4. Traefik - Reverse Proxy Configuration

Traefik serves as the reverse proxy and SSL termination point for all Infinity Tools services. This chapter covers the installation, configuration, and management of Traefik within the Infinity Tools ecosystem. For advanced configuration and troubleshooting, refer to the official Traefik documentation.

Traefik Architecture Overview

Traefik provides the following core functionality:

- **SSL/TLS Termination** - Automatic Let's Encrypt certificate management
- **Reverse Proxy** - Request routing based on Host headers
- **Load Balancing** - Distribution of traffic across service instances
- **Service Discovery** - Automatic detection of Docker containers
- **Middleware Support** - Security headers, authentication, rate limiting

Installation Process

Via Infinity Tools Menu

Navigate to the Infinity Tools menu and select:

```
SECURITY & NETWORKING → Install Traefik
```

Command Line Installation

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

# With environment variables
export ACME_EMAIL="admin@domain.com"
```

```
export PROXY_NETWORK="proxy"
sudo -E bash /opt/InfinityTools/Solutions/setup-traefik.sh --install
```

Configuration Parameters

Required Configuration

During installation, you'll be prompted for:

- **ACME Email:** Email address for Let's Encrypt certificate notifications
- **Domain Name:** Primary domain for SSL certificate generation
- **IPv6 Support:** Enable/disable IPv6 for ACME challenges
- **Network Configuration:** Docker network for service communication

Environment Variables

```
# Optional environment variables
export ACME_EMAIL="admin@domain.com"        # Let's Encrypt email
export PROXY_NETWORK="proxy"                # Docker network name
export TRAEFIK_DOMAIN="traefik.domain.com"  # Traefik dashboard domain
export TRAEFIK_PORT="8080"                  # Dashboard port (if enabled)
```

Generated Configuration

Traefik Configuration File

Location: `/opt/speedbits/traefik/traefik.yml`

```
entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
```

```
          permanent: true

  websecure:
    address: ":443"

certificatesResolvers:
  myresolver:
    acme:
      email: admin@domain.com
      storage: /letsencrypt/acme.json
      httpChallenge:
        entryPoint: web

providers:
  docker:
    exposedByDefault: false

serversTransport:
  insecureSkipVerify: true

global:
  checkNewVersion: false
  sendAnonymousUsage: false
```

Docker Compose Configuration

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

```
version: '3.8'

services:
  traefik:
    image: traefik:v3.0
    container_name: traefik
    command:
      - "--configFile=/traefik.yml"
    ports:
      - "80:80"
      - "443:443"
```

```
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /opt/speedbits/traefik/traefik.yml:/traefik.yml:ro
      - /opt/speedbits/traefik/letsencrypt:/letsencrypt
    restart: unless-stopped
    networks:
      - proxy

networks:
  proxy:
    external: true
```

SSL Certificate Management

Let's Encrypt Integration

Traefik automatically manages SSL certificates using Let's Encrypt:

- **HTTP-01 Challenge:** Validates domain ownership via HTTP
- **Automatic Renewal:** Certificates are renewed automatically
- **Wildcard Support:** Supports wildcard certificates via DNS challenge
- **Certificate Storage:** Stored in `/opt/speedbits/traefik/letsencrypt/`

Certificate Monitoring

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

# View certificate files
ls -la /opt/speedbits/traefik/letsencrypt/

# Check certificate expiration
openssl x509 -in /opt/speedbits/traefik/letsencrypt/acme.json -text -noout
```

Service Integration

Automatic Service Discovery

Traefik automatically discovers services with the following labels:

```
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.service-name.rule=Host(`service.domain.com`)"
  - "traefik.http.routers.service-name.entrypoints=websecure"
  - "traefik.http.routers.service-name.tls.certresolver=myresolver"
  - "traefik.http.services.service-name.loadbalancer.server.port=8080"
```

Network Requirements

Services must be connected to the same Docker network as Traefik:

```
networks:
  - proxy

# Ensure network exists
docker network create proxy
```

Security Configuration

Security Headers

Traefik can be configured with security middleware:

```
labels:
  - "traefik.http.middlewares.security-headers.headers.customResponseHeaders.X-Content-Type-Options=nosniff"
  - "traefik.http.middlewares.security-headers.headers.customResponseHeaders.X-Frame-Options=SAMEORIGIN"
  - "traefik.http.middlewares.security-headers.headers.customResponseHeaders.X-XSS-Protection=1; mode=block"
  - "traefik.http.routers.service-name.middlewares=security-headers"
```

Access Control

Basic authentication can be configured for services:

```
# Generate password hash
echo $(htpasswd -nb admin password) | sed -e s/\\$/\\$\\$/g

# Apply to service
labels:
  - "traefik.http.middlewares.auth.basicauth.users=admin:$$2y$$10$$..."
  - "traefik.http.routers.service-name.middlewares=auth"
```

Monitoring and Logging

Traefik Dashboard

The Traefik dashboard can be enabled for monitoring:

```
# Add to docker-compose.yml
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.dashboard.rule=Host(`traefik.domain.com`)"
  - "traefik.http.routers.dashboard.tls.certresolver=myresolver"
  - "traefik.http.routers.dashboard.service=api@internal"
```

Logging Configuration

```
# Add to traefik.yml
log:
  level: INFO
  filePath: /var/log/traefik.log

accessLog:
  filePath: /var/log/access.log
  format: json
```

Performance Optimization

Resource Limits

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

Caching Configuration

```
# Add to traefik.yml
http:
  middlewares:
    cache:
      headers:
        customRequestHeaders:
          Cache-Control: "max-age=3600"
```

Troubleshooting

Common Issues

Certificate Generation Fails:

```
# Check domain DNS resolution
dig domain.com
nslookup domain.com

# Verify port 80 accessibility
telnet domain.com 80

# Check Traefik logs
```

```
docker logs traefik | grep -i acme
```

Service Not Accessible:

```
# Check service labels
docker inspect service-name | grep -A 10 Labels

# Verify network connectivity
docker network inspect proxy

# Check Traefik routing
curl -H "Host: service.domain.com" http://localhost
```

Performance Issues:

```
# Monitor resource usage
docker stats traefik

# Check connection limits
ss -tulnp | grep :443

# Review access logs
tail -f /opt/speedbits/traefik/logs/access.log
```

Debugging Commands

```
# Check Traefik configuration
docker exec traefik traefik version

# Test configuration
docker exec traefik traefik --configFile=/traefik.yml --logLevel=DEBUG

# View active routes
curl -s http://localhost:8080/api/http/routers | jq

# Check certificate status
docker exec traefik cat /letsencrypt/acme.json | jq
```

Backup and Recovery

Configuration Backup

```
# Backup Traefik configuration
tar -czf traefik-backup.tar.gz -C /opt/speedbits/traefik .

# Backup SSL certificates
cp -r /opt/speedbits/traefik/letsencrypt/ /backup/traefik-certs/
```

Disaster Recovery

```
# Restore configuration
tar -xzf traefik-backup.tar.gz -C /opt/speedbits/traefik/

# Restart Traefik
cd /opt/speedbits/traefik
docker compose down
docker compose up -d
```

Integration with Other Services

Service Dependencies

Most Infinity Tools services check for Traefik availability:

```
# Service installation checks
if ! docker ps --format '{{.Names}}' | grep -q "^traefik$"; then
    echo "Traefik is not running!"
    echo "Please install Traefik first"
    exit 1
fi
```

Network Integration

Services automatically join the proxy network:

```
networks:
  proxy:
    external: true
    name: proxy
```

Advanced Configuration

Custom Middleware

```
# Rate limiting
labels:
  - "traefik.http.middlewares.ratelimit.ratelimit.burst=100"
  - "traefik.http.middlewares.ratelimit.ratelimit.average=50"

# IP whitelisting
labels:
  - "traefik.http.middlewares.ipwhitelist.ipwhitelist.sourcerange=192.168.1.0/24"
```

Load Balancing

```
labels:
  - "traefik.http.services.service-name.loadbalancer.server.port=8080"
  - "traefik.http.services.service-name.loadbalancer.healthcheck.path=/health"
  - "traefik.http.services.service-name.loadbalancer.healthcheck.interval=30s"
```

Next Steps

With Traefik installed and configured, you can now deploy applications that will automatically integrate with the reverse proxy system.

Verification Checklist

- [ ] Traefik container running and healthy
- [ ] SSL certificates generated for configured domains
- [ ] HTTP to HTTPS redirection working
- [ ] Docker network connectivity verified

- Service discovery functioning

Next: Application Deployment and Management (Coming Soon)

Revision #4
Created 29 October 2025 16:58:21 by bjoern
Updated 17 November 2025 16:33:26 by bjoern