified in the .
- A service-linked role is predefined by Amazon EKS and includes all of the permissions that the service requires to call other AWS services. You can use this roles for:
    - EKS clusters
    - EKS node groups
    - EKS Fargate profiles
    - EKS cluster connector
- Before you can create an EKS cluster, you must have an IAM role with the policy: .
- The EKS node kubelet daemon makes calls to AWS APIs. When creating nodes, you will need to have an IAM role with the following IAM policies:
    - AmazonEKSWorkerNodePolicy
    - AmazonEC2ContainerRegistryReadOnly
    - AmazonEKS_CNI_Policy (IPv4) or IPv6 policy
- In order to run pods on AWS Fargate, you need to attach the Amazon EKS pod execution role.
- To view a Kubernetes cluster to Amazon EKS, you will need to create an Amazon EKS connector IAM role.
- You can use or the AWS Management Console to create an OIDC provider for your cluster in order to use IAM roles for service accounts.
- You can enable envelope encryption of Kubernetes secrets using AWS KMS.
- The AWS Secrets and Configuration Provider (ASCP) can be used to display secrets from AWS Secrets Manager and parameters from AWS Systems Manager Parameter Store as files mounted in Amazon EKS pods.

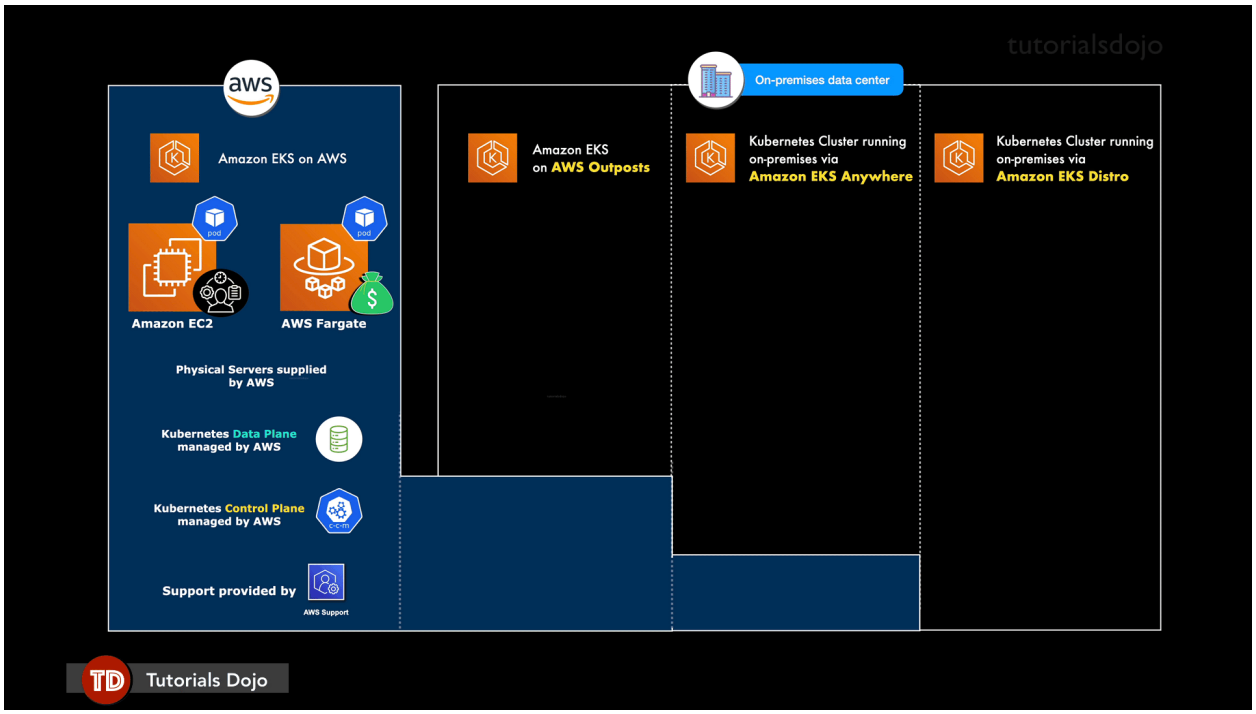
Amazon EKS Monitoring

- Amazon EKS control plane logging provides audit and diagnostic logs directly to Amazon CloudWatch Logs.
    - API server (kube-apiserver)
    - Audit (kube-apiserver-audit)
    - Authenticator (authenticator)
    - Controller manager (kube-controller-manager)
    - Scheduler (kube-scheduler)

- Logs are sent as log streams to a group in Amazon CloudWatch for each Amazon EKS cluster.
- Amazon EKS is integrated with AWS CloudTrail, and all API calls are recorded as events.
- Each event or log entry includes information about who initiated the request:
    - Root or AWS IAM user credentials.
    - Temporary security credentials for a role or Federated user
    - AWS service

Amazon EKS Deployment Options

- You can deploy your Kubernetes cluster in various ways in AWS and can include additional networking add-ons to improve your containerized architecture.
- A Kubernetes container can be deployed via the following options:
    - Amazon EKS cluster in your AWS account
    - Amazon EKS on AWS Outposts
    - Amazon EKS Anywhere
    - Amazon EKS Distro.
- The first option allows you to launch a Kubernetes cluster using managed or self-managed Amazon EC2 nodes that you can customize and control. You can also choose to deploy your Kubernetes pods on AWS Fargate to make the cluster serverless and extremely cost-effective.



Amazon EKS On AWS Outposts

- This is a deployment option that uses a physical AWS Outpost rack on your on-premises network to run your Kubernetes workloads. The data plane is also located on-premises, so you can have more control compared with running it exclusively in AWS.

Amazon EKS Anywhere

- Using Amazon EKS Anywhere is another way to deploy your containers on-premises. It works like Amazon ECS Anywhere, which allows you to run your containerized cluster entirely on your own. This means that the hardware, app deployment location, control plane, and data plane are all controlled on your own physical network. This gives you extensive control over all the components of your containerized application suite while maintaining official support from AWS.

Amazon EKS Distro

- The other deployment option that you can choose is Amazon EKS Distro. The word “distro” simply refers to the distribution of the same open-source Kubernetes software deployed by Amazon EKS in the AWS cloud. Amazon EKS Distro follows the same Kubernetes version release cycle as Amazon EKS and is provided to you as an open-source project that you can deploy on your own computer or on-site environment. It’s similar to the Amazon EKS Anywhere option, except that it does not include support services offered by AWS.

Amazon EKS Pricing

- For each Amazon EKS cluster you create, you are charged an hourly rate.
- You are charged for the AWS resources that you create to run Kubernetes worker nodes in Amazon EC2 with Amazon EKS managed node groups.

- In Amazon EKS on AWS Fargate, you are charged for the vCPU and memory resources.
- Amazon EKS on AWS Outposts charges an hourly rate for EKS clusters deployed in the cloud, but there is no additional charge for Kubernetes worker nodes running on Outposts EC2.