Apps

Productivity apps in no particular order.

8: Vaultwarden - Password Management Solution

Vaultwarden is a lightweight, self-hosted password management solution that provides full Bitwarden API compatibility while using significantly fewer resources than the official Bitwarden server. It supports all Bitwarden clients and offers enterprise-grade security features. For comprehensive configuration options, API documentation, and advanced features, please refer to the official Vaultwarden documentation.

Architecture Overview

Vaultwarden provides the following core functionality:

- **Bitwarden API Compatibility** - Full compatibility with all Bitwarden clients
- **End-to-End Encryption** - AES-256 encryption for all data
- **Multi-User Support** - Organization and user management
- **WebSocket Support** - Real-time synchronization
- **Admin Panel** - Comprehensive management interface
- **Database Flexibility** - SQLite, PostgreSQL, MySQL support

Prerequisites

Before installing Vaultwarden, ensure the following infrastructure is in place:

- **Traefik installed** (from Chapter 4)
- **Docker running** (from Chapter 3)
- **Borgmatic installed** (from Chapter 5) - Automated backup protection
- **Domain configured** (from Chapter 4.5)
- **SSL certificates** (Let's Encrypt via Traefik)

Installation Methods

Via Infinity Tools Menu

Navigate to the Infinity Tools menu and select:

```
□□APPLICATIONS → Vaultwarden → Install
```

Command Line Installation

```
# Direct script execution
sudo bash /opt/InfinityTools/Solutions/setup-vaultwarden.sh --install

# With environment variables
export VW_DOMAIN="vault.domain.com"
export VW_USE_TRAEFIK="true"
export VW_SIGNUPS="false"
export PROXY_NETWORK="proxy"
sudo -E bash /opt/InfinityTools/Solutions/setup-vaultwarden.sh --install
```

Configuration Parameters

Required Configuration

During installation, you'll configure:

- **SSL Mode:** Traefik integration or standalone
- **Domain:** FQDN for web vault access
- **Signup Policy:** Open registration or admin-only
- **Admin Token:** Generated automatically for admin access

Environment Variables

```
# SSL and Domain Configuration
export VW_USE_TRAEFIK="true"              # Use Traefik for SSL termination
export VW_DOMAIN="vault.domain.com"       # FQDN for web vault
export VW_PORT="8443"                     # Port for standalone mode

# User Management
export VW_SIGNUPS="false"                 # Disable open registration
export VW_SIGNUPS_VERIFY="true"           # Require email verification

# Network Configuration
```

```
export PROXY_NETWORK="proxy"             # Docker network name
```

Generated Configuration

Docker Compose Configuration (Traefik Mode)

Location: `/opt/speedbits/vaultwarden/docker-compose.yml`

```
version: '3.8'

services:
  vaultwarden:
    image: vaultwarden/server:1.34.3
    container_name: vaultwarden
    restart: unless-stopped
    environment:
      DOMAIN: https://vault.domain.com
      ADMIN_TOKEN_FILE: /run/secrets/admin_token.txt
      SIGNUPS_ALLOWED: "false"
      SIGNUPS_VERIFY: "true"
      DATABASE_URL: /data/db.sqlite3
      WEBSOCKET_ENABLED: "true"
      WEBSOCKET_ADDRESS: 0.0.0.0
      WEBSOCKET_PORT: 3012
    volumes:
      - /opt/speedbits/vaultwarden/data:/data
      - /opt/speedbits/vaultwarden/admin_token.txt:/run/secrets/admin_token.txt:ro
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.vaultwarden.rule=Host(`vault.domain.com`)"
      - "traefik.http.routers.vaultwarden.entrypoints=websecure"
      - "traefik.http.routers.vaultwarden.tls.certresolver=myresolver"
      - "traefik.http.services.vaultwarden.loadbalancer.server.port=80"
      - "traefik.http.routers.vaultwarden-websocket.rule=Host(`vault.domain.com`) &&
Path(`/notifications/hub`)"
      - "traefik.http.routers.vaultwarden-websocket.entrypoints=websecure"
      - "traefik.http.routers.vaultwarden-websocket.tls.certresolver=myresolver"
      - "traefik.http.services.vaultwarden-websocket.loadbalancer.server.port=3012"
```

```
    networks:
      - proxy

networks:
  proxy:
    external: true
```

Standalone Configuration

For environments without Traefik:

```
version: '3.8'

services:
  vaultwarden:
    image: vaultwarden/server:1.34.3
    container_name: vaultwarden
    restart: unless-stopped
    environment:
      DOMAIN: https://localhost:8443
      ADMIN_TOKEN_FILE: /run/secrets/admin_token.txt
      SIGNUPS_ALLOWED: "false"
      DATABASE_URL: /data/db.sqlite3
      WEBSOCKET_ENABLED: "true"
      WEBSOCKET_ADDRESS: 0.0.0.0
      WEBSOCKET_PORT: 3012
      ROCKET_TLS: '{certs="/ssl/vaultwarden.crt",key="/ssl/vaultwarden.key"}'
      ROCKET_PORT: 443
    volumes:
      - /opt/speedbits/vaultwarden/data:/data
      - /opt/speedbits/vaultwarden/admin_token.txt:/run/secrets/admin_token.txt:ro
      - /opt/speedbits/vaultwarden/ssl:/ssl:ro
    ports:
      - "8443:443"
    networks:
      - proxy
```

Security Configuration

Admin Token Management

Admin tokens are stored securely and provide access to the admin panel:

```
# Generate new admin token
openssl rand -base64 48

# Store in secure location
echo "generated_token" > /opt/speedbits/vaultwarden/admin_token.txt
chmod 600 /opt/speedbits/vaultwarden/admin_token.txt
```

Security Headers

Traefik middleware provides comprehensive security headers:

```
labels:
  - "traefik.http.middlewares.vaultwarden-security.headers.customResponseHeaders.X-Content-Type-Options=nosniff"
  - "traefik.http.middlewares.vaultwarden-security.headers.customResponseHeaders.X-Frame-Options=SAMEORIGIN"
  - "traefik.http.middlewares.vaultwarden-security.headers.customResponseHeaders.X-XSS-Protection=1; mode=block"
  - "traefik.http.middlewares.vaultwarden-security.headers.customResponseHeaders.Strict-Transport-Security=max-age=31536000; includeSubDomains"
  - "traefik.http.middlewares.vaultwarden-security.headers.customResponseHeaders.Referrer-Policy=strict-origin-when-cross-origin"
  - "traefik.http.middlewares.vaultwarden-security.headers.customResponseHeaders.Content-Security-Policy=default-src 'self'; script-src 'self' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self' data: https:; connect-src 'self' wss://vault.domain.com https://vault.domain.com; font-src 'self' data:; object-src 'none'; base-uri 'self'; form-action 'self'; frame-ancestors 'none';"
```

Database Configuration

SQLite (Default)

Vaultwarden uses SQLite by default for simplicity:

```
environment:
  DATABASE_URL: /data/db.sqlite3
```

PostgreSQL Configuration

For production environments, PostgreSQL is recommended:

```
environment:
  DATABASE_URL: postgresql://vaultwarden:password@postgres:5432/vaultwarden

# Add PostgreSQL service
services:
  postgres:
    image: postgres:15-alpine
    container_name: vaultwarden-postgres
    restart: unless-stopped
    environment:
      POSTGRES_DB: vaultwarden
      POSTGRES_USER: vaultwarden
      POSTGRES_PASSWORD: secure_password
    volumes:
      - /opt/speedbits/vaultwarden/postgres:/var/lib/postgresql/data
    networks:
      - proxy
```

Advanced Configuration

Environment Variables

Vaultwarden supports extensive configuration via environment variables:

```
environment:
  # Domain and SSL
  DOMAIN: https://vault.domain.com
  ROCKET_TLS: '{certs="/ssl/vaultwarden.crt",key="/ssl/vaultwarden.key"}'

  # Database
```

```
  DATABASE_URL: /data/db.sqlite3

  # User Management
  SIGNUPS_ALLOWED: "false"
  SIGNUPS_VERIFY: "true"
  SIGNUPS_VERIFY_RESEND_TIME: "3600"
  SIGNUPS_VERIFY_RESEND_LIMIT: "6"

  # Security
  ADMIN_TOKEN_FILE: /run/secrets/admin_token.txt
  INVITATIONS_ALLOWED: "true"
  INVITATION_ORG_NAME: "Organization Name"

  # WebSocket
  WEBSOCKET_ENABLED: "true"
  WEBSOCKET_ADDRESS: 0.0.0.0
  WEBSOCKET_PORT: 3012

  # SMTP (for email verification)
  SMTP_HOST: smtp.example.com
  SMTP_FROM: vaultwarden@example.com
  SMTP_PORT: 587
  SMTP_SECURITY: starttls
  SMTP_USERNAME: smtp_user
  SMTP_PASSWORD: smtp_password
```

Organization Management

Configure organization settings for team password sharing:

```
environment:
  ORG_CREATION_USERS: "admin@domain.com"
  ORG_NAME: "Company Name"
  ORG_OWNER_EMAIL: "admin@domain.com"
```

Monitoring and Logging

Health Checks

```
# Add health check to docker-compose.yml
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost:80/alive"]
  interval: 30s
  timeout: 10s
  retries: 3
  start_period: 30s
```

Logging Configuration

```
environment:
  LOG_LEVEL: info
  LOG_FILE: /data/vaultwarden.log
  EXTENDED_LOGGING: "true"
  LOG_TIMESTAMP: "true"
```

Backup and Recovery

Data Backup

Vaultwarden data is stored in the mounted volume:

```
# Backup Vaultwarden data
tar -czf vaultwarden-backup-$(date +%Y%m%d).tar.gz -C /opt/speedbits/vaultwarden/data .

# Backup configuration
cp /opt/speedbits/vaultwarden/docker-compose.yml /backup/vaultwarden-compose.yml
cp /opt/speedbits/vaultwarden/admin_token.txt /backup/vaultwarden-admin-token.txt
```

Disaster Recovery

```
# Restore from backup
tar -xzf vaultwarden-backup-20241201.tar.gz -C /opt/speedbits/vaultwarden/data/

# Restart service
cd /opt/speedbits/vaultwarden
```

```
docker compose down
docker compose up -d
```

Performance Optimization

Resource Limits

```
services:
  vaultwarden:
    deploy:
      resources:
        limits:
          memory: 512M
          cpus: '0.5'
        reservations:
          memory: 256M
          cpus: '0.25'
```

Database Optimization

For PostgreSQL, configure connection pooling:

```
environment:
  DATABASE_URL:
postgresql://vaultwarden:password@postgres:5432/vaultwarden?sslmode=require&max_connections=20
```

Troubleshooting

Common Issues

WebSocket Connection Failures:

```
# Check WebSocket configuration
docker logs vaultwarden | grep -i websocket

# Verify Traefik routing
```

```
curl -H "Host: vault.domain.com" http://localhost/notifications/hub
```

Database Connection Issues:

```
# Check database file permissions
ls -la /opt/speedbits/vaultwarden/data/db.sqlite3

# Verify database integrity
sqlite3 /opt/speedbits/vaultwarden/data/db.sqlite3 "PRAGMA integrity_check;"
```

SSL Certificate Problems:

```
# Check Traefik certificate status
docker logs traefik | grep -i acme

# Verify domain resolution
dig vault.domain.com
nslookup vault.domain.com
```

Debug Commands

```
# Container status
docker ps | grep vaultwarden

# Container logs
docker logs vaultwarden

# Container exec
docker exec -it vaultwarden /bin/sh

# Network connectivity
docker network inspect proxy

# Port binding
ss -tulnp | grep :443
```

Integration with Other Services

Borgmatic Backup Integration

Include Vaultwarden in automated backups:

```
# Add to borgmatic configuration
locations:
  directories:
    - /opt/speedbits/vaultwarden/data

# Exclude temporary files
exclude_patterns:
  - "*.tmp"
  - "*.log"
```

Monitoring Integration

Add Vaultwarden to monitoring systems:

```
# Health check endpoint
curl -f https://vault.domain.com/alive

# Metrics endpoint (if enabled)
curl https://vault.domain.com/metrics
```

Security Best Practices

Access Control

- Disable open registration in production
- Use strong admin tokens
- Implement IP whitelisting for admin access
- Enable two-factor authentication for all users

Network Security

- Use Traefik for SSL termination
- Implement rate limiting
- Configure fail2ban for brute force protection

- Regular security updates

Next Steps

With Vaultwarden installed and configured, you can now:

- Configure user accounts and organizations
- Set up SMTP for email notifications
- Implement backup strategies
- Integrate with existing identity providers

Verification Checklist

- [ ] Vaultwarden container running and healthy
- [ ] Web vault accessible via HTTPS
- [ ] Admin panel accessible with admin token
- [ ] WebSocket connections working
- [ ] Database accessible and writable
- [ ] SSL certificates valid

Next: Application Deployment and Management (Coming Soon)

9: Passbolt - Team Password Management

Passbolt is an OpenPGP-based, self-hosted team password manager with strong security properties and a browser-extension-centric UX. For comprehensive configuration, hardening guidance, and usage documentation, see the official Passbolt documentation.

Prerequisites

- Traefik installed (Chapter 4) and domain configured (Chapter 4.5)
- Docker running (Chapter 3)
- Borgmatic installed (Chapter 5) for automated backups
- Subdomain ready, e.g., `pass.example.com`

Installation via Infinity Tools

Menu Installation

```
APPLICATIONS → Passbolt → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-passbolt.sh --install
# or with environment variables
export PB_DOMAIN="pass.example.com"
sudo -E bash /opt/InfinityTools/Solutions/setup-passbolt.sh --install
```

Configuration Overview

- **Deployment modes:** Traefik (recommended), standalone HTTPS (self-signed), or HTTP
- **Database:** MariaDB 10.11 managed alongside Passbolt
- **Data paths:** `/opt/speedbits/passbolt/` (GPG keys, JWT keys, DB data)

- **Version pinning:** Passbolt image version pinned for stability

Environment Parameters (examples)

```
# SSL + domain
export PB_DOMAIN="pass.example.com"        # FQDN for Passbolt
# Networking
export PROXY_NETWORK="proxy"               # Traefik network name
```

What the Installer Sets Up

- Creates directories under `/opt/speedbits/passbolt/` (GPG, jwt, db_data)
- Generates secure database credentials (`db_password.txt`)
- Runs Passbolt + MariaDB containers
- Configures Traefik labels for HTTPS routing (if selected)
- Outputs access URLs and image versions on completion

Post-Install Steps

1. Open the web UI: `https://pass.example.com`
2. Follow the onboarding to create the first admin user
3. Install the Passbolt browser extension (Chrome/Firefox) when prompted
4. Configure SMTP in the Passbolt UI for email notifications

Backup & Restore

- Passbolt data (GPG, jwt) and database live under `/opt/speedbits/passbolt/`
- Borgmatic file backups include `/opt/speedbits/` by default
- Database dumps are included in the high-frequency DB backups

Operational Checks

```
# Check container states
sudo docker ps | egrep 'passbolt|passbolt-db'


# View logs
```

```
sudo docker logs passbolt --since 10m
sudo docker logs passbolt-db --since 10m

# Show current config hints (paths)
ls -la /opt/speedbits/passbolt/
```

Troubleshooting

SSL / Routing

```
# Verify Traefik is running
sudo docker ps | grep traefik

# Check ACME events
sudo docker logs traefik | grep -i acme

# Confirm DNS
dig +short pass.example.com
```

Database Connectivity

```
# Check DB container
sudo docker logs passbolt-db --since 10m

# Exec into DB and test
sudo docker exec -it passbolt-db mysql -u passbolt -p
```

Passbolt Health

```
# Application logs
sudo docker logs passbolt --since 10m

# Restart services
cd /opt/speedbits/passbolt && sudo docker compose down && sudo docker compose up -d
```

Security Notes

- Restrict admin access to known IPs (Traefik middleware optional)
- Rotate database and JWT keys as part of change management
- Ensure regular backups and test restoration

Verification

- □ Web UI reachable via HTTPS
- □ First admin created and logged in
- □ Browser extension paired
- □ Borgmatic backups succeeding

For advanced configuration (SMTP, LDAP/SSO, security hardening), consult the official Passbolt documentation.

10: Syncthing - File Synchronization

Syncthing provides continuous, peer-to-peer file synchronization across devices. For configuration reference and advanced topics (relays, discovery, ignore patterns), see the official Syncthing documentation.

Dependency check

- Required: **Docker** (Chapter 3)
- Optional: **Traefik** (Chapter 4) + subdomain (Chapter 4.5) for HTTPS/UI exposure
- Optional: **Borgmatic** (Chapter 6) for file data protection

Prerequisites

- □ Docker running (Chapter 3)
- □ Optional: Traefik installed (Chapter 4) + subdomain (Chapter 4.5), e.g., `sync.example.com`
- □ Optional: Borgmatic installed (Chapter 6) for backups

Installation via Infinity Tools

Menu Installation

```
□□APPLICATIONS → Syncthing → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-syncthing.sh --install
```

Traefik Integration

Select Traefik mode to expose the web UI via HTTPS and a domain:

```
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.syncthing.rule=Host(`sync.example.com`)"
  - "traefik.http.routers.syncthing.entrypoints=websecure"
  - "traefik.http.routers.syncthing.tls.certresolver=myresolver"
  - "traefik.http.services.syncthing.loadbalancer.server.port=8384"  # Syncthing GUI

networks:
  - proxy
```

Standalone Access

Without Traefik, map the GUI port directly and use self-signed HTTPS:

```
ports:
  - "8384:8384"   # Web GUI (HTTPS)
  # Syncthing protocol ports are internal to the container; peers connect via relay/UPnP/NAT traversal
```

Data & Configuration

```
volumes:
  - /opt/speedbits/syncthing/config:/var/syncthing
  - /opt/speedbits/syncthing/Documents:/sync/Documents
  - /opt/speedbits/syncthing/Photos:/sync/Photos
```

Security & Hardening

- Set GUI credentials (Settings → GUI)
- Restrict GUI to 0.0.0.0 only when behind Traefik; otherwise bind locally and reverse-proxy
- Consider IP allowlists via Traefik middleware for WAN exposure

Device Pairing

1. Retrieve Device ID from each peer (web UI → Actions → Show ID)
2. Add remote devices by ID; accept on the peer
3. Share specific folders with your peer device

Ignore Patterns

Use `.stignore` to exclude files:

```
# /opt/speedbits/syncthing/Documents/.stignore
*.tmp
.cache/
node_modules/
```

Operational Checks

```
# Containers
sudo docker ps | grep syncthing

# Logs
sudo docker logs syncthing --since 10m
```

Troubleshooting

- Ensure DNS resolves `sync.example.com` if using Traefik
- Confirm the GUI is reachable via Traefik or mapped port
- Validate folder permissions (UID/GID 1000 by default in LinuxServer images)

For advanced networking (relay servers, global discovery, NAT traversal), consult the Syncthing docs.

11: Nextcloud - Cloud Storage Platform

Nextcloud is a full-featured, self-hosted collaboration and file storage platform. It provides file sync and share, WebDAV, CalDAV/CardDAV, and a rich app ecosystem. For full configuration details and the admin manual, see the official Nextcloud documentation.

Architecture Overview

- **Core services:** Apache/PHP application with PostgreSQL database
- **Protocols:** WebDAV, CalDAV, CardDAV
- **Identity:** Local users; supports SSO/OIDC via apps
- **Networking:** Traefik reverse proxy (recommended) or standalone
- **Data:** Application files + user data volume + database

Resource Requirements

- Minimum: 1 vCPU, 512 MB RAM, 10 GB disk (grows with user data)
- Recommended: 2+ vCPU, 2 GB+ RAM, 20 GB+ disk

Prerequisites

- **Traefik installed** (Chapter 4) with Let's Encrypt
- **Docker installed** (Chapter 3)
- **Apprise installed** (Chapter 5) for notifications
- **Borgmatic installed** (Chapter 6) for automated backups
- **Domain configured** (Chapter 4.5) for production HTTPS

Interdependencies: The PostgreSQL service is attached to a `borgmatic-db` network for backup discovery. Borgmatic relies on Apprise for notifications.

Installation Methods

Via Infinity Tools Menu

```
□□APPLICATIONS → Nextcloud → Install
```

Command Line

```
# Show current status (no changes)
sudo bash /opt/InfinityTools/Solutions/setup-nextcloud.sh

# Run interactive installation
sudo bash /opt/InfinityTools/Solutions/setup-nextcloud.sh --install
```

Configuration Parameters

- **SSL Mode:** Traefik (HTTPS, recommended) or standalone (HTTP or self-signed HTTPS)
- **Domain:** Required for Traefik (e.g., `cloud.example.com`)
- **Standalone Port:** If not using Traefik
- **Default Quota:** Per-user storage limit in GB (recommended)
- **Credentials:** Admin and DB passwords are generated and stored in `/opt/speedbits/nextcloud/.env`

Generated Files & Directories

- `/opt/speedbits/nextcloud/.env` — Installation parameters and credentials
- `/opt/speedbits/nextcloud/docker-compose.yml` — Service definition
- `/opt/speedbits/nextcloud/html` — App files
- `/opt/speedbits/nextcloud/data` — User data
- `/opt/speedbits/nextcloud/db` — PostgreSQL data

Compose (Traefik Mode - Highlights)

```
services:
  db:
    image: postgres:${DB_VERSION}
    networks: [ ${NETWORK}, borgmatic-db ]
```

```
  nextcloud:
    image: nextcloud:${NEXTCLOUD_VERSION}
    environment:
      POSTGRES_HOST: nextcloud-db
      NEXTCLOUD_ADMIN_USER: ${NEXTCLOUD_ADMIN_USER}
      NEXTCLOUD_ADMIN_PASSWORD: ${NEXTCLOUD_ADMIN_PASSWORD}
      NEXTCLOUD_TRUSTED_DOMAINS: ${DOMAIN}
      OVERWRITEPROTOCOL: https
      OVERWRITEHOST: ${DOMAIN}
      PHP_UPLOAD_LIMIT: 16G
      PHP_MEMORY_LIMIT: 512M
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.nextcloud.rule=Host(`${DOMAIN}`)"
      - "traefik.http.routers.nextcloud.entrypoints=websecure"
      - "traefik.http.routers.nextcloud.tls.certresolver=myresolver"
      - "traefik.http.services.nextcloud.loadbalancer.server.port=80"
      - "traefik.http.middlewares.nextcloud-redirectregex.redirectRegex.permanent=true"
      - "traefik.http.middlewares.nextcloud-
redirectregex.redirectRegex.regex=https://(.*)/.well-known/(card|cal)dav"
      - "traefik.http.middlewares.nextcloud-
redirectregex.redirectRegex.replacement=https://$${1}/remote.php/dav/"
      - "traefik.http.middlewares.nextcloud-security.headers.customResponseHeaders.X-Content-
Type-Options=nosniff"
      - "traefik.http.middlewares.nextcloud-security.headers.customResponseHeaders.X-Frame-
Options=SAMEORIGIN"
      - "traefik.http.middlewares.nextcloud-security.headers.customResponseHeaders.X-XSS-
Protection=1; mode=block"
      - "traefik.http.routers.nextcloud.middlewares=nextcloud-redirectregex,nextcloud-
security"
```

Post-Install Hardening & Tasks

- Trusted domains and overwrite settings are applied automatically (Traefik mode)
- Default quota is applied if configured
- Security hardening executed: brute-force protection, file locking, log level
- Background jobs switched to **Cron**

Cron Setup

```
*/5 * * * * docker exec -u www-data nextcloud php -f /var/www/html/cron.php
```

Backup Integration (Borgmatic)

- Database container is auto-registered with Borgmatic (if available)
- Include these paths in backups:
  - `/opt/speedbits/nextcloud/data` — User files
  - `/opt/speedbits/nextcloud/db` — Database volume
  - `/opt/speedbits/nextcloud/config` (if present) — Config overrides
- Ensure Apprise is configured for notifications

Operations

```
# Logs
docker logs nextcloud

# Restart
cd /opt/speedbits/nextcloud && docker compose restart

# Update
cd /opt/speedbits/nextcloud && docker compose pull && docker compose up -d

# OCC (run as www-data)
docker exec -u www-data nextcloud php occ status
docker exec -u www-data nextcloud php occ app:list
```

Troubleshooting

- **SSL/Domain:** Verify Traefik routing, DNS A/AAAA, and ACME logs
- **Database:** Check container health; confirm credentials in `.env`
- **Storage:** Monitor `df -h /opt/speedbits/nextcloud`; enforce quotas
- **Trusted Domains:** `occ config:system:set trusted_domains 1 --value="cloud.example.com"`
- **Background Jobs:** Confirm cron runs; see `Settings → Basic settings`

Security Best Practices

- Enable 2FA for admin and users
- Enforce sane quotas to prevent disk exhaustion
- Regularly apply updates and review logs
- Restrict admin access and consider IP allowlists
- Review app permissions and disable unused apps

Verification Checklist

- [ ] Nextcloud and database containers running and healthy
- [ ] HTTPS reachable via Traefik with valid certificate
- [ ] Admin login works; quotas visible under Users
- [ ] Cron executing background jobs
- [ ] Backups configured and tested

References

- Nextcloud Admin Manual
- Nextcloud desktop & mobile clients

12: WordPress - Production-Ready Setup

WordPress is a widely used CMS for websites and blogs. This guide covers installation and runtime specifics when deploying via Infinity Tools. For platform usage, administration, and theme/plugin development, refer to the official WordPress documentation.

Architecture Overview

- **Services:** WordPress (PHP/Apache), MariaDB 10.11, optional Redis cache
- **Reverse Proxy:** Traefik with Let's Encrypt (recommended) or standalone
- **Networks:** Traefik proxy network + `borgmatic-db` for DB backups
- **Multi-instance:** Supported via `--instance=<name>`

Prerequisites

- **Traefik installed** (Chapter 4)
- **Docker installed** (Chapter 3)
- **Apprise installed** (Chapter 5) for notifications
- **Borgmatic installed** (Chapter 6) for automated backups
- **Domain configured** (Chapter 4.5) for HTTPS

Interdependencies: MariaDB is joined to `borgmatic-db` for backup discovery. Borgmatic depends on Apprise for notifications.

Installation Methods

Via Infinity Tools Menu

```
APPLICATIONS → WordPress → Install
```

Command Line

```
# Status (no changes)
sudo bash /opt/InfinityTools/Solutions/setup-wordpress.sh --status

# Default instance (interactive)
sudo bash /opt/InfinityTools/Solutions/setup-wordpress.sh --install

# Named instance
sudo bash /opt/InfinityTools/Solutions/setup-wordpress.sh --install --instance=blog2
```

Key Configuration

- **SSL Mode:** Traefik (HTTPS) or standalone (HTTP or self-signed HTTPS)
- **Domain:** Required for Traefik (e.g., `myblog.com`)
- **Standalone Port:** Required if not using Traefik
- **Redis Cache:** Optional. Can be enabled during install
- **Multi-Instance Paths:**
  - Default: `/opt/speedbits/wordpress`
  - Instance `blog2`: `/opt/speedbits/wordpress-blog2`

Generated Files & Directories

- `$WP_DIR/wp_data/` — WordPress files
- `$WP_DIR/db_data/` — MariaDB data
- `$WP_DIR/redis_data/` — Redis persistence (if enabled)
- `$WP_DIR/docker-compose.yml` — Service definition
- `$WP_DIR/php/uploads.ini` — PHP upload limits (50MB)
- `$WP_DIR/db_password.txt` — Generated DB password

Traefik Mode (Highlights)

```
services:
  db:
    image: mariadb:10.11
    networks: [ ${NETWORK}, borgmatic-db ]

  redis:                # if enabled
    image: redis:7-alpine
```

```
    command: redis-server --maxmemory 64mb --maxmemory-policy allkeys-lru

  wordpress:
    image: wordpress:latest
    environment:
      WORDPRESS_DB_HOST: wp-db:3306
      WORDPRESS_DB_USER: wpuser
      WORDPRESS_DB_PASSWORD: ${FROM_FILE}
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_TABLE_PREFIX: wp_
      WORDPRESS_CONFIG_EXTRA: |
        define('DISALLOW_FILE_EDIT', true);
        define('FORCE_SSL_ADMIN', true);
        define('WP_MEMORY_LIMIT', '512M');
        define('WP_MAX_MEMORY_LIMIT', '1024M');
        define('WP_CACHE', true);
        define('WP_POST_REVISIONS', 10);
        define('AUTOSAVE_INTERVAL', 300);
        define('WP_IMAGE_EDITORS', ['WP_Image_Editor_GD']);
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.${ROUTER_NAME}.rule=Host(`${DOMAIN}`)"
      - "traefik.http.routers.${ROUTER_NAME}.entrypoints=websecure"
      - "traefik.http.routers.${ROUTER_NAME}.tls.certresolver=myresolver"
      - "traefik.http.routers.${ROUTER_NAME}-www.rule=Host(`www.${DOMAIN}`)"
      - "traefik.http.routers.${ROUTER_NAME}-www.middlewares=${ROUTER_NAME}-redirect"
      - "traefik.http.middlewares.${ROUTER_NAME}-
redirect.redirectregex.regex=^https://www\\.${DOMAIN}/(.*)"
      - "traefik.http.middlewares.${ROUTER_NAME}-
redirect.redirectregex.replacement=https://${DOMAIN}/$${1}"
      - "traefik.http.middlewares.${ROUTER_NAME}-redirect.redirectregex.permanent=true"
```

Redis Object Cache (Recommended)

Enable Redis during installation (optional), then install the free **Redis Object Cache** plugin for significant performance gains.

- Dashboard → **Plugins → Add New**
- Search **Redis Object Cache** by Till Krüss

- Install → Activate → **Settings → Redis → Enable Object Cache**

[Redis Object Cache plugin (wordpress.org)]

Post-Install Hardening & Defaults

- File editing disabled; XML-RPC disabled
- OPcache enabled; memory limits tuned
- WWW → non-WWW redirect configured (Traefik)
- Let's Encrypt certificates via Traefik

Backup Integration (Borgmatic)

- MariaDB is auto-registered with Borgmatic (if available)
- Include in backups:
  - `$WP_DIR/wp_data`
  - `$WP_DIR/db_data`
  - `$WP_DIR/redis_data` (if enabled)
- Ensure Apprise notifications are configured

Operations

```
# Logs
docker logs wordpress
docker logs wp-db

# Restart
cd $WP_DIR && docker compose restart

# Update (safe)
cd $WP_DIR && docker compose pull && docker compose up -d

# Instance status
sudo bash /opt/InfinityTools/Solutions/setup-wordpress.sh --status

# Wipe and reinstall (destructive)
sudo bash /opt/InfinityTools/Solutions/setup-wordpress.sh --install --deleteall
```

Troubleshooting

- **Database connection errors:** verify `db_password.txt`, check `wp-db` logs, ensure volumes mounted
- **SSL/ACME issues:** check Traefik logs, DNS A/AAAA, ports 80/443 open
- **Performance:** enable Redis Object Cache; review plugins/themes bloat
- **Uploads:** default max upload 50MB via `$WP_DIR/php/uploads.ini`

Security Best Practices

- Keep core, themes, and plugins updated
- Limit admin accounts; enforce strong passwords and 2FA
- Review plugin footprint; remove unused components
- Regularly test backups and restore procedures

Verification Checklist

- [ ] WordPress, MariaDB (and Redis if used) containers running
- [ ] Site reachable over HTTPS with valid cert (Traefik)
- [ ] Admin login working; permalinks saved
- [ ] Redis Object Cache enabled (if configured)
- [ ] Backups configured and successful

13: Matomo - Web Analytics

Matomo (formerly Piwik) is a full-featured, self-hosted web analytics platform. This guide covers installation and runtime specifics when deploying via Infinity Tools. For comprehensive configuration and administration, see the official Matomo documentation.

Architecture Overview

- **Services:** Matomo (PHP/Apache) + MariaDB 10.11
- **Reverse Proxy:** Traefik with Let's Encrypt (recommended) or standalone
- **Networking:** Traefik proxy network + `borgmatic-db` for DB backups
- **Data:** App config/logs + database volume

Prerequisites

- **Traefik installed** (Chapter 4)
- **Docker installed** (Chapter 3)
- **Apprise installed** (Chapter 5) for notifications
- **Borgmatic installed** (Chapter 6) for automated backups
- **Domain configured** (Chapter 4.5) for HTTPS

Interdependencies: MariaDB joins `borgmatic-db` for backup discovery. Borgmatic relies on Apprise for notifications.

Installation Methods

Via Infinity Tools Menu

```
APPLICATIONS → Matomo → Install
```

Command Line

```
# Status (no changes)
sudo bash /opt/InfinityTools/Solutions/setup-matomo.sh --status
```

```
# Interactive installation
sudo bash /opt/InfinityTools/Solutions/setup-matomo.sh --install
```

Key Configuration

- **SSL Mode:** Traefik (HTTPS) or standalone (HTTP or self-signed HTTPS)
- **Domain:** Required for Traefik (e.g., `analytics.example.com`)
- **Standalone Port:** Required if not using Traefik
- **Database:** MariaDB 10.11 with tuned params

Generated Files & Directories

- `/opt/speedbits/matomo/.env` — Installation parameters and DB creds
- `/opt/speedbits/matomo/docker-compose.yml` — Service definition
- `/opt/speedbits/matomo/config` — Matomo config
- `/opt/speedbits/matomo/logs` — Matomo logs
- `/opt/speedbits/matomo/db` — MariaDB data
- `/opt/speedbits/matomo/db_password.txt` — DB password (root-only)

Compose (Traefik Mode - Highlights)

```
services:
  db:
    image: mariadb:${DB_VERSION}
    networks: [ ${NETWORK}, borgmatic-db ]
    command: >
      --max-allowed-packet=64M
      --innodb-buffer-pool-size=512M

  matomo:
    image: matomo:${MATOMO_VERSION}
    environment:
      MATOMO_DATABASE_HOST: matomo-db
      MATOMO_DATABASE_ADAPTER: mysql
      MATOMO_DATABASE_TABLES_PREFIX: matomo_
      MATOMO_DATABASE_USERNAME: ${MYSQL_USER}
```

```
      MATOMO_DATABASE_PASSWORD: ${MYSQL_PASSWORD}
      MATOMO_DATABASE_DBNAME: ${MYSQL_DATABASE}
      PHP_MEMORY_LIMIT: 512M
    volumes:
      - ./config:/var/www/html/config
      - ./logs:/var/www/html/logs
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.matomo.rule=Host(`${DOMAIN}`)"
      - "traefik.http.routers.matomo.entrypoints=websecure"
      - "traefik.http.routers.matomo.tls.certresolver=myresolver"
      - "traefik.http.services.matomo.loadbalancer.server.port=80"
```

Post?Install Tasks

- Complete the Matomo setup wizard in the browser (system check → DB → admin user → first website)
- Add the JavaScript tracking code to your site(s) (before `</head>`)
- Set up cron archiving (see below)

Cron Archiving

```
*/5 * * * * docker exec matomo /usr/local/bin/php /var/www/html/console core:archive
>/dev/null 2>&1
```

Backup Integration (Borgmatic)

- Database container is auto-registered with Borgmatic (if available)
- Include in backups:
  - `/opt/speedbits/matomo/db` — MariaDB data
  - `/opt/speedbits/matomo/config` — App configuration
  - `/opt/speedbits/matomo/logs` — Optional
- Ensure Apprise notifications are configured

Operations

```
# Logs
docker logs matomo
docker logs matomo-db

# Restart
cd /opt/speedbits/matomo && docker compose restart

# Update
cd /opt/speedbits/matomo && docker compose pull && docker compose up -d
```

Troubleshooting

- **DB connectivity:** verify `db_password.txt`, check `matomo-db` logs
- **SSL/ACME:** check Traefik logs, DNS, ports 80/443 open
- **Performance:** tune archiving frequency; ensure DB buffer pool sizing matches workload

Security Best Practices

- Restrict admin access; use strong credentials
- Regular updates; monitor logs
- Consider IP anonymization and consent tools for compliance

Verification Checklist

- ☐ Matomo and database containers running
- ☐ HTTPS reachable via Traefik with valid certificate
- ☐ Setup wizard completed; admin login working
- ☐ Cron archiving running
- ☐ Backups configured and tested

References

- Matomo Documentation

14: Webmin - System Administration Platform

Webmin provides a web-based system administration interface for Linux servers. It offers user management, service control, file system access, package management, network configuration, and system monitoring through a unified web UI. For module documentation, API details, and advanced configuration, see the official Webmin documentation.

Prerequisites

- □ **Docker installed** (Chapter 3)
- □ **Docker Compose** (Chapter 3)
- □ **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- □ **Optional: Domain configured** (Chapter 4.5), e.g., `webmin.example.com`

Installation via Infinity Tools

Menu Installation

```
□□APPLICATIONS → Webmin → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-webmin.sh --install

# With domain (Traefik mode)
export WEBMIN_USE_TRAEFIK=true
export WEBMIN_DOMAIN="webmin.example.com"
sudo -E bash /opt/InfinityTools/Solutions/setup-webmin.sh --install

# With host filesystem access (read-only)
export WEBMIN_HOST_ACCESS=readonly
sudo -E bash /opt/InfinityTools/Solutions/setup-webmin.sh --install
```

```
# With host filesystem access (read-write)
export WEBMIN_HOST_ACCESS=readwrite
sudo -E bash /opt/InfinityTools/Solutions/setup-webmin.sh --install

# Custom port (standalone mode)
sudo bash /opt/InfinityTools/Solutions/setup-webmin.sh --install 9443
```

Deployment Modes

Traefik Mode

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://webmin.example.com`
- Direct web access (no SSH tunnel required)
- Requires: Traefik running, DNS A record configured

Standalone Mode (Recommended)

Direct HTTPS access with self-signed certificate, accessed via SSH tunnel:

- Access via: `https://localhost:8443` (after SSH tunnel)
- SSH tunnel command: `ssh -L 8443:localhost:10000 user@server`
- Self-signed SSL (browser warning on first access)
- More secure (not directly exposed to internet)
- Default port: 8443 (configurable)

Host Filesystem Access Configuration

During installation, you'll be prompted for host filesystem access level:

- **None (default)** - Container filesystem only, no host access
- **Read-Only** - Host filesystem mounted at `/host/` (read-only)
- **Read-Write** - Host filesystem mounted at `/host/` (full access)

Volume Mount:

- Read-Only: `- /:/host:ro`
- Read-Write: `- /:/host`

Installation Process

Configuration Steps

1. **SSL Mode Selection:** Choose Traefik or Standalone
2. **If Traefik:** Provide domain name
3. **If Standalone:** Specify HTTPS port (default: 8443)
4. **Host Access:** Choose filesystem access level
5. **User Creation:** System user `webminadmin` created with random password

What Gets Created

- **Directory:** `/opt/speedbits/webmin`
- **Container:** `webmin` (johanp/webmin:latest)
- **Volumes:** `webmin-config`, `webmin-logs`
- **Docker Compose:** `/opt/speedbits/webmin/docker-compose.yml`
- **System User:** `webminadmin` with sudo privileges
- **Network:** Joins Traefik network (Traefik mode) or creates internal network (Standalone)

Access Methods

Traefik Mode

```
https://webmin.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode (SSH Tunnel)

On local machine:

```
ssh -L 8443:localhost:10000 user@server-ip
```

Then in browser:

```
https://localhost:8443
```

Accept self-signed certificate warning (Advanced → Proceed).

Alternative: Direct Container IP

```
# Get container IP
CONTAINER_IP=$(docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
webmin)

# SSH tunnel to container IP
ssh -L 8443:$CONTAINER_IP:10000 user@server-ip
```

Authentication

Default Credentials

- **Username:** `webminadmin`
- **Password:** Randomly generated (shown once during installation)

⚠ **CRITICAL:** Password is displayed only once. Save it immediately!

System Authentication

- Any user in `sudo` or `wheel` group can login
- Uses system `/etc/passwd`, `/etc/shadow`, `/etc/group`
- Container runs as root with privileged access
- PAM authentication configured (can be disabled for internal password file)

File Manager: Container vs Host Filesystem

 CRITICAL: Webmin's File Manager initially shows the container's filesystem, not the host system.

Accessing Host Files

1. Navigate to **Other → File Manager**
2. You'll see container filesystem (minimal, typically empty)
3. **To access host:** Type `/host/` in the path bar
4. Press Enter or click "Go"
5. Host filesystem is now accessible

Path Mapping

- **Container root:** `/` (Webmin container filesystem)
- **Host root:** `/host/` (mounted host filesystem)

Common Host Paths

- `/host/etc/` - Host system configuration
- `/host/home/` - Host user home directories
- `/host/opt/` - Host application data (Infinity Tools, Speedbits)
- `/host/var/log/` - Host system logs
- `/host/var/www/` - Host web directories

Access Level Behavior

- **Read-Only:** Can browse `/host/` but cannot modify files
- **Read-Write:** Full access to `/host/` (create, edit, delete)
- **None:** `/host/` directory doesn't exist

Key Features

System Administration

- User and group management
- Service management (systemd units)
- Package management (apt/yum)
- System information and monitoring
- Log file viewing

File Management

- Web-based file browser
- Text editor for configuration files
- File upload/download

- Permission management
- **Note:** Use `/host/` path for host filesystem access

Network Configuration

- Network interface configuration
- Firewall rules (iptables/ufw)
- DNS configuration
- Port forwarding

Security Configuration

Access Security

- SSH tunnel recommended for standalone mode (not directly exposed)
- Traefik mode uses Let's Encrypt SSL (production-ready)
- Self-signed certificate in standalone mode (acceptable for tunneled access)
- System authentication via PAM or internal password file

Filesystem Security

- **Read-Only:** Safe for viewing host files without modification risk
- **Read-Write:** Full access - use with caution, audit changes
- **None:** Most secure - container filesystem only

Container Security

- Container runs as root (required for system management)
- Privileged mode enabled (required for system access)
- Mounts host user databases (passwd, shadow, group)
- Optional host filesystem mount

Configuration Persistence

- **Config Volume:** `webmin-config` persists Webmin settings
- **Logs Volume:** `webmin-logs` persists log files
- **Data Directory:** `/opt/speedbits/webmin` contains docker-compose.yml
- All settings survive container restarts

Troubleshooting

Container Not Starting

```
docker logs webmin
docker ps -a | grep webmin
```

Authentication Issues

- Verify user exists: `docker exec webmin grep webminadmin /etc/passwd`
- Check sudo group: `groups webminadmin`
- Verify password file: `docker exec webmin cat /etc/webmin/miniserv.users`
- Restart container: `docker restart webmin`

File Manager Issues

- **Can't see host files:** Navigate to `/host/` in path bar
- **/host/ doesn't exist:** Host access was set to "None" during installation
- **Read-only errors:** Host access was set to "Read-Only"
- **To change access:** Reinstall with different `WEBMIN_HOST_ACCESS` setting

SSH Tunnel Issues

- Verify SSH access: `ssh user@server-ip`
- Check port matches installation: Default is 8443
- Try container IP: `ssh -L 8443:CONTAINER_IP:10000 user@server`
- Verify Webmin is running: `docker ps | grep webmin`

Traefik Routing Issues

- Verify Traefik is running: `docker ps | grep traefik`
- Check DNS resolution: `dig webmin.example.com`
- Verify SSL certificate: Check Traefik logs
- Wait 30-60 seconds after installation for certificate generation

Production Considerations

- **Access Method:** SSH tunnel (standalone) is more secure than direct web access

- **Filesystem Access:** Use "Read-Only" unless file editing is required
- **Password Management:** Store credentials in password manager (Vaultwarden)
- **User Management:** Create separate Webmin users for team members
- **Audit Logging:** Monitor Webmin access logs for security
- **Backup:** Backup Webmin configuration volume regularly

Integration with Infinity Tools

Webmin complements Infinity Tools by providing:

- Visual system administration alongside containerized applications
- File management for host system (via `/host/` mount)
- User management for system users
- Service monitoring and management

Note: Infinity Tools applications run in Docker containers. Webmin provides host system management. Use Portainer (Chapter 13) for Docker container management.

Next Steps

Webmin is now operational. Use it to:

- Manage system users and permissions
- Browse and edit host filesystem (remember `/host/` path!)
- Monitor system resources
- Configure system services
- View system logs

For advanced Webmin features, module development, and enterprise capabilities, refer to the official Webmin documentation.

15: BookStack - Documentation Platform / Wiki

BookStack is a self-hosted documentation and wiki platform built with PHP and Laravel. It provides a hierarchical content structure (Books → Chapters → Pages), WYSIWYG editing with Markdown support, full-text search, user roles, and export capabilities. For API documentation, advanced customization, and development guides, see the official BookStack documentation.

Prerequisites

- □ **Docker installed** (Chapter 3)
- □ **Docker Compose** (Chapter 3)
- □ **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- □ **Optional: Domain configured** (Chapter 4.5), e.g., `docs.example.com`

Installation via Infinity Tools

Menu Installation

```
□□APPLICATIONS → BookStack → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-bookstack.sh --install

# With domain (Traefik mode)
export BS_DOMAIN="docs.example.com"
export BS_EMAIL="admin@example.com"
sudo -E bash /opt/InfinityTools/Solutions/setup-bookstack.sh --install

# Standalone mode with custom port
export BS_USE_TRAEFIK=false
export BS_PORT=8092
```

```
sudo -E bash /opt/InfinityTools/Solutions/setup-bookstack.sh --install

# Fresh install (delete all data)
sudo bash /opt/InfinityTools/Solutions/setup-bookstack.sh --install --deleteall
```

Deployment Modes

Traefik Mode (Default)

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://docs.example.com`
- HTTP → HTTPS redirect configured
- Security headers (X-Frame-Options, CSP, etc.)
- Requires: Traefik running, DNS A record configured

Standalone Mode

Direct access with optional HTTPS (self-signed):

- HTTP: `http://SERVER_IP:8092`
- HTTPS: `https://SERVER_IP:8092` (self-signed cert via nginx proxy)
- Default port: 8092 (configurable)
- No domain required

Architecture

Containers

- **bookstack** - Main BookStack application (lscr.io/linuxserver/bookstack:latest)
- **bookstack-db** - MariaDB 10.11 database
- **bookstack-ssl-proxy** - Nginx SSL proxy (standalone HTTPS mode only)

Data Persistence

- **Database:** `/opt/speedbits/bookstack/db_data/` (MariaDB data)
- **Config:** `/opt/speedbits/bookstack/config/` (BookStack config, uploads)

- **SSL:** `/opt/speedbits/bookstack/ssl/` (standalone mode certificates)

Networks

- **Traefik network:** Joins Traefik's proxy network (Traefik mode)
- **bookstack-internal:** Isolated bridge network (standalone mode)
- **borgmatic-db:** Database backup network (for Borgmatic integration)

Installation Process

Configuration Steps

1. **SSL Mode Selection:** Choose Traefik (default) or Standalone
2. **If Traefik:** Provide domain name and email
3. **If Standalone:** Specify port (default: 8092) and SSL mode
4. **Database Setup:** Random passwords generated and saved
5. **App Key:** Encryption key generated for BookStack

What Gets Created

- **Directory:** `/opt/speedbits/bookstack`
- **Containers:** `bookstack`, `bookstack-db`
- **Docker Compose:** `/opt/speedbits/bookstack/docker-compose.yml`
- **Password Files:** `db_password.txt`, `db_root_password.txt`, `app_key.txt`
- **Volumes:** Database data, config, SSL certificates

Database Configuration

Database Details

- **Type:** MariaDB 10.11
- **Database:** bookstack
- **User:** bookstack
- **Password:** Randomly generated, saved in `db_password.txt`
- **Root Password:** Saved in `db_root_password.txt`
- **Character Set:** utf8mb4
- **Collation:** utf8mb4_unicode_ci
- **Max Packet Size:** 256MB

Accessing Database

```
# View database password
cat /opt/speedbits/bookstack/db_password.txt

# Connect to database
docker exec -it bookstack-db mysql -u bookstack -p bookstack
# Enter password from db_password.txt

# Backup database
docker exec bookstack-db mysqldump -u bookstack -p bookstack > backup.sql
```

Access Methods

Traefik Mode

```
https://docs.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

HTTP:

```
http://SERVER_IP:8092
```

HTTPS:

```
https://SERVER_IP:8092
```

Accept self-signed certificate warning (Advanced → Proceed).

Authentication

Default Credentials

- **Email:** `admin@admin.com`

- **Password:** `password`

⚠ **CRITICAL:** Change these immediately after first login! These are public defaults.

User Roles

- **Admin** - Full system access, user management, settings
- **Editor** - Can create/edit content, manage own content
- **Viewer** - Read-only access
- **Guest** - Public access (if enabled)

Environment Variables

BookStack Container

- `APP_URL` - Base URL (https://docs.example.com or http://SERVER_IP:8092)
- `APP_KEY` - Encryption key (base64 encoded, auto-generated)
- `DB_HOST` - Database host (bookstack-db:3306)
- `DB_DATABASE` - Database name (bookstack)
- `DB_USERNAME` - Database user (bookstack)
- `DB_PASSWORD` - Database password (from db_password.txt)
- `MAIL_DRIVER` - Email driver (smtp)
- `MAIL_HOST` - SMTP server
- `MAIL_PORT` - SMTP port
- `MAIL_FROM` - From email address

Key Features

Content Management

- Hierarchical structure: Books → Chapters → Pages
- WYSIWYG editor with Markdown support
- Full-text search across all content
- Image uploads and attachments
- Version history and revisions
- Tags and categorization

User Management

- Role-based access control (RBAC)
- User invitations via email
- LDAP/Active Directory integration (advanced)
- OAuth providers (GitHub, Google, etc.)

Export & Integration

- Export to PDF, HTML, Markdown, Plain Text
- REST API for programmatic access
- Webhook support
- RSS feeds

Security Configuration

Access Security

- □ Traefik mode uses Let's Encrypt SSL (production-ready)
- □ Standalone HTTPS uses self-signed certificates (acceptable for internal use)
- □ Security headers configured (X-Frame-Options, CSP, etc.)
- □ Password hashing (bcrypt)
- ⚠ Change default admin credentials immediately

Container Security

- Runs as non-root user (PUID/PGID: 1000)
- Database isolated in separate container
- Network isolation (internal networks)
- Volume mounts for data persistence

Configuration Persistence

- **Database Volume:** `db_data` persists all content
- **Config Volume:** `config` persists settings and uploads
- **Password Files:** Saved in `/opt/speedbits/bookstack/`
- All settings survive container restarts

Backup & Restore

Backup Strategy

```
# Full backup
cd /opt/speedbits
tar czf bookstack-backup-$(date +%Y%m%d).tar.gz bookstack/

# Database-only backup
docker exec bookstack-db mysqldump -u bookstack -p bookstack > bookstack-db-$(date
+%Y%m%d).sql

# Config-only backup
tar czf bookstack-config-$(date +%Y%m%d).tar.gz -C /opt/speedbits/bookstack config/
```

Restore Process

1. Stop containers: `cd /opt/speedbits/bookstack && docker compose down`
2. Restore data: Extract backup to `/opt/speedbits/bookstack/`
3. Restore database: `docker exec -i bookstack-db mysql -u bookstack -p bookstack < backup.sql`
4. Start containers: `docker compose up -d`

Troubleshooting

Container Not Starting

```
docker logs bookstack
docker logs bookstack-db
docker ps -a | grep bookstack
```

Database Connection Issues

- Verify database is running: `docker ps | grep bookstack-db`
- Check database logs: `docker logs bookstack-db`
- Test connection: `docker exec bookstack-db mysqladmin ping -h localhost`
- Verify password: `cat /opt/speedbits/bookstack/db_password.txt`

SSL Certificate Issues

- **Traefik mode:** Check Traefik logs: `docker logs traefik`
- **Traefik mode:** Verify DNS: `dig docs.example.com`
- **Standalone mode:** Check nginx proxy logs: `docker logs bookstack-ssl-proxy`
- **Standalone mode:** Verify certificates: `ls -la /opt/speedbits/bookstack/ssl/`

Performance Issues

- Check container resources: `docker stats bookstack bookstack-db`
- Review database performance: `docker exec bookstack-db mysqladmin processlist`
- Optimize images before upload
- Consider database indexing for large content

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Password Policy:** Enforce strong passwords via user management
- **Backup Strategy:** Implement automated backups (integrate with Borgmatic)
- **User Management:** Use LDAP/OAuth for enterprise environments
- **Monitoring:** Monitor container health and resource usage
- **Updates:** Re-run install script periodically for updates
- **Email:** Configure SMTP for password resets and notifications

Integration with Infinity Tools

BookStack complements Infinity Tools by providing:

- Documentation platform for Infinity Tools guides
- Knowledge base for server configurations
- Team collaboration on documentation
- Export capabilities for offline access

Note: The Infinity Tools documentation sync script (`sync-bookstack.sh`) can automatically upload HTML documentation files to BookStack.

SMTP Configuration

After installation, you can configure SMTP for email functionality:

- Run SMTP config wizard: `sudo bash Infrastructure/bookstack-smtp-config.sh`
- Or edit `docker-compose.yml` and set `MAIL_HOST`, `MAIL_PORT`, etc.

- Restart container: `docker restart bookstack`

Next Steps

BookStack is now operational. Use it to:

- Create documentation for your infrastructure
- Organize knowledge in books and chapters
- Collaborate with team members
- Export content for offline access
- Integrate with other tools via API

For advanced features, API usage, custom themes, and development guides, refer to the official BookStack documentation.

16: Uptime Kuma - Monitoring & Status Pages

Uptime Kuma is a self-hosted monitoring solution built with Node.js. It provides uptime monitoring, incident tracking, status pages, and 90+ notification integrations. Supports HTTP(s), TCP, Ping, DNS, Docker containers, and more. For API documentation, advanced configuration, and development guides, see the official Uptime Kuma repository.

Prerequisites

- □ **Docker installed** (Chapter 3)
- □ **Docker Compose** (Chapter 3)
- □ **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- □ **Optional: Domain configured** (Chapter 4.5), e.g., `status.example.com`

Installation via Infinity Tools

Menu Installation

```
□□APPLICATIONS → Uptime Kuma → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-uptime-kuma.sh --install


# With domain (Traefik mode)
export BS_DOMAIN="status.example.com"
sudo -E bash /opt/InfinityTools/Solutions/setup-uptime-kuma.sh --install
```

Deployment Modes

Traefik Mode (Default)

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://status.example.com`
- Security headers configured
- Requires: Traefik running, DNS A record configured

Standalone Mode

Direct access with HTTP or HTTPS (self-signed):

- HTTP: `http://SERVER_IP:3001`
- HTTPS: `https://SERVER_IP:3001` (self-signed cert via nginx proxy)
- Default port: 3001 (configurable)
- No domain required

Architecture

Container

- **uptime-kuma** - Main application (louislam/uptime-kuma:1)
- **uptime-kuma-ssl-proxy** - Nginx SSL proxy (standalone HTTPS mode only)

Data Persistence

- **Data:** `/opt/speedbits/uptime-kuma/data/` (SQLite database, config)
- **SSL:** `/opt/speedbits/uptime-kuma/ssl/` (standalone mode certificates)

Networks

- **Traefik network:** Joins Traefik's proxy network (Traefik mode)
- **kuma-internal:** Isolated bridge network (standalone mode)

Docker Socket Access

Optional read-only access to `/var/run/docker.sock` for Docker container monitoring:

- Enables Docker container monitoring
- Read-only mount (security best practice)
- Configured during installation

Installation Process

Configuration Steps

1. **SSL Mode Selection:** Choose Traefik (default) or Standalone
2. **If Traefik:** Provide domain name
3. **If Standalone:** Specify port (default: 3001) and SSL mode
4. **Docker Monitoring:** Optional enable Docker socket access
5. **Timezone:** Optional timezone configuration (default: UTC)

What Gets Created

- **Directory:** `/opt/speedbits/uptime-kuma`
- **Container:** `uptime-kuma`
- **Docker Compose:** `/opt/speedbits/uptime-kuma/docker-compose.yml`
- **Data Volume:** SQLite database and configuration

Access Methods

Traefik Mode

```
https://status.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

HTTP:

```
http://SERVER_IP:3001
```

HTTPS:

```
https://SERVER_IP:3001
```

Accept self-signed certificate warning (Advanced → Proceed).

Authentication

First-Time Setup

- **No default credentials** - Admin account must be created on first access
- **Setup wizard:** "Create your admin account" appears on first visit
- **Password requirements:** Minimum 8 characters (12+ recommended)
- ⚠ **CRITICAL:** Write down credentials immediately - no password reset on first setup

Password Reset

```
docker exec -it uptime-kuma npm run reset-password
```

Follow prompts to enter username and new password.

Monitor Types

Supported Protocols

- **HTTP(s)** - Websites, APIs, webhooks
- **TCP** - Port monitoring (SSH, databases, etc.)
- **Ping (ICMP)** - Server availability
- **DNS** - DNS record monitoring
- **Docker Container** - Container health (requires Docker socket)
- **Keyword** - Content-based monitoring
- **SMTP** - Email server monitoring
- **gRPC** - gRPC service monitoring

Monitor Configuration

- **Check interval** - Frequency of checks (default: 60 seconds)
- **Retry attempts** - Number of retries before marking as down
- **Timeout** - Request timeout
- **Expected status codes** - HTTP status codes to consider "up"
- **Keyword detection** - Check for specific text in response

Notification Integrations

Supported Providers

- **Discord** - Webhook integration
- **Slack** - Webhook integration
- **Telegram** - Bot API
- **Email** - SMTP
- **Apprise** - Self-hosted Apprise (80+ services)
- **Webhooks** - Custom HTTP endpoints
- **90+ providers** - See full list in Uptime Kuma settings

Apprise Integration

If Apprise is installed (Chapter 5), use it for notifications:

- Type: **Apprise (Self-hosted)**
- URL: `http://apprise:8000/notify/{YOUR-KEY}`
- Enables access to all Apprise-supported services

Status Pages

Features

- Public status pages (no authentication required)
- Customizable appearance (colors, logo, theme)
- Monitor selection (choose which monitors to display)
- Incident history
- Uptime statistics
- RSS feed support

Use Cases

- Public service status (like status.github.com)
- Internal team dashboards
- Customer-facing status pages
- Service health transparency

Environment Variables

Uptime Kuma Container

- `TZ` - Timezone (default: UTC)

Data Storage

- SQLite database stored in `/app/data`
- Persisted via Docker volume mount
- Includes monitors, notifications, status pages, user accounts

Security Configuration

Access Security

- Traefik mode uses Let's Encrypt SSL (production-ready)
- Standalone HTTPS uses self-signed certificates (acceptable for internal use)
- Security headers configured (X-Frame-Options, CSP, etc.)
- Docker socket mounted read-only (if enabled)
- Security option: `no-new-privileges:true`

Container Security

- Runs as non-root user
- Read-only Docker socket access (if enabled)
- Network isolation
- Volume mounts for data persistence

Configuration Persistence

- **Data Volume:** `data` persists all configuration
- **SQLite Database:** Stored in data directory
- All monitors, notifications, and settings survive container restarts

Backup & Restore

Backup Strategy

```
# Full backup
cd /opt/speedbits
tar czf uptime-kuma-backup-$(date +%Y%m%d).tar.gz uptime-kuma/

# Using Uptime Kuma built-in backup
# Settings → Backup → Download Backup
```

Restore Process

1. Stop container: `cd /opt/speedbits/uptime-kuma && docker compose down`
2. Restore data: Extract backup to `/opt/speedbits/uptime-kuma/`
3. Start container: `docker compose up -d`

Troubleshooting

Container Not Starting

```
docker logs uptime-kuma
docker ps -a | grep uptime-kuma
```

SSL Certificate Issues

- **Traefik mode:** Check Traefik logs: `docker logs traefik`
- **Traefik mode:** Verify DNS: `dig status.example.com`
- **Standalone mode:** Check nginx proxy logs: `docker logs uptime-kuma-ssl-proxy`

Docker Monitoring Issues

- Verify Docker socket access: `docker exec uptime-kuma ls /var/run/docker.sock`
- Check container permissions
- Verify socket is mounted read-only

Monitor Not Responding

- Check monitor configuration (URL, port, etc.)

- Verify service is actually running
- Check network connectivity from container
- Review monitor logs in Uptime Kuma dashboard

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Password Policy:** Enforce strong passwords (12+ characters)
- **2FA:** Enable two-factor authentication (Settings → Security)
- **Backup Strategy:** Implement automated backups
- **Monitoring:** Monitor Uptime Kuma itself (meta-monitoring)
- **Notifications:** Configure multiple notification channels for redundancy
- **Status Pages:** Use public status pages instead of sharing admin access

Integration with Infinity Tools

Uptime Kuma complements Infinity Tools by providing:

- Monitoring for all Infinity Tools applications
- Docker container monitoring for infrastructure
- Status pages for public service transparency
- Notification integration with Apprise

Recommended Monitors:

- Traefik dashboard
- All Infinity Tools application endpoints
- Docker daemon health
- Server resources (via Netdata if installed)

API & Automation

REST API

- Uptime Kuma provides REST API for programmatic access
- API documentation available in web interface
- Useful for automation and integration

Webhooks

- Incoming webhooks for external integrations
- Outgoing webhooks for custom notifications
- Custom payload formatting

Next Steps

Uptime Kuma is now operational. Use it to:

- Monitor all Infinity Tools applications
- Track Docker container health
- Create public status pages
- Set up multi-channel notifications
- Track uptime statistics and incidents

For advanced features, API usage, custom themes, and development guides, refer to the official Uptime Kuma repository.

17: Netdata - Real-time Performance Monitoring

Netdata is a distributed, real-time performance monitoring solution built with C and Node.js. It provides sub-second granularity metrics collection, zero-configuration operation, and comprehensive system monitoring including CPU, memory, disk, network, processes, and Docker containers. For API documentation, advanced configuration, and development guides, see the official Netdata documentation.

Prerequisites

- □ **Docker installed** (Chapter 3)
- □ **Docker Compose** (Chapter 3)
- □ **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- □ **Optional: Domain configured** (Chapter 4.5), e.g., `monitor.example.com`
- □ **Optional: Apprise installed** (Chapter 5) for alert notifications

Installation via Infinity Tools

Menu Installation

```
□□APPLICATIONS → Netdata → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-netdata.sh --install
```

Deployment Modes

Traefik Mode (Default)

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://monitor.example.com`
- Security headers configured
- Requires: Traefik running, DNS A record configured

Standalone Mode

Direct access with HTTP or HTTPS (self-signed):

- HTTP: `http://SERVER_IP:19999`
- HTTPS: `https://SERVER_IP:19999` (self-signed cert via nginx proxy)
- Default port: 19999 (configurable)
- No domain required

Architecture

Container

- **netdata** - Main application (netdata/netdata:latest)
- **netdata-ssl-proxy** - Nginx SSL proxy (standalone HTTPS mode only)

Data Persistence

- **Config:** `/opt/speedbits/netdata-client/netdata/` (configuration files)
- **Lib:** `/opt/speedbits/netdata-client/netdata/lib/` (database, metrics)
- **Cache:** `/opt/speedbits/netdata-client/netdata/cache/` (temporary data)
- **SSL:** `/opt/speedbits/netdata-client/ssl/` (standalone mode certificates)

Host Access

Netdata requires access to host system for monitoring:

- `/proc` - Process and system information (read-only)
- `/sys` - System information (read-only)
- `/var/run/docker.sock` - Docker API (read-only, for container monitoring)
- `/etc/passwd`, `/etc/group` - User information (read-only)
- `/etc/os-release` - OS information (read-only)

Networks

- **Traefik network:** Joins Traefik's proxy network (Traefik mode)
- **netdata-internal:** Isolated bridge network (standalone mode)
- **borgmatic-db:** Network for Apprise integration (if enabled)

Installation Process

Configuration Steps

1. **SSL Mode Selection:** Choose Traefik (default) or Standalone
2. **If Traefik:** Provide domain name
3. **If Standalone:** Specify port (default: 19999) and SSL mode
4. **Streaming:** Optional parent-child streaming configuration
5. **Apprise Integration:** Optional alert notification setup

What Gets Created

- **Directory:** `/opt/speedbits/netdata-client`
- **Container:** `netdata`
- **Docker Compose:** `/opt/speedbits/netdata-client/docker-compose.yml`
- **Config Files:** Health alerts, streaming config, Apprise integration

Access Methods

Traefik Mode

```
https://monitor.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

HTTP:

```
http://SERVER_IP:19999
```

HTTPS:

```
https://SERVER_IP:19999
```

Accept self-signed certificate warning (Advanced → Proceed).

Security Configuration

Access Security

- Traefik mode uses Let's Encrypt SSL (production-ready)
- Standalone HTTPS uses self-signed certificates (acceptable for internal use)
- Security headers configured (X-Frame-Options, CSP, etc.)
- ⚠ **NO default authentication** - Dashboard is publicly accessible

Container Security

- Runs with `SYS_PTRACE` and `SYS_ADMIN` capabilities (required for monitoring)
- Security option: `apparmor:unconfined`
- Read-only mounts for host filesystem access
- Docker socket mounted read-only

Authentication

⚠ **CRITICAL:** Netdata has NO username/password protection by default!

- **Traefik mode:** Add Basic Auth via `websiteprotection.sh`
- **Standalone mode:** Keep on private network or use SSH tunnel
- **VPN:** Access via WireGuard VPN for secure remote access
- **SSH Tunnel:** `ssh -L 19999:localhost:19999 user@server`

Metrics Collection

System Metrics

- **CPU** - Per-core usage, load averages, interrupts
- **Memory** - RAM, swap, buffers, cache
- **Disk** - I/O, space usage, per-filesystem metrics
- **Network** - Per-interface traffic, connections, errors
- **Processes** - Per-process CPU, memory, I/O
- **System Load** - Load averages, context switches

Docker Metrics

- Auto-discovery of all containers
- Per-container CPU, memory, disk, network
- Container health status
- Real-time metrics (1-second granularity)

Data Retention

- Configurable retention period
- Default: 1 hour of 1-second data
- Can extend for longer historical data
- Low storage footprint (~50MB RAM)

Alert Configuration

Default Alerts

Pre-configured alerts in `health.d/`:

- **CPU Usage:** Warning > 80%, Critical > 95%
- **RAM Usage:** Warning > 80%, Critical > 95%
- **Disk Space:** Warning > 80%, Critical > 90%

Apprise Integration

If Apprise is enabled:

- Alerts sent to Apprise via HTTP POST
- Apprise forwards to configured channels
- Supports all Apprise notification providers
- Config file: `health_alarm_notify.conf`

Custom Alerts

Create custom alerts in `health.d/`:

```
# Example: Custom alert
alarm: custom_metric
```

```
    on: system.cpu
lookup: average -3m unaligned of user,system
 units: %
 every: 1m
  warn: $this > 75
  crit: $this > 90
 delay: down 5m multiplier 1.5 max 1h
  info: Custom CPU alert
    to: sysadmin
```

Parent-Child Streaming

Configuration

Stream metrics to a Netdata Director (parent server):

- Configured in `stream.conf`
- Requires director hostname/IP and API key
- Enables centralized monitoring of multiple servers
- Local dashboard remains available

Use Cases

- Multi-server monitoring
- Centralized dashboards
- Unified alerting
- Historical data aggregation

Environment Variables

Netdata Container

- `NETDATA_CLAIM_TOKEN` - Optional Netdata Cloud claim token
- `NETDATA_CLAIM_ROOMS` - Optional Netdata Cloud rooms
- `NETDATA_CLAIM_URL` - Netdata Cloud URL (default: https://app.netdata.cloud)
- `DOCKER_HOST` - Docker socket path (default: /var/run/docker.sock)

Configuration Files

Main Configuration

- `netdata.conf` - Main Netdata configuration
- `stream.conf` - Parent-child streaming configuration
- `health_alarm_notify.conf` - Alert notification configuration
- `health.d/*.conf` - Individual alert definitions

Customization

```
# Edit main config
nano /opt/speedbits/netdata-client/netdata/netdata.conf

# Edit alerts
nano /opt/speedbits/netdata-client/netdata/health.d/cpu_usage.conf

# Edit streaming
nano /opt/speedbits/netdata-client/netdata/stream.conf
```

Troubleshooting

Container Not Starting

```
docker logs netdata
docker ps -a | grep netdata
```

Missing Metrics

- Verify host mounts: `docker exec netdata ls /host/proc`
- Check Docker socket: `docker exec netdata ls /var/run/docker.sock`
- Verify container capabilities

Docker Containers Not Showing

- Verify Docker socket access

- Check Docker daemon is running
- Restart Netdata container

Alerts Not Working

- Verify Apprise is running and accessible
- Check `health_alarm_notify.conf` configuration
- Test alert thresholds
- Check Netdata logs for errors

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Authentication:** Add Basic Auth protection (critical!)
- **Network Security:** Restrict access via firewall or VPN
- **Resource Usage:** ~50MB RAM, minimal CPU impact
- **Data Retention:** Configure appropriate retention period
- **Alerting:** Configure multiple notification channels for redundancy
- **Streaming:** Use director for multi-server environments

Integration with Infinity Tools

Netdata complements Infinity Tools by providing:

- Real-time monitoring of all Infinity Tools applications
- Docker container monitoring for infrastructure
- System resource monitoring
- Alert integration with Apprise

Recommended Monitoring:

- All Infinity Tools application containers
- System resources (CPU, RAM, disk)
- Network traffic
- Docker daemon health

API & Automation

REST API

- Netdata provides REST API for metrics access
- API documentation available in web interface
- Useful for automation and integration
- Export metrics to external systems

Exporting Data

- Export graphs as images
- Export metrics as JSON/CSV
- Integrate with external monitoring systems

Next Steps

Netdata is now operational. Use it to:

- Monitor all Infinity Tools applications
- Track Docker container health
- Monitor system resources in real-time
- Set up alert notifications
- Analyze performance trends

For advanced features, API usage, custom collectors, and development guides, refer to the official Netdata documentation.

18: Netdata Director - Multi-Server Monitoring Hub

Netdata Director is a parent-child streaming architecture that enables centralized monitoring of multiple servers. The Director (parent) receives metrics streams from child nodes, providing a unified dashboard, centralized alerting, and long-term historical data retention for all monitored infrastructure.

⚠ **LICENSE REQUIREMENT:** Netdata Director is a Pro+ feature requiring a license. Community Netdata provides single-server monitoring only.

For advanced features, API documentation, and streaming configuration, see the official Netdata documentation.

Prerequisites

- **Pro+ License** - Required for Director functionality
- **Docker installed** (Chapter 3)
- **Docker Compose** (Chapter 3)
- **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- **Optional: Domain configured** (Chapter 4.5), e.g., `monitoring.example.com`
- **Optional: Apprise installed** (Chapter 5) for alert notifications
- **Multiple servers** - Director is designed for 2+ server environments

Installation via Infinity Tools

Menu Installation

```
APPLICATIONS → Netdata Director → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-netdata-director.sh --install
```

Architecture

Parent-Child Streaming

- **Director (Parent):** Central dashboard server receiving streams
- **Child Nodes:** Regular Netdata installations streaming to Director
- **Stream API Key:** Authentication token for child-to-parent connection
- **Unidirectional:** Metrics flow child → parent only

Container

- **netdata-director** - Director instance (netdata/netdata:latest)
- **netdata-director-ssl-proxy** - Nginx SSL proxy (standalone HTTPS mode only)

Data Persistence

- **Config:** `/opt/speedbits/netdata-director/netdata/`
- **Lib:** `/opt/speedbits/netdata-director/netdata/lib/` (metrics database)
- **Cache:** `/opt/speedbits/netdata-director/netdata/cache/`
- **API Key:** `/opt/speedbits/netdata-director/stream-api-key.txt`

Deployment Modes

Traefik Mode (Default)

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://monitoring.example.com`
- Security headers configured
- Requires: Traefik running, DNS A record configured

Standalone Mode

Direct access with HTTP or HTTPS (self-signed):

- HTTP: `http://SERVER_IP:19999`
- HTTPS: `https://SERVER_IP:19999` (self-signed cert via nginx proxy)
- Default port: 19999 (configurable)

Stream API Key

Generation

- 32-character random key generated during installation
- Saved in `stream-api-key.txt`
- Used for child node authentication
- Must be kept secret

Usage

Child nodes use this key to authenticate when streaming metrics:

- Configured in child node's `stream.conf`
- Director validates key before accepting streams
- Multiple children can use same key (or separate keys per child)

Streaming Configuration

Director Configuration

File: `/opt/speedbits/netdata-director/netdata/stream.conf`

```
[stream]
    enabled = no  # Director doesn't stream to anyone

[$STREAM_API_KEY]
    enabled = yes
    default memory mode = dbengine
    health enabled by default = auto
    default postpone alarms on connect seconds = 60
    default history = 3600
    allow from = *
```

Child Node Configuration

Configured during child node installation:

- Director hostname/IP
- Director port (default: 19999)
- Stream API key

Access Methods

Traefik Mode

```
https://monitoring.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

HTTP:

```
http://SERVER_IP:19999
```

HTTPS:

```
https://SERVER_IP:19999
```

Security Configuration

Access Security

- □ Traefik mode uses Let's Encrypt SSL (production-ready)
- □ Standalone HTTPS uses self-signed certificates
- □ Security headers configured
- ⚠ **NO default authentication** - Dashboard is publicly accessible
- ⚠ **Basic Auth incompatible** - Blocks child node streaming

Authentication Limitations

⚠ **CRITICAL:** Basic Auth cannot be used with Director because:

- Child nodes use HTTP API to stream metrics
- Basic Auth blocks unauthenticated API requests

- Child nodes cannot authenticate via Basic Auth
- Result: Child nodes cannot connect

Security Alternatives

- **Firewall Rules:** Restrict access to trusted IPs only
- **VPN Access:** Access Director via WireGuard VPN
- **Netdata Cloud:** Use official Netdata Cloud service
- **Network Isolation:** Keep Director on private network

Stream API Key Security

- API key provides authentication for child nodes
- Keep key secret (only share with trusted servers)
- Director validates key before accepting streams
- Can use separate keys per child node (advanced)

Alert Configuration

Apprise Integration

If Apprise is enabled, Director sends alerts for ALL child nodes:

- Centralized alert management
- Alerts include server hostname
- Single notification channel for all servers
- Config file: `health_alarm_notify.conf`

Alert Flow

1. Child node detects issue
2. Alert sent to Director
3. Director forwards to Apprise
4. Apprise sends to configured channels

Data Retention

Retention Periods

- **High-resolution:** 1 hour (1-second granularity)
- **Mid-resolution:** 1 day (1-minute granularity)
- **Low-resolution:** 30 days (15-minute granularity)

Storage

- Metrics stored in `dbengine` mode
- Configurable retention in `netdata.conf`
- Storage scales with number of child nodes

Child Node Connection

Connection Process

1. Install Netdata on child server (Chapter 17)
2. Enable streaming during installation
3. Provide Director hostname/IP
4. Provide Director port (default: 19999)
5. Provide Stream API key
6. Child node connects automatically

Connection Verification

- Wait 1-2 minutes for connection to establish
- Check Director dashboard dropdown for child node
- Verify metrics appearing in Director
- Check Director logs: `docker logs netdata-director`

Troubleshooting

Child Nodes Not Connecting

- Verify API key is correct
- Check network connectivity (firewall rules)
- Verify Director port is accessible
- Check child node logs for connection errors
- Check Director logs: `docker logs netdata-director`

Streaming Issues

- Verify `stream.conf` configuration
- Check API key matches between child and parent
- Verify Director is accepting connections
- Check network connectivity

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Security:** Implement firewall rules or VPN access (cannot use Basic Auth)
- **Network:** Ensure Director is accessible from all child nodes
- **Storage:** Plan storage capacity based on number of child nodes
- **API Key Management:** Use separate keys per child for enhanced security
- **Monitoring:** Monitor Director itself (resource usage, connectivity)

Advanced Configuration

Multiple API Keys

Create separate API keys for different child nodes:

```
# In stream.conf, add multiple sections:
[api-key-1]
    enabled = yes
    allow from = 192.168.1.10

[api-key-2]
    enabled = yes
    allow from = 192.168.1.20
```

IP Restrictions

Restrict which IPs can connect:

```
[$STREAM_API_KEY]
    enabled = yes
```

```
    allow from = 192.168.1.0/24  # Only allow from this subnet
```

Integration with Infinity Tools

Netdata Director complements Infinity Tools by providing:

- Centralized monitoring of all Infinity Tools servers
- Unified alerting for entire infrastructure
- Historical data retention for capacity planning
- Cross-server performance comparison

Next Steps

Netdata Director is now operational. Use it to:

- Connect child nodes from all your servers
- Monitor entire infrastructure from one dashboard
- Set up centralized alerting
- Analyze performance trends across servers
- Plan capacity based on historical data

For advanced features, streaming configuration, API usage, and development guides, refer to the official Netdata documentation.

19: WireGuard - VPN Infrastructure

WireGuard is a modern VPN protocol using ChaCha20 encryption and Curve25519 key exchange. This installation uses WG-Easy (WireGuard-UI) for web-based client management, providing a user-friendly interface for VPN administration while maintaining WireGuard's performance and security benefits.

For protocol specifications, advanced configuration, and technical documentation, see the official WireGuard documentation.

Prerequisites

- **Docker installed** (Chapter 3)
- **Docker Compose** (Chapter 3)
- **Linux kernel 5.6+** - WireGuard kernel module support
- **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- **Optional: Domain configured** (Chapter 4.5), e.g., `vpn.example.com`
- **Firewall access** - Ability to open UDP port

Installation via Infinity Tools

Menu Installation

```
APPLICATIONS → WireGuard → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-wireguard.sh --install


# With domain (Traefik mode)
export WG_DOMAIN="vpn.example.com"
export WG_USE_TRAEFIK="true"
```

```
sudo -E bash /opt/InfinityTools/Solutions/setup-wireguard.sh --install

# Custom networks
export VPN_NETWORK_BASE="192.168.100"
export HOST_NETWORK_BASE="192.168.101"
export WG_VPN_PORT="51820"
sudo -E bash /opt/InfinityTools/Solutions/setup-wireguard.sh --install
```

Architecture

Containers

- **wireguard** - WG-Easy (WireGuard-UI) container (ngoduykhanh/wireguard-ui:latest)
- **wireguard-https** - Nginx SSL proxy (standalone HTTPS mode only)

Network Architecture

- **VPN Network:** Default `10.13.13.0/24` (configurable)
  - WireGuard server: `10.13.13.1`
  - Admin client: `10.13.13.2`
  - Client IPs: `10.13.13.3+` (auto-assigned)
- **Host Network:** Default `10.13.14.0/24` (configurable)
  - Host services IP: `10.13.14.1`
  - Accessible via VPN for host service access

Data Persistence

- **Data:** `/opt/speedbits/wireguard/data/` (WG-Easy database, client configs)
- **Configs:** `/opt/speedbits/wireguard/wg_confs/` (WireGuard config files)
- **SSL:** `/opt/speedbits/wireguard/ssl/` (standalone mode certificates)
- **Password:** `/opt/speedbits/wireguard/web-password.txt`

Host Integration

- **Kernel Module:** WireGuard kernel module loaded on host
- **Systemd Service:** `wireguard-host-network.service` for host network persistence
- **Network Interface:** `wg-host` dummy interface for host network
- **iptables Rules:** NAT and forwarding rules for VPN ↔ Host network

Deployment Modes

Traefik Mode

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://vpn.example.com`
- Security headers configured
- Requires: Traefik running, DNS A record configured

Standalone Mode (Default)

Direct access with HTTPS (self-signed):

- HTTPS: `https://SERVER_IP:8445` (self-signed cert via nginx proxy)
- Default web UI port: 8445 (configurable)
- VPN port: 51820 UDP (configurable)
- No domain required

Installation Process

Configuration Steps

1. **Network Configuration:** VPN network base (default: 10.13.13) and Host network base (default: 10.13.14)
2. **DNS Configuration:** Auto-detected from server's `/etc/resolv.conf`
3. **SSL Mode Selection:** Choose Traefik or Standalone
4. **VPN Port:** UDP port for VPN connections (default: 51820)
5. **Server Endpoint:** Public IP or domain name for client connections
6. **Kernel Module:** WireGuard kernel module installed and loaded
7. **Systemd Service:** Host network service created and enabled

What Gets Created

- **Directory:** `/opt/speedbits/wireguard`
- **Containers:** `wireguard`, `wireguard-https` (standalone mode)
- **Docker Compose:** `/opt/speedbits/wireguard/docker-compose.yml`
- **Systemd Service:** `/etc/systemd/system/wireguard-host-network.service`

- **Host Scripts:** `host-network-setup.sh`, `host-network-cleanup.sh`

Access Methods

Traefik Mode

```
https://vpn.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

```
https://SERVER_IP:8445
```

Accept self-signed certificate warning (Advanced → Proceed).

Authentication

Web UI Credentials

- **Username:** `admin` (fixed)
- **Password:** Randomly generated (20 characters)
- **Storage:** `/opt/speedbits/wireguard/web-password.txt`
- **Hash:** bcrypt hash stored in container environment

VPN Client Authentication

- Each client gets unique public/private key pair
- Server validates client public key
- No username/password required for VPN connection
- Keys generated cryptographically secure

Network Configuration

VPN Network (10.13.13.0/24)

- **Purpose:** WireGuard clients and Docker services
- **Server IP:** `10.13.13.1`
- **Client IPs:** `10.13.13.3+` (auto-assigned)
- **Routing:** Clients can access Docker containers via container names

Host Network (10.13.14.0/24)

- **Purpose:** Access host services via VPN
- **Host IP:** `10.13.14.1`
- **Interface:** `wg-host` dummy interface
- **Routing:** NAT and forwarding rules configured

iptables Rules

```
# NAT for VPN → Host network
iptables -t nat -A POSTROUTING -s 10.13.13.0/24 -d 10.13.14.0/24 -j MASQUERADE

# Forwarding rules
iptables -A FORWARD -s 10.13.13.0/24 -d 10.13.14.0/24 -j ACCEPT
iptables -A FORWARD -s 10.13.14.0/24 -d 10.13.13.0/24 -j ACCEPT
```

Environment Variables

WireGuard Container

- `WGUI_USERNAME` - Web UI username (default: admin)
- `WGUI_PASSWORD` - Web UI password (randomly generated)
- `WGUI_SERVER_INTERFACE_ADDRESSES` - Server IP/CIDR (e.g., 10.13.13.1/24)
- `WGUI_SERVER_LISTEN_PORT` - VPN UDP port (default: 51820)
- `WGUI_DEFAULT_CLIENT_ALLOWED_IPS` - Default allowed IPs for clients
- `WGUI_DEFAULT_CLIENT_USE_SERVER_DNS` - Use server DNS (default: true)
- `WGUI_MANAGE_START` - Auto-start WireGuard (default: true)
- `WGUI_MANAGE_RESTART` - Auto-restart WireGuard (default: true)
- `SESSION_SECRET` - Session encryption key (randomly generated)

Client Management

Web UI Features

- Add/remove clients via web interface
- Generate QR codes for mobile devices
- Download .conf files for desktop clients
- Enable/disable clients individually
- View connection statistics
- Monitor traffic usage

Client Configuration

Clients are created via web UI. Each client gets:

- Unique public/private key pair
- Auto-assigned IP address
- Pre-configured AllowedIPs (VPN + Host networks)
- Server endpoint and public key

Security Configuration

Encryption

- **Cipher:** ChaCha20 (symmetric encryption)
- **Key Exchange:** Curve25519 (elliptic curve)
- **Hash:** BLAKE2s
- **Handshake:** Noise protocol framework

Access Security

- Traefik mode uses Let's Encrypt SSL (production-ready)
- Standalone HTTPS uses self-signed certificates (acceptable for internal use)
- Security headers configured (X-Frame-Options, CSP, etc.)
- Random password generation (20 characters)
- Unique keys per client

Container Security

- Requires `NET_ADMIN` and `SYS_MODULE` capabilities
- IP forwarding enabled
- Kernel module access for WireGuard
- Host network access for routing

Firewall Configuration

Required Ports

- **UDP 51820** (or custom VPN port) - VPN connections (MUST be open)
- **TCP 8445** (standalone mode) - Web UI (optional, can be closed after VPN setup)
- **TCP 443** (Traefik mode) - Web UI via Traefik

Firewall Best Practices

```
# Open VPN port (REQUIRED)
sudo ufw allow 51820/udp

# Close other public ports (access via VPN instead)
sudo ufw delete allow 8443  # Webmin
sudo ufw delete allow 8444  # Apprise
sudo ufw delete allow 8445  # WireGuard web UI
```

Systemd Service

Host Network Service

Service: `wireguard-host-network.service`

- Creates `wg-host` dummy interface
- Configures host network IP (10.13.14.1)
- Sets up routing and iptables rules
- Persists across reboots

Service Management

```
# Check status
systemctl status wireguard-host-network.service

# Restart service
sudo systemctl restart wireguard-host-network.service
```

```
# View logs
journalctl -u wireguard-host-network.service
```

Troubleshooting

VPN Connection Issues

- Verify firewall: `sudo ufw status | grep 51820`
- Check server endpoint accessibility
- Verify client config (AllowedIPs, endpoint, keys)
- Check WireGuard logs: `docker logs wireguard`
- Test UDP connectivity: `nc -u -v SERVER_IP 51820`

Host Network Issues

- Check systemd service: `systemctl status wireguard-host-network.service`
- Verify interface: `ip addr show wg-host`
- Check routing: `ip route show | grep 10.13.14`
- Verify iptables rules: `iptables -t nat -L -n -v`

Web UI Issues

- Check container status: `docker ps | grep wireguard`
- View logs: `docker logs wireguard`
- Verify password: `cat /opt/speedbits/wireguard/web-password.txt`
- Test API: `curl -u admin:PASSWORD http://localhost:5000/api/sessions`

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Firewall:** Open only VPN port, close other public ports
- **Password Management:** Store web UI password securely
- **Client Management:** Regularly review and disable unused clients
- **Monitoring:** Monitor VPN connections and traffic
- **Backup:** Backup client configurations and keys
- **Updates:** Re-run install script periodically for updates

Integration with Infinity Tools

WireGuard complements Infinity Tools by providing:

- Secure remote access to all Infinity Tools applications
- Access to Docker services without exposing ports publicly
- Access to host services (Webmin, Apprise) securely
- Centralized VPN management via web interface

Recommended Setup:

- Open only VPN port (UDP 51820) publicly
- Close other public ports (Webmin, Apprise, etc.)
- Access all services via VPN
- Use VPN network (10.13.13.x) for Docker services
- Use Host network (10.13.14.1) for host services

Advanced Configuration

Custom Networks

Configure custom network ranges:

```
export VPN_NETWORK_BASE="192.168.100"
export HOST_NETWORK_BASE="192.168.101"
sudo -E bash setup-wireguard.sh --install
```

Custom VPN Port

```
export WG_VPN_PORT="51821"
sudo -E bash setup-wireguard.sh --install
```

Custom DNS

```
export VPN_DNS="8.8.8.8,8.8.4.4"
sudo -E bash setup-wireguard.sh --install
```

Client Configuration Export

Via Web UI

- Download .conf file for desktop clients
- Scan QR code for mobile clients
- View client details (IP, public key, etc.)

Via Command Line

```
# View admin client config
sudo bash setup-wireguard.sh --show-config

# Client configs stored in
ls /opt/speedbits/wireguard/data/peer_*/peer.conf
```

Next Steps

WireGuard is now operational. Use it to:

- Create VPN clients for all your devices
- Access Infinity Tools applications securely
- Access host services without exposing ports
- Manage clients via web interface
- Monitor VPN connections and traffic

For advanced features, protocol specifications, and development guides, refer to the official WireGuard documentation.

20: Warpgate - SSH Bastion Host

Warpgate is a modern SSH/RDP bastion host providing centralized access control, session recording, and web-based management. It acts as a gateway for all SSH connections, reducing the attack surface by eliminating direct server access.

For protocol specifications, advanced configuration, and technical documentation, see the official Warpgate documentation.

Prerequisites

- **Docker installed** (Chapter 3)
- **Docker Compose** (Chapter 3)
- **Optional: Traefik installed** (Chapter 4) for HTTPS with Let's Encrypt
- **Optional: Domain configured** (Chapter 4.5), e.g., `warpgate.example.com`
- **Firewall access** - Ability to open port 2222 (SSH bastion)

Installation via Infinity Tools

Menu Installation

```
APPLICATIONS → Warpgate → Install
```

CLI Installation

```
sudo bash /opt/InfinityTools/Solutions/setup-warpgate.sh --install

# With domain (Traefik mode)
export WARPGATE_DOMAIN="warpgate.example.com"
export WG_USE_TRAEFIK="true"
sudo -E bash /opt/InfinityTools/Solutions/setup-warpgate.sh --install
```

```
# Standalone mode
export WG_USE_TRAEFIK="false"
export WG_PORT="8888"
sudo -E bash /opt/InfinityTools/Solutions/setup-warpgate.sh --install

# With domain argument
sudo bash /opt/InfinityTools/Solutions/setup-warpgate.sh --install warpgate.example.com
```

Architecture

Container

- **warpgate** - Warpgate container (ghcr.io/warp-tech/warpgate:latest)

Ports

- **2222** - SSH bastion port (exposed directly, TCP)
- **8888** - Web interface port (via Traefik or standalone, HTTPS)

Data Persistence

- **Data:** `/opt/speedbits/warpgate/data/` (configuration, database)
- **Config:** `/opt/speedbits/warpgate/data/warpgate.yaml` (main configuration)
- **Database:** `/opt/speedbits/warpgate/data/db/` (SQLite database)
- **SSL:** `/opt/speedbits/warpgate/ssl/` (standalone mode certificates)

Deployment Modes

Traefik Mode (Default)

Uses Traefik for SSL termination and domain routing:

- Automatic Let's Encrypt certificate provisioning
- Domain-based access: `https://warpgate.example.com`
- SSH bastion: `ssh -p 2222 user@warpgate.example.com`
- Requires: Traefik running, DNS A record configured

Standalone Mode

Direct access with HTTPS (self-signed):

- HTTPS: `https://SERVER_IP:8888` (self-signed cert)
- SSH bastion: `ssh -p 2222 user@SERVER_IP`
- Default web UI port: 8888 (configurable)
- No domain required

Installation Process

Configuration Steps

1. **SSL Mode Selection:** Choose Traefik or Standalone
2. **Domain Configuration:** If Traefik, specify domain (e.g., warpgate.example.com)
3. **Port Configuration:** If Standalone, specify web UI port (default: 8888)
4. **Container Creation:** Warpgate container created and started
5. **Interactive Setup:** Admin account creation via `warpgate setup` command

What Gets Created

- **Directory:** `/opt/speedbits/warpgate`
- **Container:** `warpgate`
- **Docker Compose:** `/opt/speedbits/warpgate/docker-compose.yml`
- **Configuration:** `/opt/speedbits/warpgate/data/warpgate.yaml`
- **Database:** SQLite database in `/opt/speedbits/warpgate/data/db/`

Access Methods

Traefik Mode

```
# Web interface
https://warpgate.example.com


# SSH bastion
ssh -p 2222 user@warpgate.example.com
```

Direct web access after DNS propagation and SSL certificate generation (30-60 seconds).

Standalone Mode

```
# Web interface
https://SERVER_IP:8888

# SSH bastion
ssh -p 2222 user@SERVER_IP
```

Accept self-signed certificate warning (Advanced → Proceed).

Initial Setup

Admin Account Creation

After container creation, Warpgate runs interactive setup:

```
docker run --rm -it \
  -v /opt/speedbits/warpgate/data:/data \
  ghcr.io/warp-tech/warpgate:latest \
  setup
```

Prompts:

- **Admin username:** Username for admin account
- **Admin password:** Password for admin account
- **Confirm password:** Password confirmation

Configuration File

After setup, configuration is stored in:

```
/opt/speedbits/warpgate/data/warpgate.yaml
```

File permissions: `600` (owner: uid 1000)

Authentication

Web Interface Authentication

- Username/password authentication
- Admin account created during setup
- Additional users created via web interface

SSH Bastion Authentication

- Warpgate username/password authentication
- After authentication, user selects target
- Warpgate connects to target using configured credentials

Target Configuration

Adding Targets

Targets are servers that users can connect to through Warpgate:

- **Name:** Friendly name for the target
- **Host:** IP address or hostname (use `localhost` for same server)
- **Port:** SSH port (usually 22)
- **Username:** SSH username for the target
- **Key-based auth:** Optional SSH key configuration

Same-Server Target

For accessing the server where Warpgate runs:

- **Host:** `localhost` or `127.0.0.1`
- **Port:** `22` (or custom SSH port)
- **Username:** Server username

User Management

Web Interface

- Create users via web interface
- Assign access to specific targets
- Manage user permissions

- View user sessions

User Access Control

- Users can only access targets they're granted access to
- Access can be granted/revoked per user per target
- Session recording available per user/target

SSH Connection Flow

Connection Process

1. Client connects to Warpgate on port 2222
2. Warpgate authenticates user (username/password)
3. Warpgate presents available targets
4. User selects target
5. Warpgate connects to target using configured credentials
6. Session is established and optionally recorded

SSH Command

```
# Traefik mode
ssh -p 2222 warpgate-user@warpgate.example.com


# Standalone mode
ssh -p 2222 warpgate-user@SERVER_IP
```

Security Configuration

Access Security

- Traefik mode uses Let's Encrypt SSL (production-ready)
- Standalone HTTPS uses self-signed certificates (acceptable for internal use)
- SSH bastion port (2222) exposed directly
- Direct SSH port (22) can be closed after Warpgate setup

Firewall Best Practices

```
# Open Warpgate SSH bastion port
sudo ufw allow 2222/tcp

# Close direct SSH access (after testing Warpgate)
sudo ufw delete allow 22/tcp

# Open web interface port (if standalone)
sudo ufw allow 8888/tcp
```

Container Security

- Runs as uid 1000 (non-root)
- Data directory mounted with proper permissions
- Configuration file secured (600 permissions)

Environment Variables

Standalone Mode

- `WARPGATE_HTTP_LISTEN` - HTTP listen address (default: 0.0.0.0:8888)
- `WARPGATE_SSH_LISTEN` - SSH listen address (default: 0.0.0.0:2222)

Troubleshooting

Web Interface Issues

- Check container status: `docker ps | grep warpgate`
- View logs: `docker logs warpgate`
- Verify configuration: `cat /opt/speedbits/warpgate/data/warpgate.yaml`
- Check file permissions: `ls -la /opt/speedbits/warpgate/data/`

SSH Connection Issues

- Verify firewall: `sudo ufw status | grep 2222`
- Test connectivity: `nc -v SERVER_IP 2222`
- Check user credentials in web interface
- Verify target configuration

- Check user access permissions

Target Connection Failures

- Verify target host/IP is correct
- Check target SSH port
- Verify target username
- Test direct connection to target
- Check SSH key configuration (if using key-based auth)

Production Considerations

- **Access Method:** Use Traefik mode for production (trusted SSL)
- **Firewall:** Close direct SSH port (22) after Warpgate verification
- **User Management:** Regularly review and remove unused users
- **Session Recording:** Enable for security auditing
- **Monitoring:** Monitor SSH sessions and access patterns
- **Backup:** Backup configuration and database regularly
- **Updates:** Re-run install script periodically for updates

Integration with Infinity Tools

Warpgate complements Infinity Tools by providing:

- Centralized SSH access management
- Secure gateway for all server access
- Session recording and auditing
- User access control

Recommended Setup:

- Open only Warpgate SSH port (2222) publicly
- Close direct SSH port (22) after testing
- Use Traefik for web interface HTTPS
- Enable session recording for security
- Regularly audit user access

Advanced Configuration

Custom Ports

```
# Custom web UI port (standalone)
export WG_PORT="9999"
sudo -E bash setup-warpgate.sh --install

# SSH port is always 2222 (exposed directly)
```

Configuration File

Edit configuration directly:

```
# Backup first
cp /opt/speedbits/warpgate/data/warpgate.yaml
/opt/speedbits/warpgate/data/warpgate.yaml.backup

# Edit configuration
nano /opt/speedbits/warpgate/data/warpgate.yaml

# Restart container
docker restart warpgate
```

Session Recording

Enabling Recording

Session recording can be enabled per user or per target in the web interface. Recorded sessions are stored in the database and can be reviewed for security auditing.

User Management Script

Infinity Tools provides a helper script for creating system users:

```
sudo bash /opt/InfinityTools/Infrastructure/add-warpgate-user.sh
```

This script creates a `warpgate` system user with SSH key access. Edit the script to add your SSH public key before running.

Next Steps

Warpgate is now operational. Use it to:

- Add targets (servers) users can connect to
- Create users and assign access
- Connect via SSH through Warpgate
- Monitor sessions and access
- Close direct SSH access for better security

For advanced features, API documentation, and development guides, refer to the official Warpgate documentation.