Manual de Arquitetura e Implementação de Docker Swarm em Infraestrutura VPS Distribuída: Hardening, Persistência e Orquestração de Alta Disponibilidade

1. Fundamentos Arquiteturais e Planejamento Estratégico

A implementação de sistemas distribuídos em ambientes de nuvem pública, especificamente utilizando Virtual Private Servers (VPS) geograficamente dispersos ou logicamente isolados sem uma Virtual Private Cloud (VPC) nativa, impõe desafios de engenharia significativos. Este documento técnico detalha a arquitetura, configuração e manutenção de um cluster Docker Swarm sobre três nós KVM (KVM 2, KVM 4, KVM 8), focando na segurança de tráfego em rede hostil (Internet pública), persistência de dados centralizada e roteamento de tráfego eficiente.

1.1 Análise da Topologia de Infraestrutura e Heterogeneidade de Recursos

A infraestrutura proposta baseia-se em uma tríade de servidores com capacidades computacionais assimétricas. Esta heterogeneidade exige uma estratégia de orquestração que transcenda o balanceamento de carga "Round Robin" padrão, necessitando de um agendamento determinístico de tarefas (Task Scheduling) baseado em restrições de hardware e afinidade de dados.[1]

Inventário de Recursos e Atribuição de Funções no Plano de Controle:

Nó (Hostname)	Recurso (Modelo)	Função no Swarm	Responsabilidade Crítica	Estratégia de Placement
KVM 2	VPS (Baixa Cap.)	Manager + Worker	Manutenção de Quórum Raft; Roteamento de Borda.	Stateless Workers; Proxy Ingress Secundário.
KVM 4	VPS (Média Cap.)	Manager + Worker	Manutenção de Quórum	Stateless Microservices;

			Raft; Processamento de Aplicação.	Web Servers.
KVM 8	VPS (Alta Cap.)	Manager + Worker	**Líder Preferencial**; Persistência de Dados (DB/NFS).	Stateful Services (DB, Storage); Traefik Controller.

A decisão de configurar todos os três nós como **Managers** adere às melhores práticas de disponibilidade para clusters pequenos. O algoritmo de consenso Raft, utilizado pelo Docker Swarm para gerenciar o estado global do cluster, exige um quórum de $(N/2) + 1$ para validar escritas. Em um cluster de 3 nós, o sistema tolera a falha de exatamente 1 nó.[2] Se dois nós falharem simultaneamente, o cluster entra em modo somente leitura para preservar a integridade, impedindo novas implantações ou rebalanceamento.[3]

A ausência de uma rede privada isolada (VPC/LAN) entre as VPSs da Hostinger implica que todo o tráfego de controle (porta 2377) e plano de dados (VXLAN/Overlay) trafegará pela interface pública.[4] Isso eleva a criptografia de "opcional" para **mandatória**, exigindo o uso de redes overlay criptografadas com IPSEC (AES-GCM), o que introduz overhead de CPU no encapsulamento/desencapsulamento de pacotes, fator que deve ser considerado especialmente no nó KVM 2.[4]

1.2 O Teorema CAP e a Latência de Rede WAN

Em ambientes distribuídos sobre WAN, a latência de rede torna-se um fator preponderante. O Docker Swarm prioriza a Consistência (C) e Tolerância a Partição (P) em detrimento da Disponibilidade (A) imediata em caso de partição de rede (CP system). A latência entre os nós Managers deve ser monitorada rigorosamente; latências superiores a 500ms podem causar eleições de líder instáveis, onde os nós acreditam que o líder caiu devido a timeouts no *heartbeat* do Raft.[2]

Para mitigar riscos de latência e segurança, a arquitetura de armazenamento será centralizada via NFSv4 no nó de maior capacidade (KVM 8), enquanto o processamento será distribuído. O banco de dados também será "pinado" (fixado) no KVM 8 para garantir performance de I/O local, evitando a latência proibitiva de rodar um banco de dados sobre um volume de rede montado via WAN.

---

2. Preparação do Sistema Operacional e Otimização do Kernel

A estabilidade do Docker Swarm, especialmente quando operando com criptografia de rede overlay, depende intrinsecamente da configuração do kernel Linux. Distribuições modernas como Ubuntu 22.04 LTS e 24.04 LTS introduziram mudanças na gestão de módulos de criptografia e firewall que, se não tratadas, impedem silenciosamente o funcionamento da rede criptografada.

2.1 Identidade e Resolução de Nomes (Mesh DNS Estático)

O Swarm depende de resolução de nomes consistente para a comunicação entre nós. Sem um servidor DNS interno privado, a utilização do arquivo /etc/hosts é a abordagem mais robusta para criar uma malha de resolução confiável.

Procedimento de Configuração de Identidade:

Execute os comandos abaixo em cada nó respectivo para definir hostnames persistentes que refletem a hierarquia do cluster:

Bash

```
# No Nó KVM 2
sudo hostnamectl set-hostname kvm2-manager

# No Nó KVM 4
sudo hostnamectl set-hostname kvm4-manager

# No Nó KVM 8
sudo hostnamectl set-hostname kvm8-core
```

A seguir, edite o arquivo /etc/hosts em **todos** os nós. É imperativo utilizar os IPs Públicos (Public IPs) fornecidos pela Hostinger, dado que não há rede privada.[5] A consistência deste arquivo previne erros de resolução durante o handshake TLS mútuo entre os nós.

Bash

```
# /etc/hosts - Adicionar em TODOS os nós
# Substitua X.X.X.X pelos IPs Reais
X.X.X.2  kvm2-manager
X.X.X.4  kvm4-manager
X.X.X.8  kvm8-core
```

2.2 Requisitos Críticos de Kernel para Criptografia Overlay (IPSEC)

O Docker Swarm utiliza o kernel Linux para realizar a criptografia de tráfego VXLAN através do protocolo IPSEC ESP (Encapsulating Security Payload). Em versões recentes do Ubuntu (22.04+), certos módulos necessários para essa operação não são carregados por padrão ou, em alguns kernels virtualizados (KVM otimizados), podem estar ausentes.[7]

A falha em carregar esses módulos resulta em um cenário onde a rede overlay é criada, mas os containers em nós diferentes não conseguem se comunicar (pacotes são dropados), ou o tráfego flui sem criptografia se o Docker falhar em aplicar as regras de IPSEC.

Auditoria e Carregamento de Módulos: Verifique a presença dos módulos overlay, br_netfilter, xt_u32 e esp4. O módulo xt_u32 é vital para que o iptables consiga filtrar pacotes VXLAN baseados no cabeçalho SPI (Security Parameter Index) e aplicar a descriptografia correta.[10]

Crie um arquivo de carregamento persistente em /etc/modules-load.d/swarm-crypto.conf:

Bash

```
overlay
br_netfilter
ip_tables
iptable_filter
# Módulos críticos para criptografia IPSEC em Overlay
esp4
xfrm_user
xfrm_algo
xt_u32
# Necessário para balanceamento de carga IPVS
ip_vs
ip_vs_rr
ip_vs_wrr
ip_vs_sh
```

Aplique o carregamento imediato:

Bash

```
sudo modprobe overlay
sudo modprobe br_netfilter
sudo modprobe xt_u32
sudo modprobe xfrm_user
sudo modprobe esp4
```

Análise de Risco: Se o comando modprobe xt_u32 falhar com "Module not found", é indicativo de que o kernel fornecido pela imagem da VPS é uma versão minimalista. Neste caso, é obrigatória a instalação dos pacotes de módulos extras:

Bash

```
sudo apt-get update
sudo apt-get install linux-modules-extra-$(uname -r)
```

2.3 Tuning de Sysctl para Redes de Alta Latência

A rede overlay sobre WAN sofre com latência e jitter. Os parâmetros padrão do stack TCP/IP do Linux são otimizados para LANs de baixa latência. Ajustes são necessários para evitar desconexões prematuras dos nós do cluster.

Crie o arquivo /etc/sysctl.d/99-swarm-tuning.conf:

Ini, TOML

```
# Habilitar Forwarding IPv4 (Obrigatório para roteamento de containers)
net.ipv4.ip_forward = 1
net.ipv4.conf.all.forwarding = 1

# Aumentar a tolerância de vizinhança ARP (Evita falhas em redes grandes/lentas)
```

```
net.ipv4.neigh.default.gc_thresh1 = 8192
net.ipv4.neigh.default.gc_thresh2 = 12288
net.ipv4.neigh.default.gc_thresh3 = 16384

# Ajustes de Keepalive para detecção mais rápida de falhas em conexões ociosas
net.ipv4.tcp_keepalive_time = 600
net.ipv4.tcp_keepalive_intvl = 30
net.ipv4.tcp_keepalive_probes = 5

# Aumentar backlog de conexões para lidar com bursts de tráfego no Traefik
net.core.somaxconn = 65535
```

Aplique as configurações: sudo sysctl --system.

---

3. Estratégia de Segurança de Rede: Firewall em Profundidade

A segurança desta infraestrutura depende de uma abordagem de "Defesa em Profundidade", utilizando duas camadas de firewall: o Firewall de Borda da Hostinger e o UFW (Uncomplicated Firewall) no nível do host. O maior vetor de risco é a interação conflitante entre o Docker e o firewall local, onde o Docker insere regras no iptables que "furam" o bloqueio padrão do UFW.[11]

3.1 Camada 1: Firewall de Borda (Hostinger)

O firewall da Hostinger atua antes que o tráfego atinja a interface de rede da VPS, economizando ciclos de CPU que seriam gastos processando pacotes descartados. Esta é a camada ideal para bloqueios grossos.

Matriz de Configuração Recomendada (Painel Hostinger):

Protocolo	Porta	Ação	Origem (Source)	Justificativa Técnica
TCP	22	Allow	Seu IP Residencial/VPN	Acesso SSH Administrativo restrito.
TCP	80, 443	Allow	0.0.0.0/0	Tráfego Web

			(Any)	público para o Traefik (Proxy Reverso).
TCP	2377	Allow	IPs de KVM 2, 4, 8	Comunicação do Cluster Management Plane (Raft).[4]
TCP/UDP	7946	Allow	IPs de KVM 2, 4, 8	Comunicação Gossip entre nós (Discovery).[4]
UDP	4789	Allow	IPs de KVM 2, 4, 8	Tráfego de Dados Overlay (VXLAN).[4]
ESP	(Protocol 50)	Allow	IPs de KVM 2, 4, 8	Tráfego Criptografado IPSEC.[7]
TCP	2049	Allow	IPs de KVM 2, 4, 8	Protocolo NFSv4 para persistência de dados.[13]

Nota: Se o painel da Hostinger não permitir regras baseadas em Protocolo IP (como ESP/50) ou não oferecer granularidade de IP de origem para todas as regras, configure-o para "Allow All" nas portas necessárias e delegue a filtragem estrita para a Camada 2 (UFW).

3.2 Camada 2: UFW e a Cadeia DOCKER-USER

O problema clássico ao usar Docker com UFW é que o Docker manipula a tabela NAT do iptables diretamente. Quando você publica uma porta (ex: -p 8080:80), o Docker insere uma regra DNAT na cadeia PREROUTING que redireciona o pacote antes que ele atinja a cadeia INPUT do UFW.[11] Resultado: a porta fica aberta para o mundo, ignorando ufw deny 8080.

Para corrigir isso sem desabilitar a manipulação de iptables do Docker (o que quebraria o DNS interno e redes overlay), devemos inserir regras na cadeia DOCKER-USER, que o Docker garante ser avaliada antes das suas regras automáticas.[14]

3.2.1 Configuração Base do UFW (Todos os Nós)

Primeiro, configure as regras padrão para serviços que rodam no host (SSH, NFS Server) e não em containers.

Bash

```
# Resetar para estado limpo
sudo ufw reset
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Permitir SSH (Idealmente restrito ao seu IP)
sudo ufw allow proto tcp from <SEU_IP_CASA> to any port 22
```

3.2.2 Estabelecendo a Malha de Confiança (Trust Mesh)

Devemos permitir explicitamente todo o tráfego entre os nós do cluster nas interfaces públicas. Isso resolve a comunicação do Swarm e do NFS sem precisar abrir portas individuais para a internet.

No KVM 8 (Exemplo):

Bash

```
# Permitir tráfego vindo do KVM 2
sudo ufw allow from <IP_KVM2> to any comment 'Trust KVM2 Cluster Traffic'
# Permitir tráfego vindo do KVM 4
sudo ufw allow from <IP_KVM4> to any comment 'Trust KVM4 Cluster Traffic'
# Permitir protocolo ESP para criptografia (Importante!)
sudo ufw allow proto esp from <IP_KVM2> to any
sudo ufw allow proto esp from <IP_KVM4> to any
```

Repita a lógica invertida nos nós KVM 2 e KVM 4.

3.2.3 Solução Definitiva para Portas Docker (Traefik)

Para o Traefik, que ouvirá nas portas 80 e 443, queremos que ele seja acessível publicamente. Como o Docker "fura" o UFW, por padrão isso já aconteceria. Porém, para serviços que *não* devem ser públicos (ex: um banco de dados de teste exposto temporariamente), a segurança

padrão é falha.

A boa prática é **não publicar portas diretamente nos workers** para serviços internos. Use a rede Overlay. Apenas o Traefik deve ter portas 80/443 publicadas.

Se houver necessidade de bloquear portas de containers expostos, utilize a ferramenta ufw-docker ou adicione regras manuais ao arquivo /etc/ufw/after.rules. Dado o escopo deste manual, assumiremos que a segurança de borda (Hostinger) e a não-exposição de portas desnecessárias (uso de rede overlay interna) são as medidas primárias.

Ative o firewall:

Bash

```
sudo ufw enable
```

---

4. Arquitetura de Armazenamento: NFSv4 "Pinned"

O requisito de centralizar o armazenamento no KVM 8 exige uma configuração robusta de NFS. O protocolo NFSv3 é inadequado para ambientes WAN/Firewall devido à sua dependência do rpcbind (Porta 111) e alocação dinâmica de portas para mountd e lockd.[13] Implementaremos um ambiente **NFSv4-only**, que opera exclusivamente sobre uma única porta TCP (2049), simplificando drasticamente o firewall.[17]

4.1 Configuração do Servidor NFS (KVM 8)

Instalação e Estrutura de Diretórios:

Bash

```
sudo apt update
sudo apt install nfs-kernel-server -y

# Criar diretório raiz para exportação (Pseudo-root)
sudo mkdir -p /srv/nfs/swarm_data
# Ajustar permissões (NFS mapeia root para nobody por padrão - root_squash)
```

```
sudo chown nobody:nogroup /srv/nfs/swarm_data
sudo chmod 777 /srv/nfs/swarm_data # Ajustar conforme necessidade de segurança interna
```

Configuração do Daemon para NFSv4 Exclusivo: Edite o arquivo /etc/nfs.conf (padrão no Ubuntu 22.04+) para desabilitar versões antigas e UDP.[19]

Ini, TOML

```
# /etc/nfs.conf
[nfsd]
vers3=n
vers4=y
vers4.0=y
vers4.1=y
vers4.2=y
udp=n
tcp=y
```

Além disso, é necessário garantir que o rpcbind não exponha serviços desnecessários, embora o NFSv4 tecnicamente não precise dele para descoberta se montado corretamente, o serviço systemd ainda pode ativá-lo. A proteção principal é o firewall (já configurado para bloquear porta 111 externa).

Definição de Exportações (/etc/exports):

Utilizaremos a opção fsid=0 para definir a "raiz" do NFSv4. Isso permite que os clientes montem / em vez do caminho completo, ocultando a estrutura interna do servidor.

Bash

```
# /etc/exports
/srv/nfs/swarm_data  <IP_KVM2>(rw,sync,no_subtree_check,no_root_squash,fsid=0)
/srv/nfs/swarm_data  <IP_KVM4>(rw,sync,no_subtree_check,no_root_squash,fsid=0)
```

Detalhe Técnico: A opção no_root_squash é frequentemente necessária para volumes Docker, pois o daemon do Docker no cliente escreve arquivos como root. Sem isso, o servidor NFS converteria essas escritas para nobody, causando erros de permissão na inicialização de

containers (ex: PostgreSQL chown falhando). Em um ambiente de confiança controlada (IPs estáticos), isso é um risco aceitável.

Aplique a configuração:

Bash

```
sudo exportfs -rav
sudo systemctl restart nfs-kernel-server
```

4.2 Configuração dos Clientes (KVM 2 e KVM 4)

Instale o suporte a NFS:

Bash

```
sudo apt install nfs-common -y
```

Teste a montagem manualmente antes de configurar no Docker:

Bash

```
# Note que montamos a raiz "/" devido ao fsid=0
sudo mount -t nfs -o vers=4,proto=tcp,port=2049 <IP_KVM8>:/ /mnt
```

Se o comando retornar sem erros e df -h mostrar o volume, a conectividade NFSv4 via TCP 2049 está operacional.

---

5. Implementação do Docker Swarm e Rede Overlay Criptografada

Com a infraestrutura de rede, kernel e armazenamento pronta, a inicialização do orquestrador

é o próximo passo.

5.1 Instalação do Docker Engine

Evite pacotes snap do Ubuntu para Docker, pois eles introduzem complexidade em montagens de arquivos e permissões. Utilize o repositório oficial upstream.[14]

Bash

```
# Em TODOS os nós (KVM 2, 4, 8)
# Remover versões legadas
sudo apt-get remove docker docker-engine docker.io containerd runc

# Instalar repositório oficial
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# Criar usuário docker para evitar uso de root direto (opcional mas recomendado)
sudo usermod -aG docker $USER
```

5.2 Inicialização do Cluster e Criptografia

O KVM 8 será o iniciador do cluster. A flag --data-path-port=4789 é o padrão, mas explicitá-la ajuda na documentação.

Bash

```
# No KVM 8
docker swarm init --advertise-addr <IP_KVM8>
```

O comando retornará um token para adicionar Managers. Como o requisito é que **todos** sejam managers/workers (uma topologia "plana"), usaremos este token.

No KVM 2 e KVM 4:

Bash

```
docker swarm join --token <TOKEN_GERADO> <IP_KVM8>:2377
```

Validação de Nós e Rotulagem (Labels):

Para garantir que os serviços sejam executados nos nós corretos (Pinning), utilizaremos labels lógicos. Isso desacopla a configuração do hostname físico.

No KVM 8 (Líder), execute:

Bash

```
# Rotular KVM 8 como nó de Storage e Database
docker node update --label-add role=storage --label-add role=db kvm8-core

# Rotular KVM 2 e 4 como workers genéricos (embora sejam managers no raft)
docker node update --label-add role=worker kvm2-manager
docker node update --label-add role=worker kvm4-manager
```

5.3 Criação da Rede Overlay Segura

A criação da rede deve incluir a opção --opt encrypted. Isso ativa o driver IPSEC no kernel. Sem os módulos esp4 e xfrm_user verificados na Seção 2.2, esta rede falharia em transmitir dados, embora o comando de criação funcionasse.[6]

Bash

```
docker network create --driver overlay --opt encrypted --attachable swarm-secure-net
```

Nota de Performance: A criptografia de todo o tráfego Leste-Oeste (entre containers) consome CPU. No KVM 2, monitore o uso de si (software interrupt) no top durante testes de carga. Se o overhead for proibitivo, considere não criptografar tráfego não-sensível criando uma segunda rede overlay sem a opção --opt encrypted.

---

6. Orquestração de Serviços: Database, Storage e Proxy

Esta seção define os manifestos (Stacks) para implantar a infraestrutura de aplicação, respeitando os requisitos de pinning e persistência.

6.1 Banco de Dados: PostgreSQL Pinado (KVM 8)

Bancos de dados relacionais exigem baixa latência de disco e garantias ACID. Executá-los sobre NFS em WAN é uma receita para corrupção de dados. Portanto, o banco será fixado no KVM 8 usando armazenamento local direto (Bind Mount ou Docker Volume Local).

Crie o arquivo db-stack.yml:

YAML

```
version: '3.8'

services:
  database:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD_FILE: /run/secrets/db_password
      POSTGRES_DB: app_db
    secrets:
```

```
      - db_password
    networks:
      - swarm-secure-net
    volumes:
      # Volume local no disco do KVM 8. Alta performance I/O.
      - pg_data:/var/lib/postgresql/data
    deploy:
      replicas: 1
      placement:
        constraints:
          # Constraint RÍGIDA: Apenas roda se o nó tiver label role=db (KVM 8)
          - node.labels.role == db
      restart_policy:
        condition: on-failure
        delay: 5s

volumes:
  pg_data:
    # Este volume é local ao nó onde o container roda.
    # Como o container está pinado no KVM 8, o volume sempre estará lá.

networks:
  swarm-secure-net:
    external: true

secrets:
  db_password:
    external: true
```

Análise de Risco: Se o KVM 8 falhar catastroficamente (perda de disco), os dados do banco serão perdidos a menos que haja uma rotina de backup externa (ex: dump enviado para S3). O Docker Swarm não replica volumes locais entre nós automaticamente.

6.2 Traefik: Proxy Reverso e Desafios de ACME

O Traefik atuará como gateway de entrada (Ingress). O desafio crítico em clusters distribuídos é a gestão de certificados SSL (Let's Encrypt). O Traefik Community Edition **não suporta** armazenamento distribuído de certificados (como Consul/Etcd) nativamente de forma estável. Ele armazena certificados em um arquivo JSON (acme.json).

Se tivermos réplicas do Traefik em nós diferentes (KVM 2, 4, 8), cada um tentará gerar certificados, atingindo o limite de taxa da Let's Encrypt, ou os arquivos ficarão dessincronizados.[21]

Solução Arquitetural: "Traefik Controller Pinned". Executaremos uma única instância do Traefik responsável pela geração de certificados, fixada no KVM 8 (onde temos persistência garantida). Para alta disponibilidade de *recebimento* de tráfego, o Docker Swarm Mesh Routing permite que requisições na porta 80/443 de *qualquer* nó (KVM 2 ou 4) sejam roteadas internamente para o Traefik no KVM 8.[23]

Crie o arquivo traefik-stack.yml:

YAML

```yaml
version: '3.8'

services:
  traefik:
    image: traefik:v2.10
    command:
      - "--providers.docker=true"
      - "--providers.docker.swarmMode=true"
      - "--providers.docker.exposedByDefault=false"
      - "--providers.docker.network=swarm-secure-net"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      # Redirecionamento HTTP -> HTTPS
      - "--entrypoints.web.http.redirections.entryPoint.to=websecure"
      - "--entrypoints.web.http.redirections.entryPoint.scheme=https"
      # Configuração Let's Encrypt (HTTP Challenge)
      - "--certificatesresolvers.le.acme.httpchallenge=true"
      - "--certificatesresolvers.le.acme.httpchallenge.entrypoint=web"
      - "--certificatesresolvers.le.acme.email=admin@seudominio.com"
      - "--certificatesresolvers.le.acme.storage=/letsencrypt/acme.json"
    ports:
      # Publicação em modo Ingress (Mesh).
      # Portas 80/443 abertas em KVM 2, 4 e 8.
      - target: 80
        published: 80
        protocol: tcp
        mode: ingress
      - target: 443
        published: 443
        protocol: tcp
```

```
      mode: ingress
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock:ro
    # Persistência do ACME no KVM 8
    - traefik_certs:/letsencrypt
  networks:
    - swarm-secure-net
  deploy:
    replicas: 1
    placement:
      constraints:
        # Fixado no KVM 8 para garantir consistência do acme.json
        - node.labels.role == db # Ou criar label específica role=proxy
    restart_policy:
      condition: on-failure

volumes:
 traefik_certs:

networks:
 swarm-secure-net:
   external: true
```

Insight de Rede: Graças ao "Ingress Mesh" do Swarm, você pode apontar o DNS do seu domínio para os IPs de KVM 2, KVM 4 e KVM 8 (Round Robin DNS). Se uma requisição bater no KVM 2, o Swarm a encaminhará via túnel criptografado para o container do Traefik no KVM 8. Isso oferece HA de *rede*, embora o *serviço* Traefik seja um ponto único de falha se o KVM 8 cair.

6.3 Serviços Stateless com Armazenamento NFS (Workers)

Para serviços que rodam nos workers (KVM 2, 4) e precisam acessar arquivos (ex: upload de imagens de um CMS), utilizaremos o volume NFS montado via driver Docker.

Exemplo de serviço web genérico:

YAML

```
version: '3.8'

services:
```

```yaml
  webapp:
    image: nginx:alpine
    networks:
      - swarm-secure-net
    volumes:
      - type: volume
        source: nfs-storage
        target: /usr/share/nginx/html
    deploy:
      replicas: 3 # Uma cópia em cada nó
      placement:
        constraints:
          - node.role == worker # Ou sem constraints para espalhar globalmente
      labels:
        - "traefik.enable=true"
        - "traefik.http.routers.web.rule=Host(`app.seudominio.com`)"
        - "traefik.http.routers.web.entrypoints=websecure"
        - "traefik.http.routers.web.tls.certresolver=le"
        - "traefik.http.services.web.loadbalancer.server.port=80"

volumes:
  nfs-storage:
    driver: local
    driver_opts:
      type: nfs
      o: "addr=<IP_KVM8>,rw,nfsvers=4,soft,rsize=8192,wsize=8192,timeo=14,retrans=2"
      device: ":/"
```

Explicação das Opções de Montagem NFS:

- nfsvers=4: Força o uso do protocolo v4 (TCP 2049).
- soft: Impede que o container trave indefinidamente (hang) se o KVM 8 cair. Ele retornará erro de I/O após timeout, permitindo que a aplicação trate a falha.
- rsize/wsize=8192: Tamanhos de bloco conservadores para WAN. Valores maiores (32k, 64k) podem sofrer fragmentação se o MTU da rede pública variar, causando perda de performance.
- device: ":/": Sintaxe específica do NFSv4 quando fsid=0 é usado no servidor.

---

7. Procedimentos de Manutenção e Monitoramento

7.1 Backup e Recuperação

O KVM 8 é o "SPOF" (Single Point of Failure) de dados.

- **Backup do Banco:** Configure um cron job no KVM 8 para executar docker exec <ID_CONTAINER_DB> pg_dumpall > dump.sql e envie para um armazenamento externo (S3/Backblaze).
- **Backup do NFS:** Use rsync para espelhar /srv/nfs/swarm_data para fora do ambiente Hostinger regularmente.

7.2 Monitoramento de Conectividade Overlay

Monitore logs do kernel (dmesg) em busca de mensagens "IPSEC" ou "xfrm". Erros aqui indicam falha na rotação de chaves do Swarm ou bloqueio de firewall na porta ESP/4789.

Para verificar se o tráfego está sendo criptografado, use tcpdump em um worker:

Bash

```
sudo tcpdump -i <INTERFACE_PUBLICA> -n proto 50
```

Se você ver pacotes ESP fluindo entre os IPs dos nós, a criptografia está ativa e funcional.

7.3 Atualização de Nós (Drain)

Para manutenção no KVM 2 ou 4:

Bash

```
docker node update --availability drain kvm2-manager
```

Isso moverá os containers para outros nós. Após a manutenção:

Bash

```
docker node update --availability active kvm2-manager
```

Atenção: Drenar o KVM 8 causará indisponibilidade do Banco de Dados e Traefik, pois eles estão pinados e não podem migrar. Planeje janelas de manutenção para este nó.

Referências citadas

1. Placement Constraints - MedStack Docs, acessado em fevereiro 2, 2026, https://support.medstack.co/docs/placement-constraints
2. How nodes work - Docker Docs, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/
3. Administer and maintain a swarm of Docker Engines, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/swarm/admin_guide/
4. Getting started with Swarm mode - Docker Docs, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/swarm/swarm-tutorial/
5. Does the Hostinger allow communication between two different VPS instances via private IP within their local network? - Reddit, acessado em fevereiro 2, 2026, https://www.reddit.com/r/Hostinger/comments/1j51rw6/does_the_hostinger_allow_communication_between/
6. Manage swarm service networks - Docker Docs, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/swarm/networking/
7. Security update: Encrypted overlay networks in Moby and Mirantis Container Runtime, acessado em fevereiro 2, 2026, https://www.mirantis.com/blog/security-update-encrypted-overlay-networks-in-moby-and-mirantis-container-runtime/
8. CVE-2023-28841 Detail - NVD, acessado em fevereiro 2, 2026, https://nvd.nist.gov/vuln/detail/cve-2023-28841
9. Docker Swarm Overlay Networking not working with systemd please help! - Reddit, acessado em fevereiro 2, 2026, https://www.reddit.com/r/selfhosted/comments/uqhks4/docker_swarm_overlay_networking_not_working_with/
10. Encrypted overlay network traffic may be unencrypted · Advisory - GitHub, acessado em fevereiro 2, 2026, https://github.com/moby/moby/security/advisories/GHSA-33pg-m6jh-5237
11. Packet filtering and firewalls - Docker Docs, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/network/packet-filtering-firewalls/
12. TIL: Docker overrides ufw and iptables rules by injecting it's own rules : r/selfhosted - Reddit, acessado em fevereiro 2, 2026, https://www.reddit.com/r/selfhosted/comments/1atjsra/til_docker_overrides_ufw_and_iptables_rules_by/
13. Open NFS ports | Portworx Enterprise Documentation, acessado em fevereiro 2, 2026, https://docs.portworx.com/portworx-enterprise/provision-storage/create-pvcs/open-nfs-ports
14. Install Docker Engine on Ubuntu, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/install/ubuntu/
15. nfs is blocked by ufw even though ports are opened - Ask Ubuntu, acessado em

fevereiro 2, 2026, https://askubuntu.com/questions/103910/nfs-is-blocked-by-ufw-even-though-ports-are-opened
16. Changing NFS version 4 to version 3 - Ask Ubuntu, acessado em fevereiro 2, 2026, https://askubuntu.com/questions/1532611/changing-nfs-version-4-to-version-3
17. NFSv4 with only one open port : r/linux - Reddit, acessado em fevereiro 2, 2026, https://www.reddit.com/r/linux/comments/e0c8ws/nfsv4_with_only_one_open_port/
18. How to Set Up an NFS Mount on Ubuntu (Step-by-Step Guide) | DigitalOcean, acessado em fevereiro 2, 2026, https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-20-04
19. Network File System (NFS) - Ubuntu Server documentation, acessado em fevereiro 2, 2026, https://documentation.ubuntu.com/server/how-to/networking/install-nfs/
20. Disable NFSv3 on Ubuntu 22.04 and newer - LANsible.com, acessado em fevereiro 2, 2026, https://lansible.com/2023/10/disable-nfsv3-on-ubuntu-22.04-and-newer/
21. Is there a best practice solution for distributed certificate storage in HA docker swarm cluster with traefik CE. - Reddit, acessado em fevereiro 2, 2026, https://www.reddit.com/r/Traefik/comments/xvym9a/is_there_a_best_practice_solution_for_distributed/
22. Help with storing Let's Encrypt acme.json in a Docker volume for reuse - Reddit, acessado em fevereiro 2, 2026, https://www.reddit.com/r/docker/comments/16701ar/help_with_storing_lets_encrypt_acmejson_in_a/
23. Use Swarm mode routing mesh - Docker Docs, acessado em fevereiro 2, 2026, https://docs.docker.com/engine/swarm/ingress/