Networking

Networking in Kubernetes (KCNA Relevant)

Networking is a core component of Kubernetes, enabling communication between Pods, Services, and external resources. Below are the relevant **Networking** topics for **Kubernetes** in the context of the **KCNA** exam:

---

1. Kubernetes Networking Basics:

- **Pod-to-Pod Communication:**
  - Pods can communicate with each other within a Kubernetes cluster using their IP addresses.
  - Kubernetes assigns each Pod a unique IP address, and Pods on different nodes can communicate with each other over the cluster network.
- **Flat Network Model:**
  - Kubernetes assumes that every Pod can communicate with every other Pod in the cluster without NAT (Network Address Translation).

2. Services in Kubernetes:

- **ClusterIP (default):**
  - Exposes a service on a cluster-internal IP address. This type of service is only accessible within the Kubernetes cluster.
- **NodePort:**
  - Exposes a service on a specific port on each Node's IP address. Allows external access to the service through `<NodeIP>:<NodePort>`.
- **LoadBalancer:**

    - Provisioned by cloud providers to expose services externally, typically using an external load balancer (e.g., AWS ELB, GCP Load Balancer).
- **ExternalName:**
    - Maps a service to an external DNS name, allowing Kubernetes to access external services by their DNS names.

3. DNS (Domain Name System):

- **CoreDNS:**
    - Kubernetes uses **CoreDNS** for service discovery. Each Service gets a DNS entry that can be accessed using its name within the cluster.
- **Service Discovery:**
    - Pods can access Services using DNS names (e.g., `my-service.my-namespace.svc.cluster.local`).

4. Network Policies:

- **Network Policies:**
    - Allows you to control the communication between Pods. You can define rules to allow or block traffic between Pods based on labels, IP blocks, or namespaces.
- **Ingress and Egress Rules:**
    - Ingress: Incoming traffic to Pods.
    - Egress: Outgoing traffic from Pods.
- **Pod Security:**
    - Control which Pods can communicate with others, enhancing network isolation and security.

5. Ingress and Egress Controllers:

- **Ingress Controller:**
    - Manages HTTP/HTTPS traffic into the cluster. It routes traffic based on domain name, paths, or other rules defined in the Ingress resource.
    - Popular Ingress controllers: **NGINX Ingress**, **Traefik**, **HAProxy**.
- **Egress Controllers:**

- Manage outbound traffic from the cluster to external services. Ensures control and security of traffic leaving the cluster.

6. CNI (Container Network Interface):

- **CNI Plugins:**
  - Kubernetes uses CNI plugins to manage networking for containers. Popular CNI plugins include **Flannel**, **Calico**, **Weave**, and **Cilium**.
- **Networking Model:**
  - The CNI ensures that Pods on different nodes can communicate using an overlay network or other networking strategies.
- **Network Overlay:**
  - Virtual networks that enable Pod-to-Pod communication across different physical machines or nodes.

7. Load Balancing:

- **Service Load Balancing:**
  - Kubernetes Services can automatically distribute traffic to Pods based on service type (e.g., ClusterIP, NodePort, LoadBalancer).
- **Ingress Load Balancing:**
  - Ingress Controllers handle the distribution of HTTP(S) traffic across multiple Pods, supporting features like SSL termination, routing, etc.

8. Network Security:

- **mTLS (Mutual TLS) with Service Mesh:**
  - Service meshes like **Istio** can be used to enforce mTLS for secure communication between microservices.
- **Network Isolation:**
  - Using Network Policies to isolate services and restrict communication between Pods.
  - Restrict which services can access certain Pods based on labels and namespaces.

9. External Connectivity:

- **Outbound Networking:**
  - Pods can access external services outside the cluster, managed through **egress rules** and **NAT** configurations.
- **External IPs:**
  - Assigning external IP addresses to services (e.g., using LoadBalancer services or NodePort for external access).

10. Troubleshooting Networking Issues:

- **kubectl commands** like `kubectl get pods -o wide`, `kubectl describe pod <pod-name>`, `kubectl logs <pod-name>`, and `kubectl exec` to troubleshoot Pod networking issues.
- **Network Diagnostics Tools** like **ping**, **traceroute**, and **curl** to test connectivity between Pods, Services, and external endpoints.

Revision #2
Created 18 November 2024 21:51:46 by Admin
Updated 21 November 2024 20:03:04 by Admin