# Dokuwiki Deployment

## Requirements

In order to proceed, you must have a k8s platform, the kubectl command line utility, and a persistent storage volume.

Cerate a namespace for the wiki to live

```
kubectl create namespace dokuwiki
```

## Dokuwiki manifests

Deploy Dokuwiki to Kubernetes with YAML manifests that declare these resources:

- **Deployment:** runs the Dokuwiki container (image, port, replicas).
- **PersistentVolume / PVC:** stores wiki pages and uploads persistently.
- **Service:** exposes the pod within the cluster (ClusterIP/NodePort/LoadBalancer).
- **Ingress:** routes external HTTP(S) traffic to the Service (optional; depends on ingress controller).

Notes: set resource limits, securityContext, and back up volumes for production.

### dokuwiki-pvc.yaml

This file requests persistent storage for the wiki using Longhorn (or your preferred Kubernetes storage class – update the config accordingly). See [previous post] for more info.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: dokuwiki-pvc
  namespace: dokuwiki
spec:
  accessModes:
```

```
    - ReadWriteOnce
  storageClassName: longhorn
  resources:
    requests:
      storage: 2Gi
---
```

dokuwiki-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dokuwiki
  name: dokuwiki
  namespace: dokuwiki
spec:
  replicas: 1
  selector:
    matchLabels:
      app: dokuwiki
  template:
    metadata:
      labels:
        app: dokuwiki
    spec:
      containers:
      - name: dokuwiki-container
        image: lscr.io/linuxserver/dokuwiki
        imagePullPolicy: IfNotPresent
        env:
          - name: PUID
            value: "1001"
          - name: PGID
            value: "1001"
          - name: TZ
            value: "Europe/Amsterdam"
        ports:
          - containerPort: 80
            name: "http-wiki"
        volumeMounts:
          - name: dokuwiki-storage
            mountPath: "/config"
      volumes:
        - name: dokuwiki-storage
          persistentVolumeClaim:
            claimName: dokuwiki-pvc # Uses your existing Longhorn PVC
```

```
---
```

### Apply these manifests

Use these files to deploy a PersistentVolumeClaim and the Deployment to your cluster:

```
kubectl apply -n dokuwiki -f dokuwiki-pvc-yaml -f dokuwiki-
deployment.yaml
```

Verifiy objects are ready:

```
kubectl -n dokuwiki get deployments
kubectl -n dokuwiki get pvc
```

## Create an in-cluster Service object

Once the deployment is fully initialized — including successful binding of the PVC — a Service resource must be created to expose the Dokuwiki container.

There are two equivalent approaches:

1. **Declarative**: Apply a Service manifest (recommended for reproducibility)
2. **Imperative**: Create the Service directly via the command line

### Option 1: Using a Service YAML file (declarative)

#### dokuwiki-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: dokuwiki
  namespace: dokuwiki
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: dokuwiki
```

Apply the configuration:

```
kubectl apply -f dokuwiki-service.yaml
```

---

### Option 2: Using kubectl directly (imperative)

Create a ClusterIP Service that routes internal cluster traffic to the Dokuwiki pods:

```
kubectl expose deployment dokuwiki \
  --type=ClusterIP \
  --port=80 \
  --target-port=8080 \
  --name=dokuwiki
```

---

### Verification

To confirm that the Service has been created and is correctly configured:

```
kubectl describe svc dokuwiki
```

Optional quick check:

```
kubectl get svc dokuwiki -o wide
```

## Accessing the Wiki with Nginx Proxy Manager

I'm using Nginx Proxy Manager (NPM) as proxy service and AdGuard Home as DNS to handle external access and Let's Encrypt certificates for HTTPS. This simplifies exposing the wiki!

**You can skip the TLS Secret and Ingress sections below if using NPM.** NPM handles TLS termination and routing directly.

Here's how to configure NPM:

1. **Proxy Host:** Create a new proxy host in NPM, pointing to your DokuWiki service (e.g., `<ip-of-dokuwiki-service:80`).
2. **Domain Name:** Configure the desired domain name (e.g., `wiki.example.net`).
3. **Let's Encrypt:** Enable Let's Encrypt for automatic certificate management.

NPM will handle the SSL/TLS certificate and route traffic to your DokuWiki service within Kubernetes. No need to configure Ingress or Kubernetes Secrets for TLS!

**Regarding DNS:**

Create DNS A records on your local network pointing `wiki.example.net` to the IP address of your Proxmox host running NPM. NPM will then forward traffic to the DokuWiki service. This is also only needed if you do not have a DNS server in your home network. I am using AdGuard Home for this which forwards all requests to NPM first and then to the outside world. A piHole-Project server should also work fine here.

## Accessing DokuWiki from Outside the Cluster through Ingress

If you're not using Nginx Proxy Manager, you can expose the DokuWiki service using a Kubernetes Ingress. This acts as a reverse proxy, routing external traffic to your wiki.

### Securing with TLS (Optional if using NPM)

For secure HTTPS access, you'll need a TLS certificate. You can use a wildcard certificate for your local network, e.g. create a certificate with `mkcert`. If you prefer unencrypted HTTP, you can skip this step.

Here's an example of a TLS secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: testsecret-tls
  namespace: default
data:
  tls.crt: base64 encoded certificate data
  tls.key: base64 encoded key data
type: kubernetes.io/tls
```

### Defining the Ingress Resource

The Ingress resource defines how external traffic reaches the DokuWiki service.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dokuwiki-ingress
```

```
spec:
  tls: # Omit this section if not using HTTPS
  - hosts:
      - wiki.example.net
    secretName: testsecret-tls
  rules:
  - host: wiki.example.net
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: dokuwiki
            port:
              number: 80
```

This configuration directs traffic for `wiki.example.net` to the `dokuwiki` service on port 80. If you've configured TLS, the `tls` section associates the certificate with the domain.

---

Revision #2

Created 2026-03-20 09:39:52 UTC by Carsten

Updated 2026-03-20 10:25:55 UTC by Carsten