How to Deploy a Simple Website on Kubernetes Using NGINX

This guide demonstrates deploying a simple static website using Kubernetes with a NodePort Service & ConfigMap. We'll serve the website via an NGINX container, and the NodePort Service will expose it on a specific port of the Kubernetes cluster nodes.

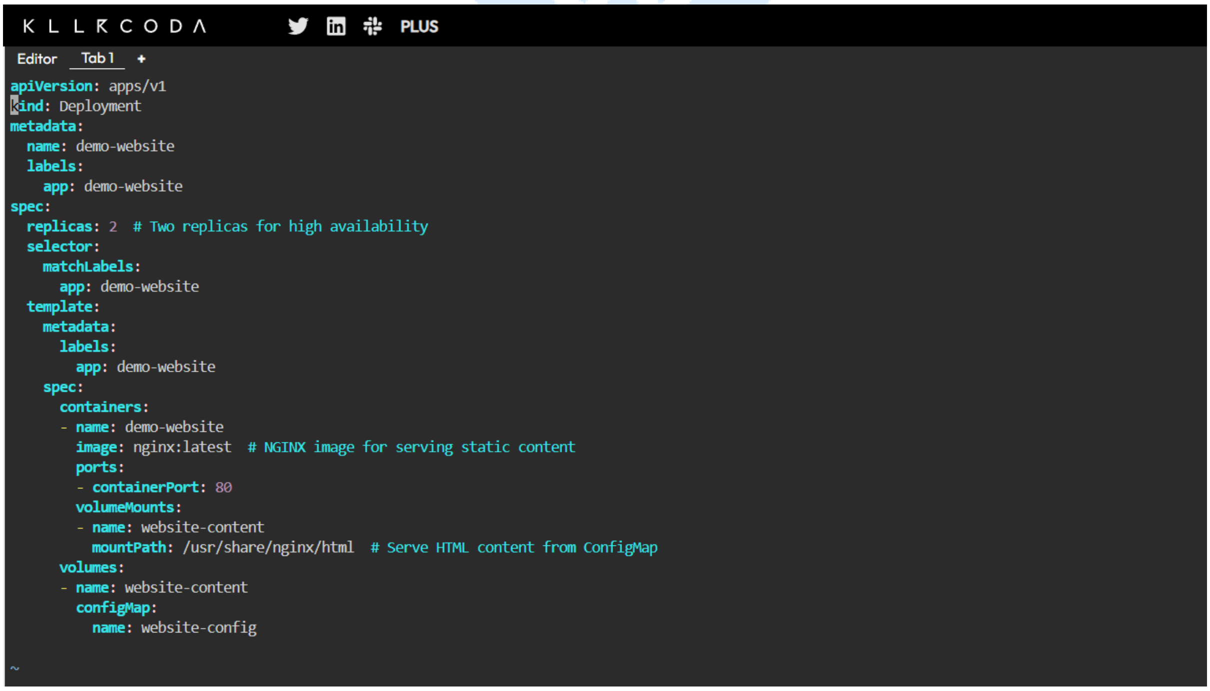
Prerequisites

1. A Kubernetes cluster (local like Minikube or any cloud-based cluster).
2. kubectl installed and configured to access the cluster.
3. Basic knowledge of Kubernetes YAML manifests.

Step 1: Create the Deployment File

The Deployment ensures the application runs with the desired number of replicas.

1. Create a file named `k8s/deployment.yaml`.

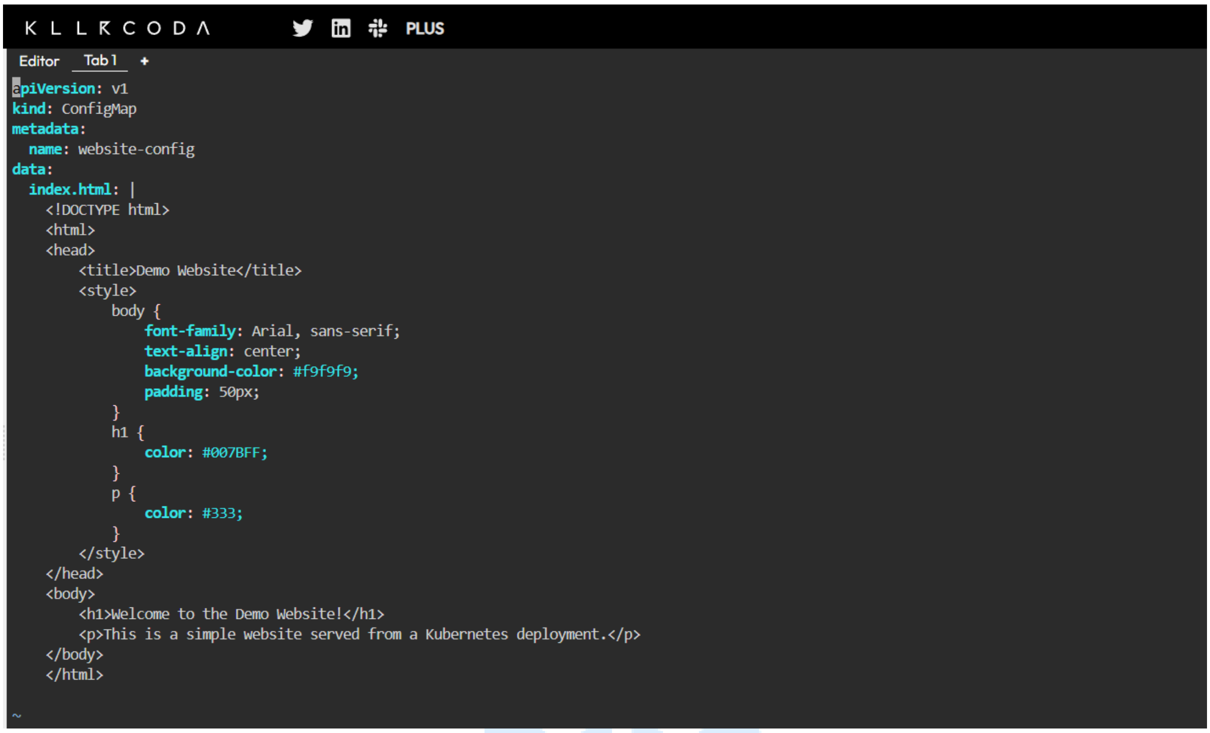


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-website
  labels:
    app: demo-website
spec:
  replicas: 2  # Two replicas for high availability
  selector:
    matchLabels:
      app: demo-website
  template:
    metadata:
      labels:
        app: demo-website
    spec:
      containers:
      - name: demo-website
        image: nginx:latest  # NGINX image for serving static content
        ports:
        - containerPort: 80
        volumeMounts:
        - name: website-content
          mountPath: /usr/share/nginx/html  # Serve HTML content from ConfigMap
      volumes:
      - name: website-content
        configMap:
          name: website-config
```

Step 2: Create the ConfigMap File

The ConfigMap stores the HTML content to be served by NGINX.

1. Create a file named `k8s/configmap.yaml`.



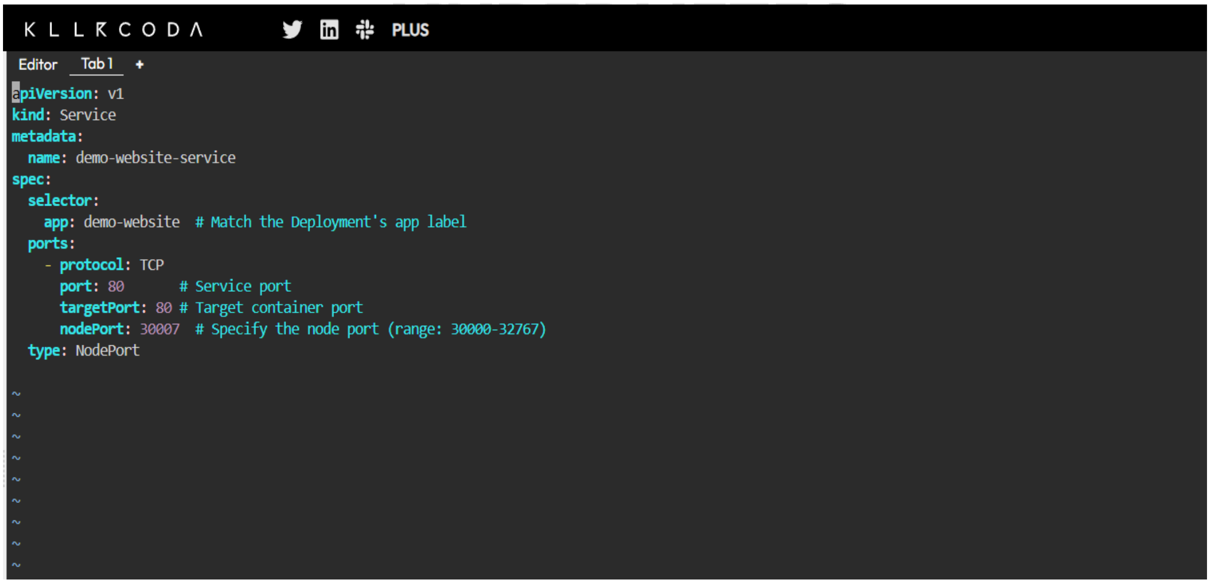
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: website-config
data:
  index.html: |
    <!DOCTYPE html>
    <html>
    <head>
        <title>Demo Website</title>
        <style>
            body {
                font-family: Arial, sans-serif;
                text-align: center;
                background-color: #f9f9f9;
                padding: 50px;
            }
            h1 {
                color: #007BFF;
            }
            p {
                color: #333;
            }
        </style>
    </head>
    <body>
        <h1>Welcome to the Demo Website!</h1>
        <p>This is a simple website served from a Kubernetes deployment.</p>
    </body>
    </html>

~
```

Step 3: Create the NodePort Service

A NodePort Service exposes the application on a specific port of each cluster node.

1. Create a file named `k8s/service.yaml`.



```
apiVersion: v1
kind: Service
metadata:
  name: demo-website-service
spec:
  selector:
    app: demo-website  # Match the Deployment's app label
  ports:
    - protocol: TCP
      port: 80        # Service port
      targetPort: 80 # Target container port
      nodePort: 30007  # Specify the node port (range: 30000-32767)
  type: NodePort

~
~
~
~
~
~
~
~
~
```

Step 4: Apply Kubernetes Manifests

Deploy the ConfigMap, Deployment, and Service.

1. Apply the ConfigMap:

```
kubectl apply -f k8s/configmap.yaml
```

2. Apply the Deployment:

```
kubectl apply -f k8s/deployment.yaml
```

3. Apply the Service:

```
kubectl apply -f k8s/service.yaml
```

Step 5: Verify the Deployment

1. **Check Pod Status**:

```
kubectl get pods
```

Ensure all pods are running.

2. **Check the NodePort Service**:

```
kubectl get svc demo-website-service
```

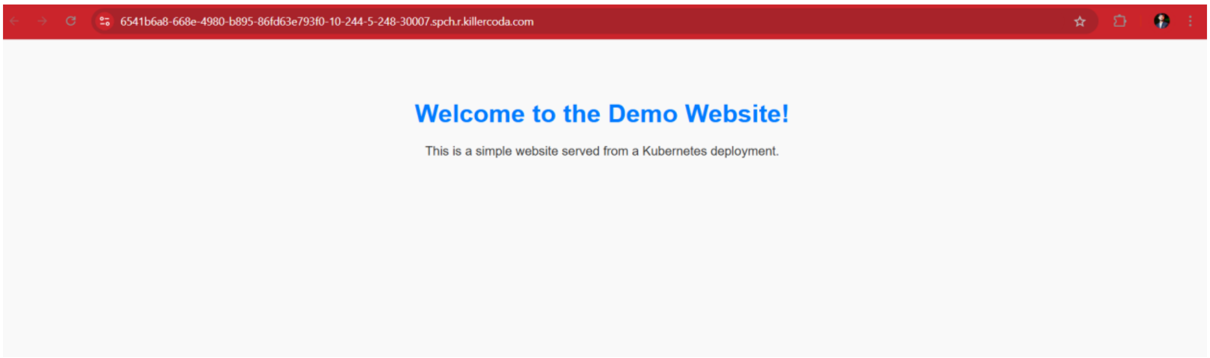
- Note the `NodePort` (30007 in this case).
- Retrieve the IP of a node in the cluster:

```
kubectl get nodes -o wide
```

Step 6: Access the Website

1. Use the following URL to access the website:

http://<Node-IP>:30007



6541b6a8-668e-4980-b895-86fd63e793f0-10-244-5-248-30007.spch.r.killercoda.com

Welcome to the Demo Website!

This is a simple website served from a Kubernetes deployment.

Conclusion

This guide demonstrated how to deploy a simple static website using Kubernetes with an NGINX container and expose it using a **NodePort Service**. This setup is ideal for local development or quick testing of applications. For production-grade deployments, consider using **LoadBalancer** or **Ingress** for enhanced scalability and flexibility.