AWS Elastic Kubernetes Service (EKS) Detailed Notes

Overview:

- **Amazon EKS (Elastic Kubernetes Service)** is a fully managed Kubernetes service that helps run Kubernetes applications in the AWS cloud or on-premises using **AWS Outposts**.
- EKS provides a highly available, scalable, and secure Kubernetes control plane while enabling you to manage containerized applications using Kubernetes APIs and tools.
- It integrates with AWS services like **IAM**, **CloudWatch**, **AWS VPC**, **Elastic Load Balancing (ELB)**, **Auto Scaling**, and **ECR** to manage the infrastructure for Kubernetes.

Key Concepts in EKS and Kubernetes:

1. **Kubernetes Control Plane (Managed by AWS):**
    - AWS manages the **Kubernetes control plane**, including the API server, etcd (the key-value store), and controller manager.
    - The control plane is automatically distributed across multiple AWS Availability Zones (AZs) to ensure high availability.
    - AWS handles the provisioning, scaling, and patching of the control plane, ensuring it runs smoothly.
2. **Worker Nodes (Managed by You):**
    - **Worker nodes** are the EC2 instances or Fargate profiles where Kubernetes pods run.
    - You can either manage the worker nodes by launching and maintaining a set of EC2 instances or use **AWS Fargate** for a serverless experience where AWS handles the underlying infrastructure.
    - The worker nodes are part of an Amazon EC2 Auto Scaling group for elasticity and redundancy.
3. **Kubernetes Pods:**
    - A **pod** is the smallest deployable unit in Kubernetes and usually contains one or more tightly coupled containers.
    - Pods run on worker nodes, and Kubernetes ensures that the desired state (the number of pods running) is maintained.
4. **Namespaces:**
    - Kubernetes **namespaces** provide a mechanism for dividing cluster resources between multiple teams or projects.
    - They are useful for managing complex environments with multiple microservices, environments, or teams.
5. **Kubernetes Deployments:**

    - A **deployment** in Kubernetes is a controller that manages stateless applications. It ensures that the correct number of pod replicas are running and provides capabilities like rolling updates, scaling, and rollback.
6. **Kubernetes Services:**
    - A **service** in Kubernetes abstracts the communication between pods and external clients. It provides stable IP addresses and DNS names for accessing pods.
    - Types of services include **ClusterIP** (internal access), **NodePort** (external access via node IP), and **LoadBalancer** (creates a cloud load balancer).
7. **Kubernetes Ingress:**
    - **Ingress** in Kubernetes manages external access to services in a cluster, usually HTTP/HTTPS traffic. It provides load balancing, SSL termination, and name-based virtual hosting.

EKS Launch Options:

1. **EKS on EC2:**
    - You can provision and manage worker nodes on Amazon EC2 instances, allowing full control over the EC2 instance types, networking, and scaling.
    - Offers flexibility in configuring worker nodes but requires you to handle scaling, patching, and maintenance of the instances.
2. **EKS with AWS Fargate:**
    - **AWS Fargate** provides a serverless compute environment for EKS where you don't need to provision or manage EC2 instances.
    - You define the CPU and memory requirements for each pod, and Fargate automatically provisions the necessary infrastructure.
    - Ideal for applications where you want to focus purely on the Kubernetes application layer without managing worker nodes.

Networking in EKS:

1. **Amazon VPC CNI Plugin:**
    - EKS uses the **Amazon VPC Container Network Interface (CNI)** plugin, which allows Kubernetes pods to have a native **VPC IP address**, enabling them to communicate with other AWS services securely and efficiently.
    - Each pod gets an Elastic Network Interface (ENI) in the VPC, allowing granular control through security groups and network access control lists (ACLs).
2. **Service Discovery:**
    - EKS integrates with **AWS Cloud Map** and **CoreDNS** to provide service discovery mechanisms for services running in the Kubernetes cluster.
3. **Ingress Controllers:**

- EKS supports Kubernetes ingress controllers like **NGINX**, **ALB Ingress Controller**, and **AWS App Mesh** to handle external HTTP/HTTPS traffic.
- You can also use AWS **Elastic Load Balancers (ELB)** to route traffic into Kubernetes services.

Storage in EKS:

1. **Persistent Volumes (PV) and Persistent Volume Claims (PVC):**
   - Kubernetes provides **Persistent Volumes (PV)** for storage that persists beyond the lifecycle of individual pods. These volumes are usually provisioned on AWS **Elastic Block Store (EBS)**.
   - **Persistent Volume Claims (PVC)** allow pods to request storage dynamically based on their needs.
2. **Elastic File System (EFS):**
   - EKS integrates with **Amazon EFS** to provide scalable, shared, persistent storage for Kubernetes pods. EFS can be mounted to multiple pods simultaneously, which is useful for stateful applications.

EKS Cluster Setup:

1. **Create an EKS Cluster:**

   You can create an EKS cluster using the AWS Management Console, **eksctl** (a simple command-line tool), or AWS CLI.

   The cluster control plane is created across multiple Availability Zones, and you can register EC2 instances or Fargate profiles as worker nodes.

2. **IAM Role for Worker Nodes:**

   The worker nodes in your EKS cluster need an **IAM role** to interact with AWS services. This role grants permission to the worker nodes to pull images from ECR, store logs in CloudWatch, and manage resources like Elastic Load Balancers.

3. **kubectl Configuration:**

kubectl is the command-line tool used to manage Kubernetes clusters. You configure **kubectl** to interact with your EKS cluster using the AWS CLI:

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

Monitoring and Logging in EKS:

1. **CloudWatch Logs:**

   EKS integrates with **CloudWatch Logs** for logging container activity and application logs. You can configure Kubernetes **Fluentd** to forward logs from your pods to CloudWatch Logs.

2. **CloudWatch Metrics:**

   EKS provides cluster performance metrics, such as CPU and memory usage, through **CloudWatch**.

   Prometheus and **Grafana** can be integrated with EKS to provide advanced monitoring and alerting capabilities.

3. **AWS X-Ray:**

   EKS integrates with **AWS X-Ray** for distributed tracing, allowing you to monitor, trace, and debug distributed applications and microservices running within your Kubernetes clusters.

4. **Kubernetes Dashboard:**

   You can deploy the **Kubernetes Dashboard** on your EKS cluster for a graphical view of the cluster's health, running pods, deployments, and resource usage.

Scaling in EKS:

1. **Horizontal Pod Autoscaling:**

   EKS supports **Horizontal Pod Autoscaling (HPA)**, which automatically adjusts the number of pod replicas in a deployment based on CPU, memory usage, or custom metrics.

2. **Cluster Autoscaler:**

   The **Cluster Autoscaler** for EKS automatically adjusts the size of your worker node group. If there aren't enough nodes to run pending pods, it adds nodes to the cluster, and if there are underutilized nodes, it terminates them.

   The autoscaler integrates with EC2 Auto Scaling groups.

Security in EKS:

1. **IAM Roles for Service Accounts (IRSA):**

   EKS integrates **IAM roles with Kubernetes service accounts**. You can assign IAM roles to Kubernetes service accounts, allowing your pods to access AWS services securely without managing credentials inside the containers.

2. **Pod Security Policies:**

   EKS supports **Pod Security Policies (PSPs)**, which define a set of conditions that a pod must meet to be allowed to run in the cluster, such as limiting the capabilities available to containers or enforcing certain security controls.

3. **Security Groups for Pods:**

   With **Amazon VPC CNI** plugin, each pod can be associated with a different security group, enabling fine-grained security control over network traffic to and from individual pods.

4. **Encryption:**

   EKS supports **encryption at rest** for Kubernetes secrets stored in **etcd** using AWS **KMS (Key Management Service)**.

Use Cases for EKS:

1. **Microservices:**

   EKS is well-suited for running microservices architectures, where different services are deployed as independent pods with their own lifecycle and scaling requirements.

2. **Hybrid Deployments:**

   EKS supports **AWS Outposts**, enabling you to run Kubernetes workloads both on-premises and in the cloud, using the same control plane and tools.

3. **Machine Learning:**

   Kubernetes provides a scalable environment for machine learning workloads, and with EKS, you can deploy ML frameworks like **TensorFlow** and **PyTorch** on a managed Kubernetes infrastructure.

4. **CI/CD Pipelines:**

EKS integrates with AWS **CodePipeline** and **CodeBuild** to set up continuous integration and deployment pipelines for containerized applications.

Pricing:

- **Control Plane:**

  AWS charges a fixed fee per EKS cluster for managing the control plane. This cost does not depend on the number of worker nodes or workloads running in the cluster.

- **Worker Nodes:**

  For EC2-based worker nodes, you pay for the EC2 instances, Elastic Load Balancing, EBS volumes, and data transfer.

- **Fargate Pricing:**

  When using Fargate for EKS, you are charged based on the vCPU and memory resources allocated to the running pods.