■ Kubernetes Ingress Controller Setup Explained

This document explains how a Kubernetes Ingress Controller using NGINX is set up. It includes all the necessary components: ServiceAccount, Deployment, ConfigMap, and Service, along with an explanation of how they work together to route traffic into your Kubernetes cluster.

■ ServiceAccount

The ServiceAccount grants permissions to the NGINX Ingress Controller so it can watch Ingress resources and services within the Kubernetes cluster.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nginx-ingress-serviceaccount
```

■ Deployment

The Deployment runs a Pod with the nginx-ingress-controller container. This controller listens on ports 80 and 443, monitors the Kubernetes API for Ingress resources, and dynamically updates NGINX to route traffic accordingly.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-ingress-controller
spec:
  replicas: 1
  selector:
    matchLabels:
      name: nginx-ingress-controller
  template:
    metadata:
      labels:
        name: nginx-ingress
    spec:
      containers:
        - name: nginx-ingress-controller
          image: quay.io/kubernetes-ingress-controller/nginx-ingress-controller:0.21.0
          args:
            - /nginx-ingress-controller
            - --configmap=$(POD_NAMESPACE)/nginx-configuration
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
          ports:
            - name: http
              containerPort: 80
            - name: https
              containerPort: 443
```

■■ ConfigMap

The ConfigMap provides runtime configuration for the NGINX controller. You can specify timeouts, headers, and other NGINX behaviors here.

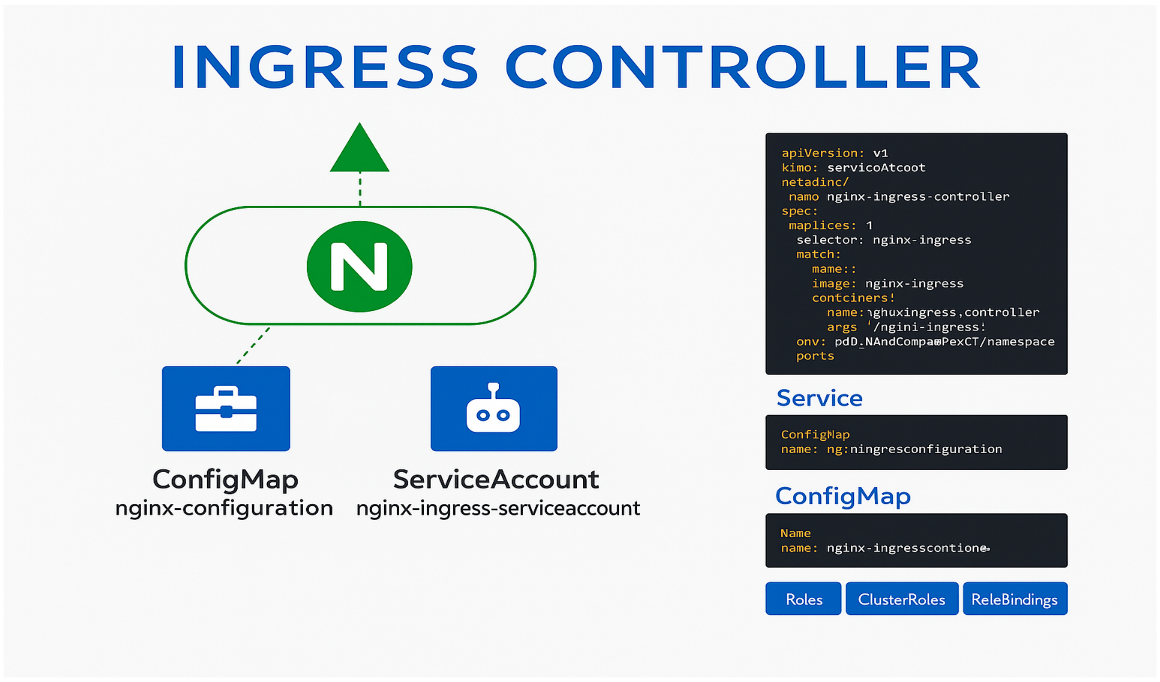
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-configuration
```

■ Service (NodePort)

This service exposes the NGINX Ingress Controller outside the cluster using NodePort, allowing access from external clients through specific ports (like 30080 for HTTP or 30443 for HTTPS).

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-ingress
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      name: http
    - port: 443
      targetPort: 443
      protocol: TCP
      name: https
  selector:
    name: nginx-ingress
```

■■ Diagram Overview



```
apiVersion: v1
kimo: servicoAtcoot
netadinc/
 namo nginx-ingress-controller
spec:
 maplices: 1
  selector: nginx-ingress
  match:
    mame::
    image: nginx-ingress
    contciners!
      name:ıghuxingress,controller
      args '/ngini-ingress!
  onv: pdD_NAndCompa■PexCT/namespace
  ports
```

```
ConfigMap
name: ng:ningresconfiguration
```

```
Name
name: nginx-ingresscontione
```

This setup creates a powerful, flexible way to expose multiple services using a single entry point. With NGINX as the Ingress Controller, you can perform path-based routing, TLS termination, and more.