Nginx Ingress Controller Installation Guide

Method 1: Using Helm (Recommended)

Prerequisites

```bash
# Install Helm if not already installed
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Verify Helm installation
helm version
```

Installation Steps

```bash
# Add the ingress-nginx repository
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

# Update your local Helm chart repository cache
helm repo update

# Install the ingress-nginx chart
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --set controller.replicaCount=2 \
  --set controller.nodeSelector."kubernetes\.io/os"=linux \
  --set defaultBackend.nodeSelector."kubernetes\.io/os"=linux \
  --set controller.admissionWebhooks.patch.nodeSelector."kubernetes\.io/os"=linux

# Verify installation
kubectl get pods -n ingress-nginx
kubectl get services -n ingress-nginx
```

Custom Values (Optional)

Create a `values.yaml` file for custom configuration:

```yaml
controller:
  replicaCount: 2

  service:
    type: LoadBalancer
    # For AWS ELB
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: nlb
      service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: '

  # Resource limits
  resources:
    limits:
      cpu: 500m
      memory: 512Mi
    requests:
      cpu: 100m
      memory: 128Mi

  # Enable metrics
  metrics:
    enabled: true
    service:
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "10254"

  # SSL configuration
  config:
    ssl-protocols: "TLSv1.2 TLSv1.3"
    ssl-ciphers: "ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-RSA-AES128-GCM-SHA256,ECDHE-ECDSA

defaultBackend:
  enabled: true
  image:
    repository: k8s.gcr.io/defaultbackend-amd64
    tag: "1.5"
```

Install with custom values:

bash

```bash
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --values values.yaml
```

Method 2: Using Kubectl (YAML Manifests)

bash

```bash
# Install using the official manifest
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller·

# For bare metal installations
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller·

# Verify installation
kubectl get pods -n ingress-nginx
kubectl get services -n ingress-nginx
```

Method 3: Cloud Provider Specific

AWS EKS

bash

```bash
# Install using AWS Load Balancer Controller
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --set controller.service.type=LoadBalancer \
  --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balance
  --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balance
```

Google GKE

bash

```bash
# Install for GKE
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --set controller.service.type=LoadBalancer \
  --set controller.service.annotations."cloud\.google\.com/load-balancer-type"="Externa
```

Azure AKS

```bash
# Install for AKS
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --set controller.service.type=LoadBalancer \
  --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balanc
```

Verification and Testing

Check Installation Status

```bash
# Check pods
kubectl get pods -n ingress-nginx

# Check services
kubectl get services -n ingress-nginx

# Check ingress class
kubectl get ingressclass

# Check logs
kubectl logs -n ingress-nginx -l app.kubernetes.io/name=ingress-nginx
```

Get Load Balancer IP/Hostname

```bash
# Get external IP or hostname
kubectl get service ingress-nginx-controller -n ingress-nginx

# Wait for external IP assignment (may take a few minutes)
kubectl get service ingress-nginx-controller -n ingress-nginx --watch
```

Test with Sample Application

Create a test application and ingress:

yaml

```yaml
# test-app.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: test-app
  template:
    metadata:
      labels:
        app: test-app
    spec:
      containers:
      - name: test-app
        image: nginx:latest
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: test-app-service
spec:
  selector:
    app: test-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-app-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: test.example.com
    http:
      paths:
      - path: /
```

```yaml
        pathType: Prefix
        backend:
          service:
            name: test-app-service
            port:
              number: 80
```

Apply the test:

```bash
kubectl apply -f test-app.yaml

# Check ingress
kubectl get ingress test-app-ingress

# Test (replace with your actual IP/hostname)
curl -H "Host: test.example.com" http://<LOAD_BALANCER_IP>
```

Configuration Options

Common Helm Values

yaml

```yaml
controller:
  # Number of replicas
  replicaCount: 2

  # Service configuration
  service:
    type: LoadBalancer
    # type: NodePort  # For on-premises
    # type: ClusterIP # For internal only

  # Resource limits
  resources:
    limits:
      cpu: 1000m
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 128Mi

  # Autoscaling
  autoscaling:
    enabled: true
    minReplicas: 2
    maxReplicas: 10
    targetCPUUtilizationPercentage: 80
    targetMemoryUtilizationPercentage: 80

  # Node affinity
  nodeSelector:
    kubernetes.io/os: linux

  # Pod anti-affinity
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: app.kubernetes.io/name
              operator: In
              values:
              - ingress-nginx
          topologyKey: kubernetes.io/hostname
```

Troubleshooting

Common Issues

1. **Pods not starting:**

   bash

   ```bash
   kubectl describe pods -n ingress-nginx
   kubectl logs -n ingress-nginx -l app.kubernetes.io/name=ingress-nginx
   ```

2. **Service not getting external IP:**

   bash

   ```bash
   # Check if your cluster supports LoadBalancer
   kubectl get nodes -o wide

   # For cloud providers, check if load balancer controller is installed
   kubectl get pods -A | grep -i load
   ```

3. **Ingress not working:**

   bash

   ```bash
   # Check ingress resource
   kubectl describe ingress <ingress-name>

   # Check controller logs
   kubectl logs -n ingress-nginx -l app.kubernetes.io/component=controller
   ```

Useful Commands

bash

```bash
# Upgrade ingress controller
helm upgrade ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx

# Uninstall
helm uninstall ingress-nginx -n ingress-nginx

# Check version
kubectl exec -n ingress-nginx -it deployment/ingress-nginx-controller -- /nginx-ingress
```

Security Considerations

1. **Enable admission webhooks** (enabled by default in Helm)
2. **Use TLS certificates** for HTTPS

3. **Configure proper RBAC** permissions
4. **Limit source IP ranges** in service annotations
5. **Enable rate limiting** in ingress annotations
6. **Use network policies** to restrict traffic

Monitoring and Observability

Enable Metrics

```yaml
controller:
  metrics:
    enabled: true
    service:
      annotations:
        prometheus.io/scrape: "true"
        prometheus.io/port: "10254"
```

Grafana Dashboard

- Import dashboard ID: 9614 (Official Nginx Ingress Controller)
- Monitor request rates, error rates, and response times

This guide covers the most common installation methods and configurations for Nginx Ingress Controller in Kubernetes clusters.