NEEDFUL DINGS // BLACK EDITION

Offizielles System-Handbuch & Technische Betriebsdokumentation

Inhaltsverzeichnis

1. System-Übersicht & Architektur-Topologie
2. Kryptografie, Schlüssel & Zugriffsberechtigungen (PKI)
3. 2.1 SSH Keypair Infrastructure
4. 2.2 WireGuard Mesh VPN Schlüssel & Profile
5. 2.3 SSL / TLS Zertifikate & HTTPS-Terminierung
6. 2.4 K3s Cluster mTLS & Internal CAs
7. Netzwerk- & Routing-Architektur
8. Microservice-Katalog & API-Referenz
9. Betriebs-Runbook, Monitoring & Wartung
10. Disaster Recovery & Notfall-Prozeduren

1. System-Übersicht & Architektur-Topologie

Das Gesamtsystem **NeedfulDings // Black Edition** ist als zweistufige hybride Virtualisierungs- und Container-Plattform konzipiert:

```
flowchart TD
    subgraph WAN["Öffentliches Netz (Internet)"]
        CLIENT[Browser / Externe Clients / API SDKs]
        COLAB[Remote GPU Worker / Colab Bridge]
    end

    subgraph Host["Stufe 1: Root Hypervisor Host (Debian 13 Trixie)"]
        PUB_IP["Öffentliche IP: 82.165.188.200"]
        IPT[iptables NAT & DNAT Forwarding]
        WG["WireGuard Daemon (wg0: 10.8.0.1 :51820)"]
        CK["Cockpit Web Terminal (:9090)"]
        KVM[KVM / QEMU / libvirt Engine]
        VNET["Virtual Bridge (virbr0: 192.168.122.1/24)"]

        PUB_IP --> IPT
        PUB_IP --> CK
        PUB_IP --> WG
        IPT --> VNET
        KVM --> VNET
    end

    subgraph Guest["Stufe 2: KVM Virtual Machine (debian-k3s)"]
        VM_IP["Statische IP: 192.168.122.237 (4 vCPU / 8 GB RAM / 150 GB SSD)"]
        ING["Traefik Ingress Controller (:80 / :443)"]
        ZMQ["ZeroMQ Broker HostBinding (:5598 / :5599)"]

        subgraph K8S["Kubernetes Namespace: needfuldings"]
            HQ["NEEDFUL HQ (:8080) -> /hq & /"]
            OMNI["OmniRoute AI Gateway (:8000) -> /omniroute, /v1, /risk"]
            SPIDER["News-Spider & Graph Blog (:8001) -> /spider, /blog"]
            SHOP["4D Finance Desk (:8002) -> /shop, /finance"]
            LK["LiveKit Media & Avatar Studio (:7880) -> /livekit"]
            N8N["n8n Workflow Studio (:5678) -> /n8n"]
        end

        VNET --> VM_IP
        VM_IP --> ING
        VM_IP --> ZMQ
        ING --> HQ
        ING --> OMNI
        ING --> SPIDER
        ING --> SHOP
        ING --> LK
        ING --> N8N
    end

    CLIENT -->|HTTP/S :80, :443| PUB_IP
    CLIENT -->|ZeroMQ :5598, :5599| PUB_IP
    COLAB -->|WireGuard UDP :51820| WG
```

Hardware & Spezifikationen

Parameter	Root Hypervisor Host	KVM Gast (`debian-k3s`)
Betriebssystem	Debian GNU/Linux 13 (Trixie)	Debian GNU/Linux 12 (Bookworm Cloud)
Kernel	6.12.117+deb13-amd64	6.1.0-56-cloud-amd64
CPU Zuweisung	AMD EPYC-Milan (6 vCPUs physisch)	4 vCPUs (host-model passthrough)
RAM Zuweisung	12 GB physisch (4 GB für Host/Cockpit reserviert)	8 GB (8192 MB) fest dediziert
Massenspeicher	360 GB NVMe (/dev/vda1)	150 GB virtio qcow2 Disk
Primäre IP	`82.165.188.200`	`192.168.122.237` (statisch über libvirt DHCP)
MAC-Adresse	`02:01:f0:2d:ea:9c` (ens6)	`52:54:00:26:ad:24` (vnet0)

2. Kryptografie, Schlüssel & Zugriffsberechtigungen (PKI)

2.1 SSH Keypair Infrastructure

Für die administrative Authentifizierung zwischen Host und KVM-Gast sowie für externe Administratoren wird Ed25519-Kryptografie genutzt.

Root Host Keypair

- **Dateipfade:**
- Private Key: `/root/.ssh/id_ed25519` (Dateiberechtigung: `0600`)
- Public Key: `/root/.ssh/id_ed25519.pub` (Dateiberechtigung: `0644`)
- **Public Key:**
  `text ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICoECiYF57gcesbZ6elRFpIuRP61D3uj5VgSZ/oxVfrw root@32a67601-d273-4312-851c-2ed28467fd6c`
- **Fingerprint:**
  `text SHA256:DQbyquD3sjdDIxyXgXd6U7PobE+x6lHGuM7/Hd9n3mo`

KVM-Gast Authentifizierung

Der Public Key des Hypervisors ist in der KVM-Instanz in `/root/.ssh/authorized_keys` hinterlegt. Direkte SSH-Verbindung vom Hypervisor in den Gast:

```
ssh -i /root/.ssh/id_ed25519 root@192.168.122.237
```

2.2 WireGuard Mesh VPN Schlüssel & Profile

Das WireGuard-VPN dient der verschlüsselten Standortvernetzung und dem Einbinden externer Worker (z. B. Google Colab GPU-Instanzen oder Remote-Clients) in das interne private Subnetz `10.8.0.0/24`.

- **Server-Interface:** `wg0` auf Root Hypervisor
- **Port:** `51820/udp`
- **Subnetz:** `10.8.0.1/24`

Server-Schlüssel (`/etc/wireguard/`)

- **Server Public Key:**
  `text xOtLH6geSiHcrhcYPu9jUIEBrKDEaDPj9uhJLMN7Tww=`
- **Server Private Key:** Gespeichert in `/etc/wireguard/server_private.key` (Berechtigung: `0600`)

Client-Schlüssel & Profil (Colab GPU Bridge / Remote Worker)

- **Client Public Key:**
  `text SsEg+BnHwhqcmEqV1hK/kxhZ/ppD6QpbLvQSV1t9M0I=`
- **Client Private Key:** Gespeichert in `/etc/wireguard/client_private.key`

Einsatzbereite Client-Konfiguration (`/root/wireguard-client.conf`):

```
[Interface]
PrivateKey = UCH+rMynNXCVQ5nYft9NNYBSgTNExG2KDsvXId95hHQ=
Address = 10.8.0.2/24
DNS = 1.1.1.1

[Peer]
PublicKey = xOtLH6geSiHcrhcYPu9jUIEBrKDEaDPj9uhJLMN7Tww=
Endpoint = 82.165.188.200:51820
AllowedIPs = 10.8.0.0/24
PersistentKeepalive = 25
```

2.3 SSL / TLS Zertifikate & HTTPS-Terminierung

Die Verschlüsselung von Webverkehr erfolgt auf Ebene des Traefik Ingress Controllers innerhalb von K3s.

Let's Encrypt / Certbot Automatisierung

Wenn die Domain `needfuldings.de` auf die IP `82.165.188.200` zeigt, kann Let's Encrypt über Traefik oder Certbot aktiviert werden.

Methode A: Traefik ACME Ingress (Automatisch)

Fügen Sie der Traefik-Konfiguration in K3s folgendes Manifest hinzu (`/var/lib/rancher/k3s/server/manifests/traefik-config.yaml` auf dem KVM-Gast):

```
apiVersion: helm.cattle.io/v1
kind: HelmChartConfig
metadata:
  name: traefik
  namespace: kube-system
spec:
  valuesContent: |-
    additionalArguments:
      - "--certificatesresolvers.le.acme.email=admin@needfuldings.de"
      - "--certificatesresolvers.le.acme.storage=/data/acme.json"
      - "--certificatesresolvers.le.acme.httpchallenge.entrypoint=web"
```

Methode B: Standalone Certbot Zertifikate via DNS/HTTP-01

Auf dem Root-Host oder Gast ausführen:

```
apt install -y certbot
certbot certonly --standalone -d needfuldings.de -d www.needfuldings.de
```

Die erzeugten Zertifikate werden in Kubernetes als TLS-Secret hinterlegt:

```
kubectl create secret tls needfuldings-tls \
  --cert=/etc/letsencrypt/live/needfuldings.de/fullchain.pem \
  --key=/etc/letsencrypt/live/needfuldings.de/privkey.pem \
  -n needfuldings
```

Und im Ingress (`05-ingress.yaml`) aktiviert:

```
spec:
  tls:
    - hosts:
        - needfuldings.de
      secretName: needfuldings-tls
```

Cockpit Web Console TLS (`:9090`)

Cockpit generiert standardmäßig ein selbstsigniertes Zertifikat in `/etc/cockpit/ws-certs.d/`.
Um ein offizielles Let's Encrypt Zertifikat in Cockpit einzubinden:

```
cat /etc/letsencrypt/live/needfuldings.de/fullchain.pem \
    /etc/letsencrypt/live/needfuldings.de/privkey.pem \
    > /etc/cockpit/ws-certs.d/50-needfuldings.cert
systemctl restart cockpit
```

2.4 K3s Cluster mTLS & Internal CAs

Alle Kubernetes-Komponenten kommunizieren untereinander über mutual TLS (mTLS).
- **Zertifikatsverzeichnis im KVM-Gast:** `/var/lib/rancher/k3s/server/tls/`
- **Root Cluster CAs:**

- `server-ca.crt` & `server-ca.key`
- `client-ca.crt` & `client-ca.key`
- `request-header-ca.crt` & `request-header-ca.key`
- **Cluster API Zertifikat Subject Alternative Names (SANs):**
- `127.0.0.1`
- `192.168.122.237` (KVM Internal IP)
- `82.165.188.200` (Öffentliche Root IP)
- `needfuldings.de`
- `kubernetes`, `kubernetes.default`, `kubernetes.default.svc`
- **Kubeconfig für Root-Zugriff:** Gespeichert auf dem Root-Host unter `/root/.kube/config`.

---

3. Netzwerk- & Routing-Architektur

Port-Matrix & Weiterleitungsregeln

Der Hypervisor leitet eingehenden Datenverkehr über die physische Netzwerkschnittstelle `ens6` transparent an die KVM-Instanz `192.168.122.237` weiter:

Port	Protokoll	Quell-Interface	Ziel-IP & Port	Dienst / Verwendung
80	TCP	`ens6` / Host	`192.168.122.237:80`	HTTP Web Traffic (Traefik Ingress)
443	TCP	`ens6` / Host	`192.168.122.237:443`	HTTPS Verschlüsselter Web Traffic
5598	TCP	`ens6` / Host	`192.168.122.237:5598`	ZeroMQ RPC (REQ/REP)
5599	TCP	`ens6` / Host	`192.168.122.237:5599`	ZeroMQ Pub/Sub Event Stream
6443	TCP	`ens6` / Host	`192.168.122.237:6443`	K3s Kubernetes API Server
9090	TCP	Host lokal	Host `127.0.0.1:9090`	Cockpit Web Management Console
51820	UDP	Host lokal	Host `wg0`	WireGuard Mesh VPN Daemon

iptables Konfiguration (Persistiert)

Die Routing-Regeln sind über `netfilter-persistent` gesichert und bleiben nach einem Server-Neustart vollständig erhalten:

```
# Eingehendes Port-Forwarding
iptables -t nat -A PREROUTING -i ens6 -p tcp -m multiport --dports 80,443,6443,5598,5599 -j
DNAT --to-destination 192.168.122.237
iptables -A FORWARD -p tcp -d 192.168.122.237 -m multiport --dports 80,443,6443,5598,5599 -m
state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Lokales Loopback-Forwarding für Host-Abfragen
iptables -t nat -A OUTPUT -p tcp -d 82.165.188.200 -m multiport --dports 80,443,6443,5598,5599
-j DNAT --to-destination 192.168.122.237
```

---

4. Microservice-Katalog & API-Referenz

Alle Microservices laufen im Namespace `needfuldings` und sind über Traefik Ingress geroutet:

1. NEEDFUL HQ // Black Edition

- **Zweck:** Oberste Management-Matrix, Status-Dashboard und 1-Click Systemüberwachung.
- **URL:** http://82.165.188.200/hq (bzw. `/`)
- **API Endpunkte:**
- `GET /health` — Status und Laufzeit
- `GET /api/matrix` — JSON-Live-Matrix aller 11 Subsysteme

2. OmniRoute AI Gateway & RiskOfficer

- **Zweck:** Multi-Provider LLM Proxy (OpenAI, Gemini, Anthropic, DeepSeek) mit automatischem Fallback-Routing und Trading-Risikofilter.
- **URL:** http://82.165.188.200/omniroute
- **API Endpunkte:**
- `GET /v1/models` — OpenAI-kompatible Modell-Liste
- `POST /v1/chat/completions` — OpenAI-kompatibler Prompt-Endpunkt
- `POST /risk/evaluate` — Evaluierung von Hebel, Ordergröße und Risikosperren

3. Agentic News-Spider & Graph Blog Engine

- **Zweck:** Autonomer News-Crawler mit automatischer Extraktion von Wissensgraphen im Obsidian-Format (`[[Keywords]]`).
- **URL:** http://82.165.188.200/spider (bzw. `/blog`)
- **API Endpunkte:**
- `GET /spider/api/nodes` — Graph-Knoten und Kanten im JSON-Format
- `POST /spider/api/crawl` — Manueller Sofortstart eines Crawl-Zyklus

4. NeedfulShop // 4D Finance Desk

- **Zweck:** Quantitative Arbitrage- und Handelskonsole mit Echtzeit-Quotes (BTC/USDT, ETH/USDT, SOL/USDT) und Order-Blotter.
- **URL:** http://82.165.188.200/shop (bzw. `/finance`, `/desk`)
- **API Endpunkte:**
- `GET /api/quotes` — Live-Preise, Spreads und 24h-Volumen
- `POST /api/order` — Orderausführung über ZeroMQ

5. LiveKit Real-Time Media & Avatar Studio

- **Zweck:** WebRTC-Medienverarbeitung und Live-Broadcasting für `needful-avatar-newsroom-live`.
- **URL:** http://82.165.188.200/livekit
- **API Endpunkte:**
- `GET /livekit/api/rooms` — Aktive WebRTC-Räume und Audio/Video-Tracks

6. n8n Workflow Automation Studio

- **Zweck:** Visuelle Orchestrierungs- und Workflow-Engine zur Verknüpfung von Webhooks, APIs, ZMQ-Events und KI-Agenten.
- **URL:** http://82.165.188.200/n8n

7. ZeroMQ Hochgeschwindigkeits-Broker

- **RPC Endpunkt:** `tcp://82.165.188.200:5598` (REQ/REP Pattern)
- Befehl: `{"action": "ping"}` $\rightarrow$ `{"status": "ok", "response": "pong"}`
- Befehl: `{"action": "status"}` $\rightarrow$ Uptime & Request-Counter
- **PUB/SUB Endpunkt:** `tcp://82.165.188.200:5599`
- Topic: `telemetry` (Broadcast-Heartbeat alle 2 Sekunden)

---

5. Betriebs-Runbook, Monitoring & Wartung

Wichtigste Administrationsbefehle

1. Cluster & Pods überwachen

```
# Alle Pods im NeedfulDings Namespace prüfen
kubectl get pods -n needfuldings -o wide

# Logs eines bestimmten Dienstes einsehen (z. B. OmniRoute)
kubectl logs -n needfuldings -l app=needful-omniroute --tail=50 -f

# Dienst neu starten (Rolling Restart)
kubectl rollout restart deployment/needful-hq -n needfuldings
```

2. KVM Gast-VM verwalten

```
# VM-Status prüfen
virsh list --all

# VM stoppen / starten / neu starten
virsh reboot debian-k3s
virsh shutdown debian-k3s
virsh start debian-k3s

# Direkte serielle Konsole öffnen (Beenden mit Ctrl + ])
virsh console debian-k3s
```

3. WireGuard VPN Status

```
wg show
systemctl status wg-quick@wg0
```

4. Cockpit Management

```
systemctl status cockpit.socket
# Erreichbar unter: https://82.165.188.200:9090
```

---

6. Disaster Recovery & Notfall-Prozeduren

Szenario A: Physischer Host-Reboot

Alle Komponenten sind für autonomen Kaltstart konfiguriert:
1. `libvirtd` startet automatisch und bootet die KVM-VM `debian-k3s` (`autostart` aktiv).
2. `netfilter-persistent` lädt alle Port-Forwardings sofort.
3. `wg-quick@wg0` fährt den VPN-Tunnel hoch.
4. K3s innerhalb der VM startet automatisch alle Pods im Namespace `needfuldings`.
Kein manueller Eingriff erforderlich.

Szenario B: K3s Cluster Snapshot erstellen

Um den Zustand aller Workloads, ConfigMaps und Ingress-Routen zu sichern:

```
ssh -i /root/.ssh/id_ed25519 root@192.168.122.237 \
  "k3s etcd-snapshot save --name needful-backup-$(date +%Y%m%d)"
```

Szenario C: Snapshot der KVM-Festplatte

Vor tiefgreifenden Systemänderungen kann ein Live-Snapshot der virtuellen Disk erstellt werden:

```
virsh snapshot-create-as --domain debian-k3s \
  --name "snapshot-$(date +%Y%m%d)" \
  --description "Pre-Upgrade Snapshot" \
  --atomic
```

Szenario D: Wiederherstellung der Kubernetes-Manifeste

Sämtliche YAML-Definitionen liegen versioniert im Verzeichnis:

```
cd /root/needfuldings-infra/k8s/
kubectl apply -f .
```

Dokumentation generiert am 23. November 2026 für NeedfulDings // Black Edition.